ML Methods

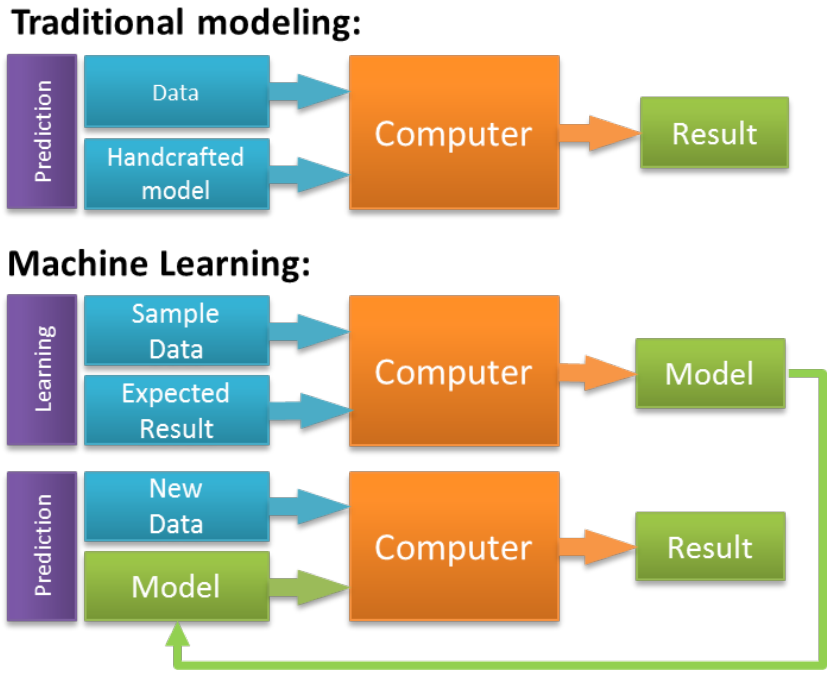
E. Le Pennec

Fall 2019

Outline

Outline

Machine Learning

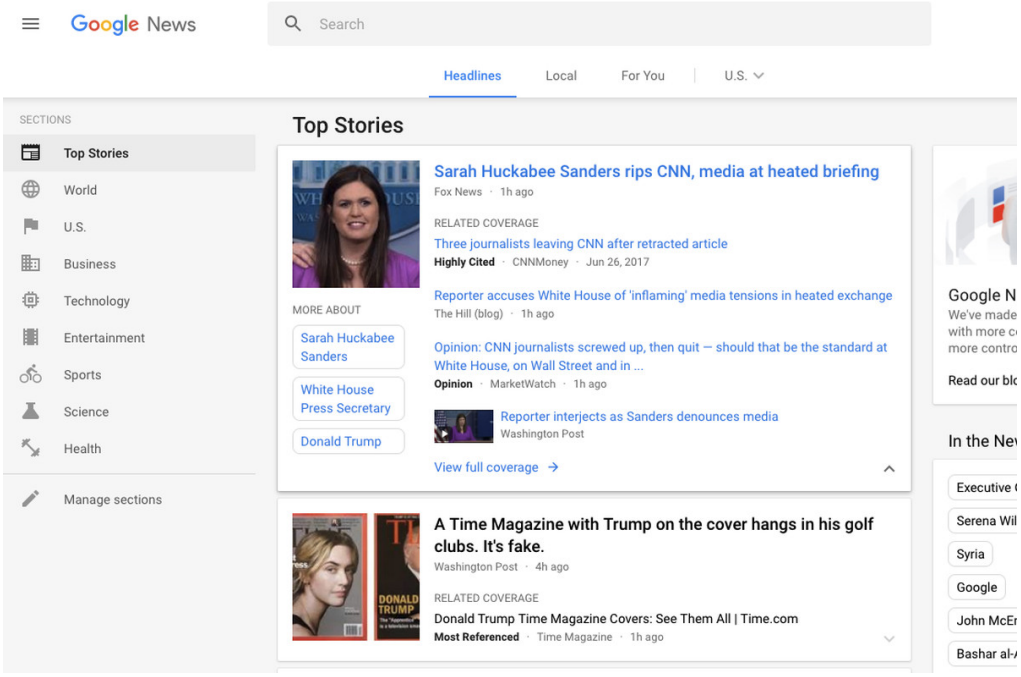


A definition by Tom Mitchell (http://www.cs.cmu.edu/~tom/)

A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T , as measured by P, improves with experience E.

Source: azavea

News Clustering

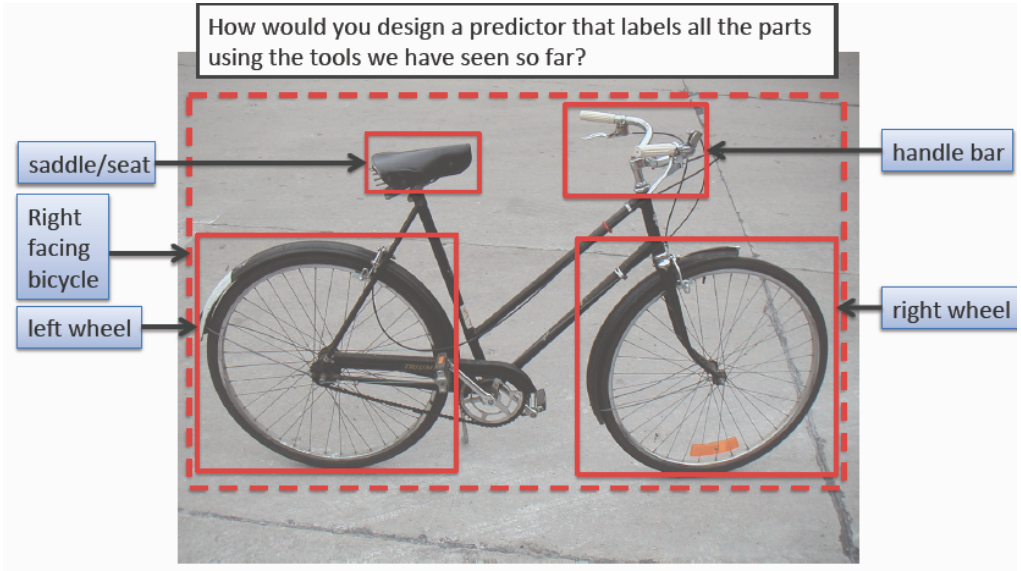


A news clustering algorithm:

- **Task**: group article corresponding to the same news
- **Performance**: quality of the clusters
- **Experience**: set of articles

Source: theverge.com

Object Recognition



A detection/recognition algorithm:

- **Task**: say if an object is present or not in the image
- **Performance**: number of errors
- **Experience**: set of previously seen labeled images

Source: Unknown

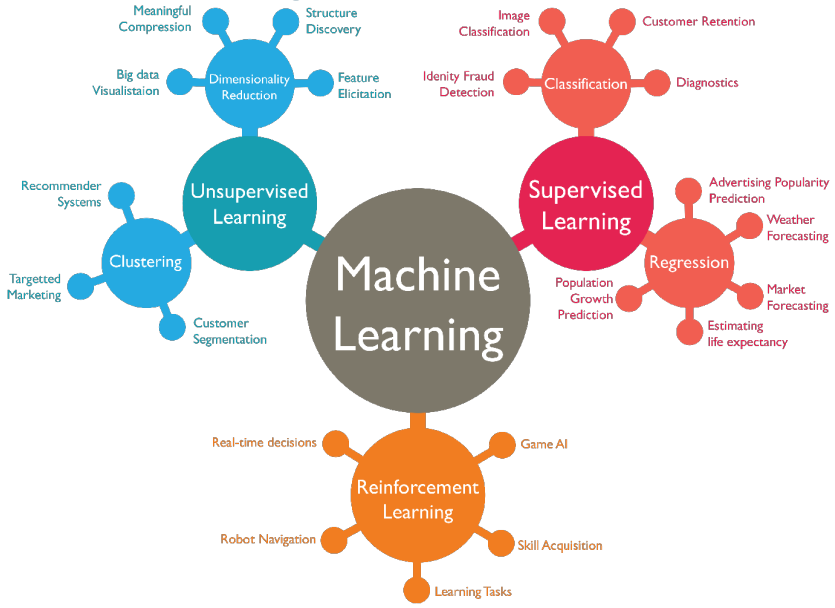
A Robot that Learns

A robot endowed with a set of sensors and an online learning algorithm:

- **Task**: play football
- **Performance**: score evolution
- **Experience**:
  - current environment and outcome,
  - past games

Source: Rutgers Prep School

Three Kinds of Learning



Unsupervised Learning

- **Task:** Clustering/DR
- **Performance:** Quality
- **Experience:** Raw dataset (No Ground Truth)

Supervised Learning

- **Task:** Prediction
- **Performance:** Average error
- **Experience:** Predictions (Ground Truth)

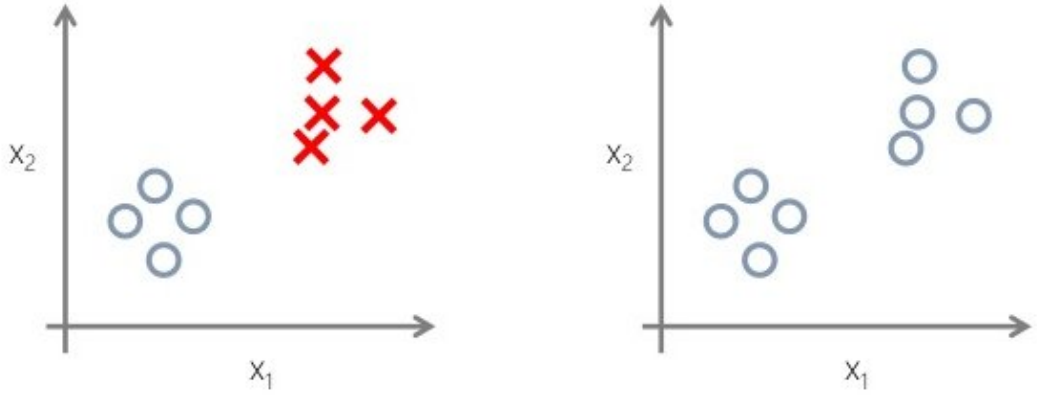
Reinforcement Learning

- **Task:** Action
- **Performance:** Total reward
- **Experience:** Reward from env. (Interact. with env.)

- **Timing:** Offline/Batch (learning from past data) vs Online (continuous learning)
- **Implicit stationarity assumption: Tomorrow is the same as yesterday!**

Source: A. Wahid

Supervised and Unsupervised



Supervised Learning (Imitation)

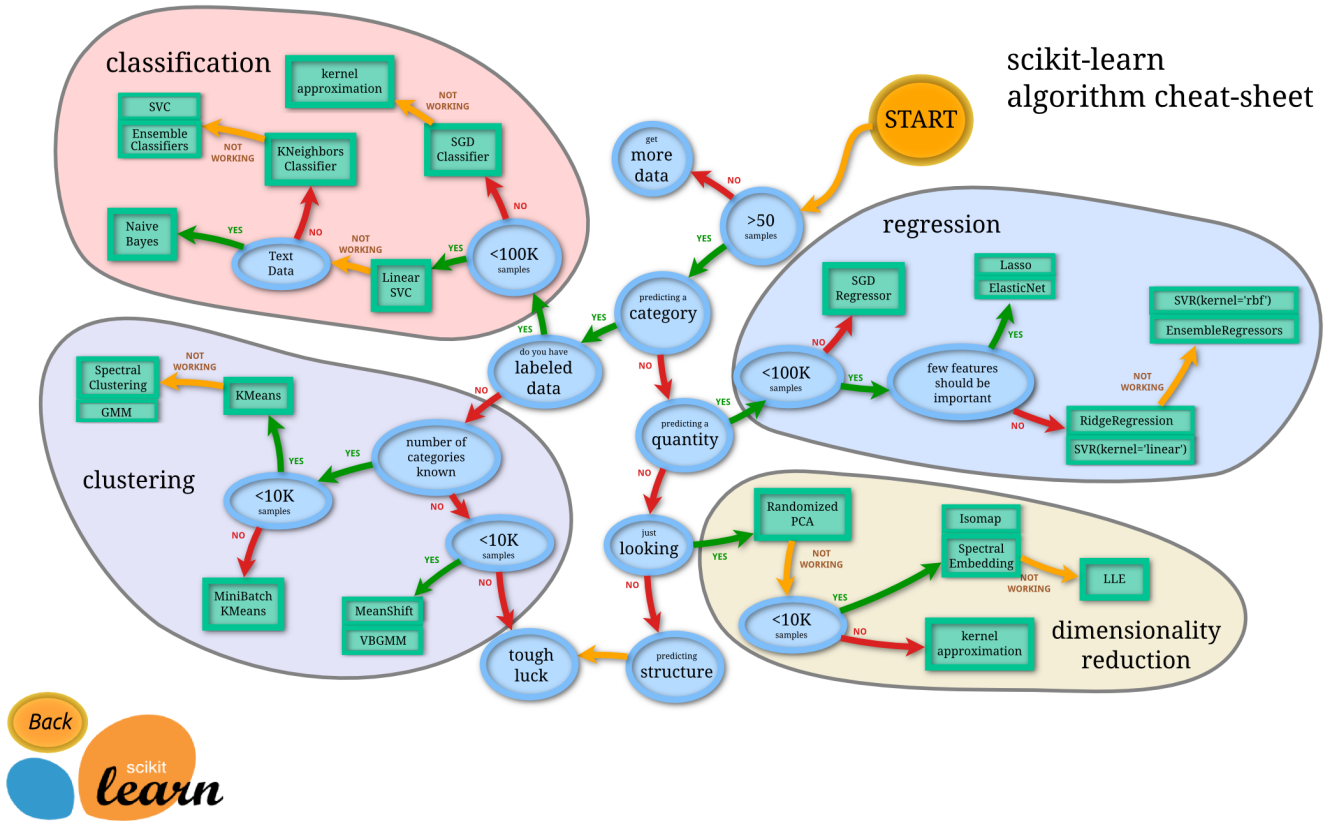
- **Goal:** Learn a function f predicting a variable Y from an individual $\underline{X}$.
- **Data:** Learning set with labeled examples $(\underline{X}_i, Y_i)$
- **Assumption:** Future data behaves as past data!
- **Predicting is not explaining!**

Unsupervised Learning (Structure Discovery)

- **Goal:** Discover a structure within a set of individuals $(\underline{X}_i)$.
- **Data:** Learning set with unlabeled examples $(\underline{X}_i)$
- Unsupervised learning is not a well-posed setting....

Source: KDnuggets

Machine Learning

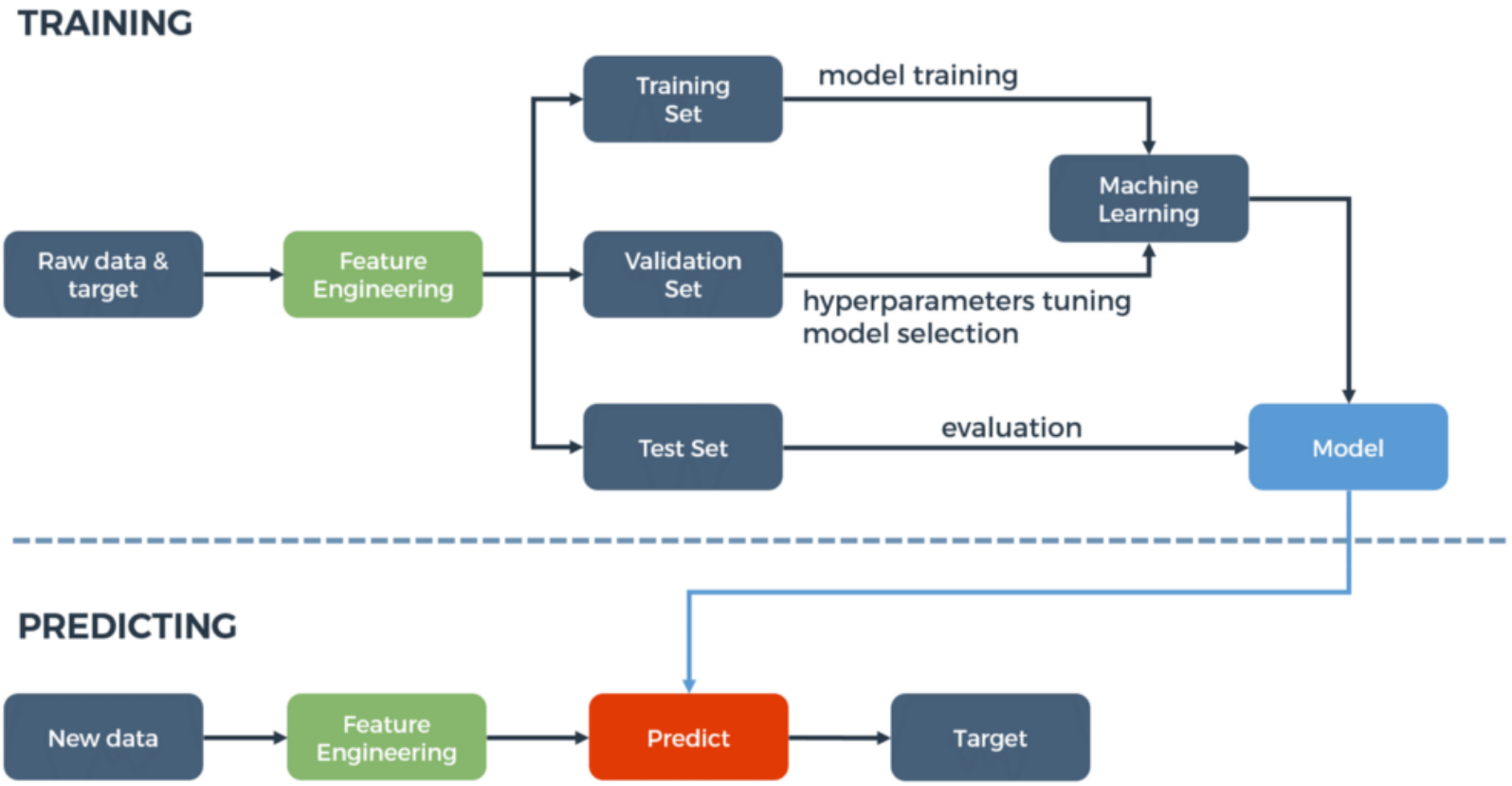


ML Methods

- Huge catalog of methods,
- Need to define the performance,
- Feature design...

Source: scikit-learn.org

ML Pipeline



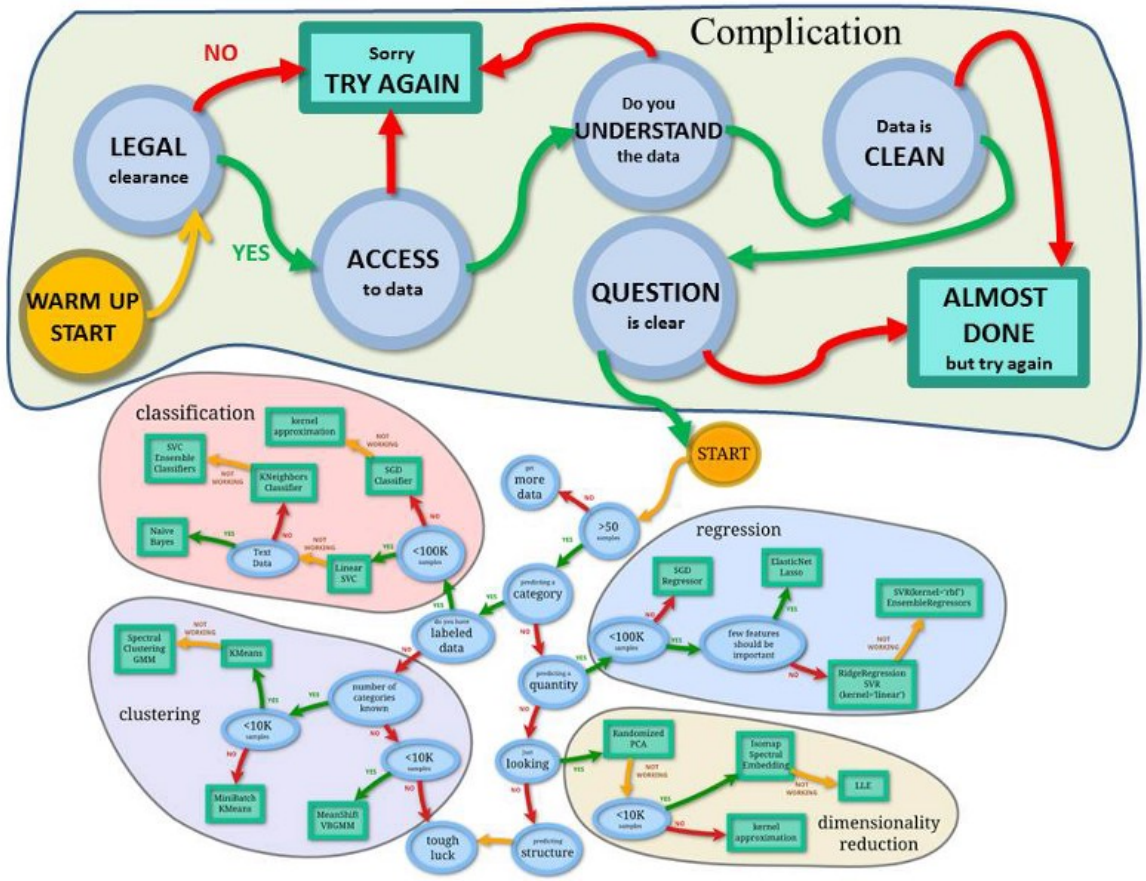
Learning pipeline

- Test and compare models.

- Deployment pipeline is different!

Source: CDiscount

DS ≠ ML



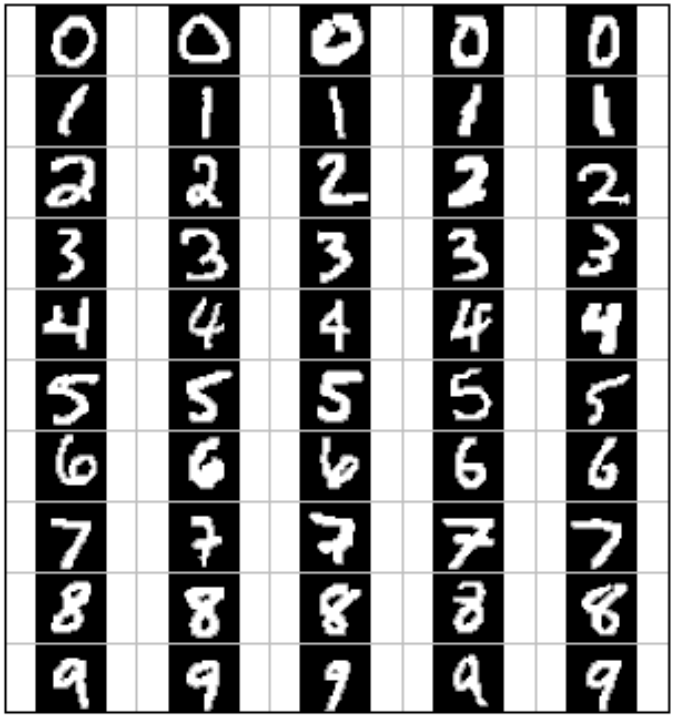
Main DS difficulties

- Figuring out the problem,
- Accessing the data,
- Not the ML part!

Source: Ch. Bourguignat

Outline

Number

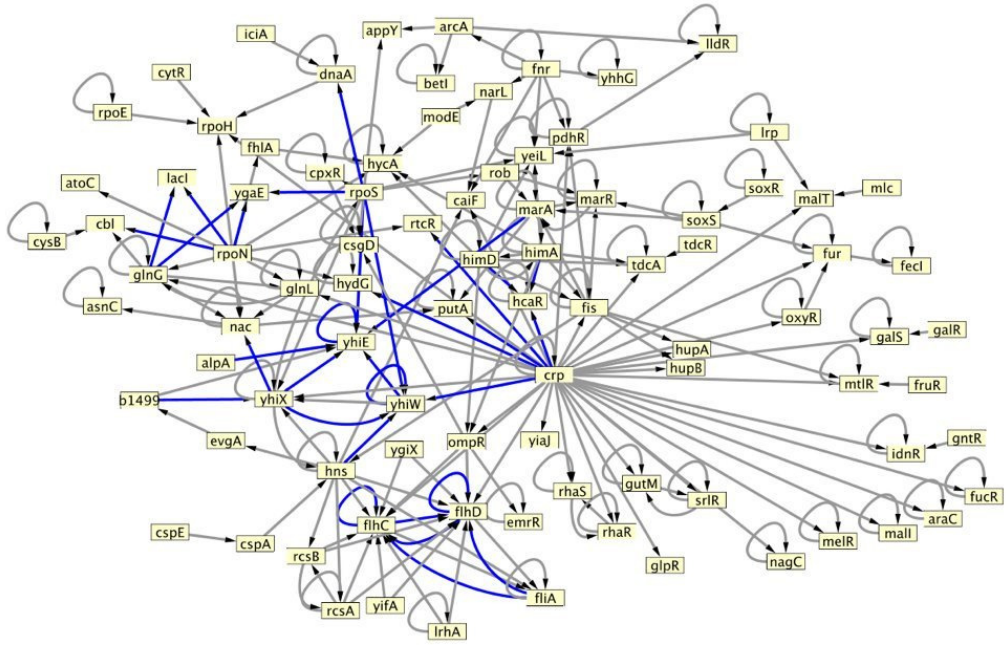


Reading a ZIP code on an envelop

- **Task:** give a number from an image.
- **Data:** $\underline{X}$ = image / Y = corresponding number.
- **Performance measure:** error rate.

Source: Y. LeCun

Bio

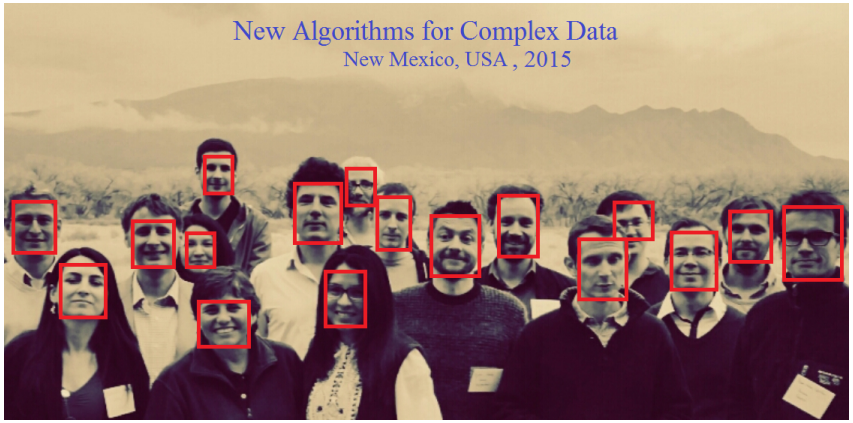


Predicting protein interaction

- **Task:** Predict (unknown) interactions between proteins.
- **Data:** $\underline{X}$ = pair of proteins / Y = existence or no of interaction.
- **Performance measure:** error rate.
- Numerous similar questions in bio(informatics): genomic,...

Source: Unknown

Detection

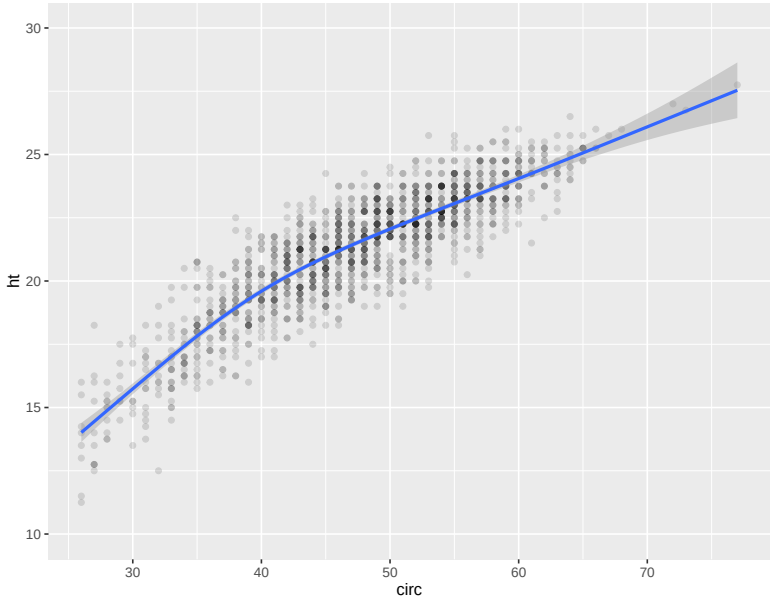


Face detection

- **Task:** Detect the position of faces in an image
- Different setting?
- Reformulation as a supervised learning problem.
- **Goal:** Detect the presence of faces at several positions and scales.
- **Data:** $\underline{X}$ = sub image / Y = presence or no of a face...
- **Performance measure:** error rate.
- Lots of detections in an image: post processing required...
- **Performance measure:** box precision.

Source: A. Fermin

Eucalyptus

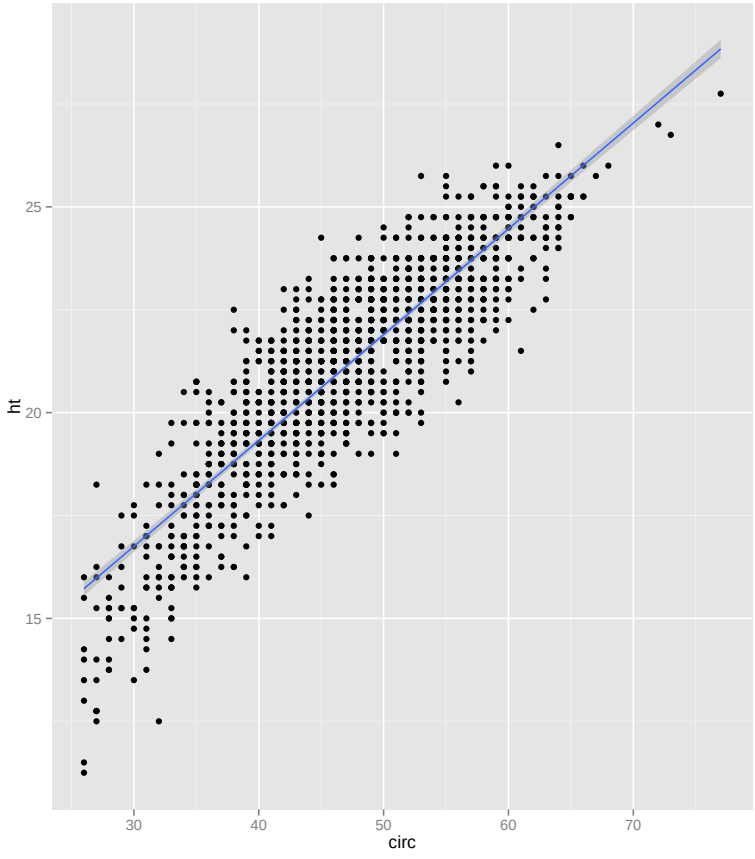


Height estimation

- Simple (and classical) dataset.
- **Task:** predict the height from circumference.
- **Data:** $\underline{X}$ = circumference /
- Y = height.
- **Performance measure:** means squared error.

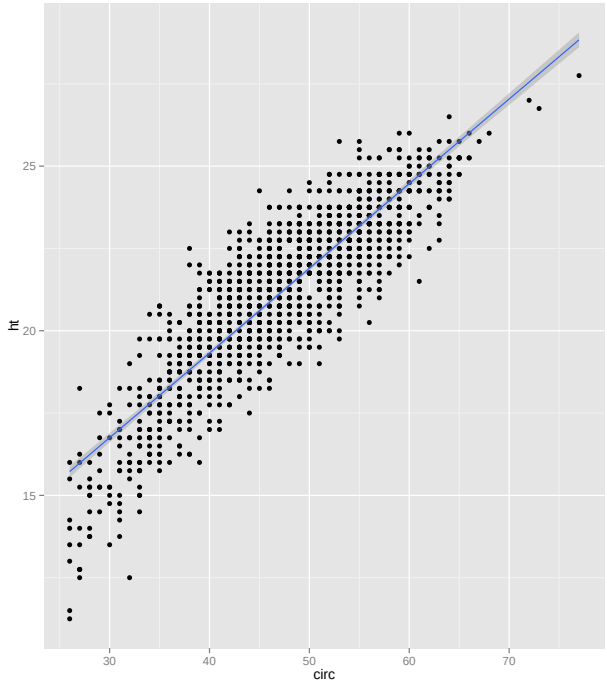
Outline

Eucalyptus



- Simple (and classical) dataset.
- Goal: predict the height from circumference
- $\underline{X} = \texttt{circ} =$ circumference.
- $Y = ht =$ height.

Eucalyptus



Linear Model

- Parametric model:

$$f_\beta(\texttt{circ}) = \beta^{(1)} + \beta^{(2)}\texttt{circ}$$

- How to choose $\beta = (\beta^{(1)}, \beta^{(2)})$?

Least Squares

Methodology

- Natural goodness criterion:

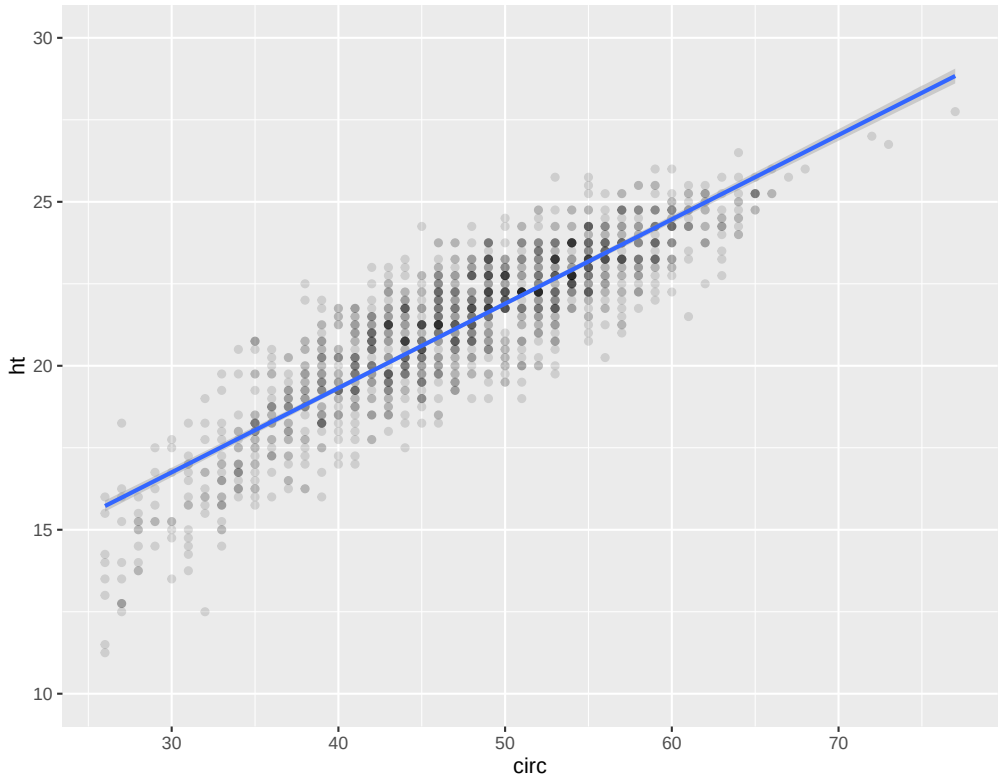
$$\sum_{i=1}^{n} |Y_i - f_\beta(\underline{X}_i)|^2 = \sum_{i=1}^{n} |\mathrm{ht}_i - f_\beta(\mathtt{circ}_i)|^2$$
$$= \sum_{i=1}^{n} |\mathrm{ht}_i - (\beta^{(1)} + \beta^{(2)}\mathtt{circ}_i)|^2$$

- Choice of β that minimizes this criterion!

$$\widehat{\beta} = \underset{\beta \in \mathbb{R}^2}{\mathrm{argmin}} \sum_{i=1}^{n} |h_i - (\beta^{(1)} + \beta^{(2)}\mathtt{circ}_i)|^2$$

- Easy minimization with an explicit solution!

Prediction



Prediction

- Linear prediction for the height:

$$\widehat{\texttt{ht}} = f_{\widehat{\beta}}(\texttt{circ}) = \widehat{\beta}^{(1)} + \widehat{\beta}^{(2)}\texttt{circ}$$

Heuristic

Linear Regression

- Statistical model: $(\texttt{circ}_i, \texttt{ht}_i)$ **i.i.d.** with the same law than a generic $(\texttt{circ}, \texttt{ht})$.
- **Performance criterion:** Look for f with a **small average error**

$$\mathbb{E}\left[|\texttt{ht} - f(\texttt{circ})|^2\right]$$

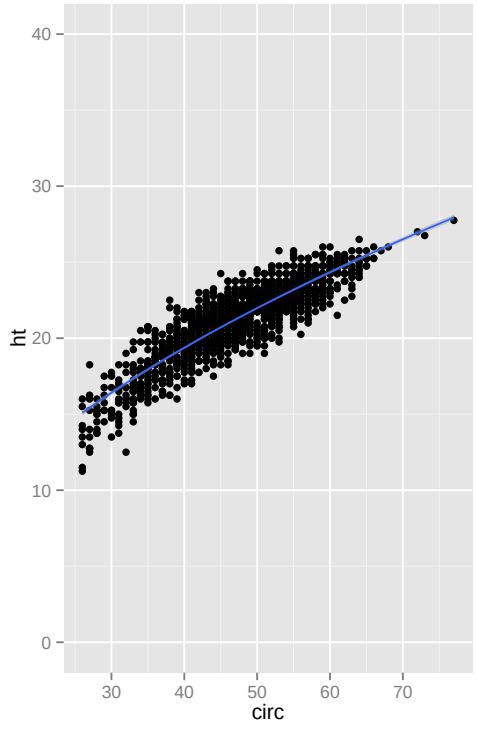
- **Empirical criterion:** Replace the unknown law by its **empirical** counterpart

$$\frac{1}{n}\sum_{i=1}^{n} |\texttt{ht}_i - f(\texttt{circ}_i)|^2$$

- **Predictor model:** As the minimum over all function is 0 (if all the $\texttt{circ}_i$ are different), **restrict** to the linear functions $f(\texttt{circ}) = \beta^{(1)} + \beta^{(2)}\texttt{circ}$ to avoid over-fitting.
- **Model fitting:** Explicit formula here.

- This model can be **too simple**!

Polynomial Regression

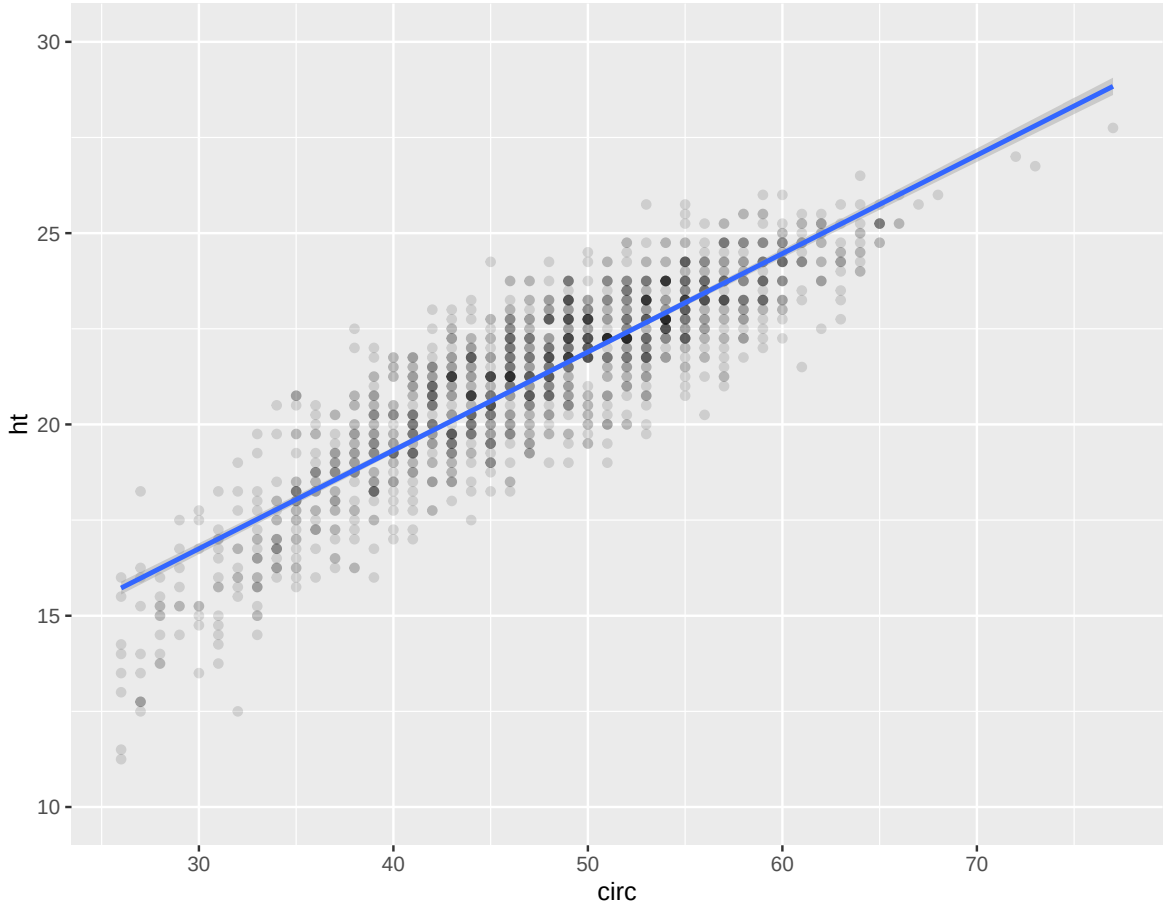


Polynomial Model

- Polynomial model: $f_\beta(\texttt{circ}) = \sum_{l=1}^{p} \beta^{(l)} \texttt{circ}^{l-1}$

- Linear in β!
- Easy least squares estimation for any degree!

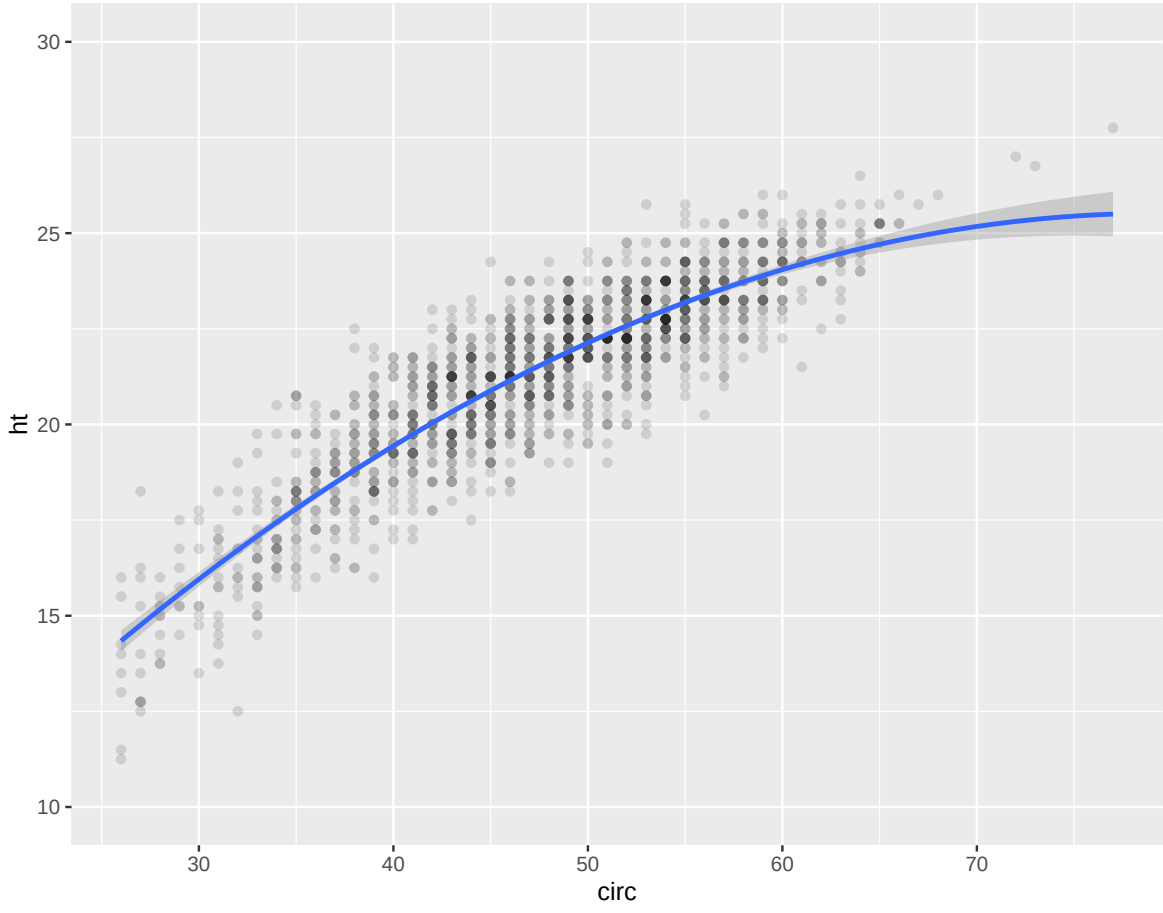
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

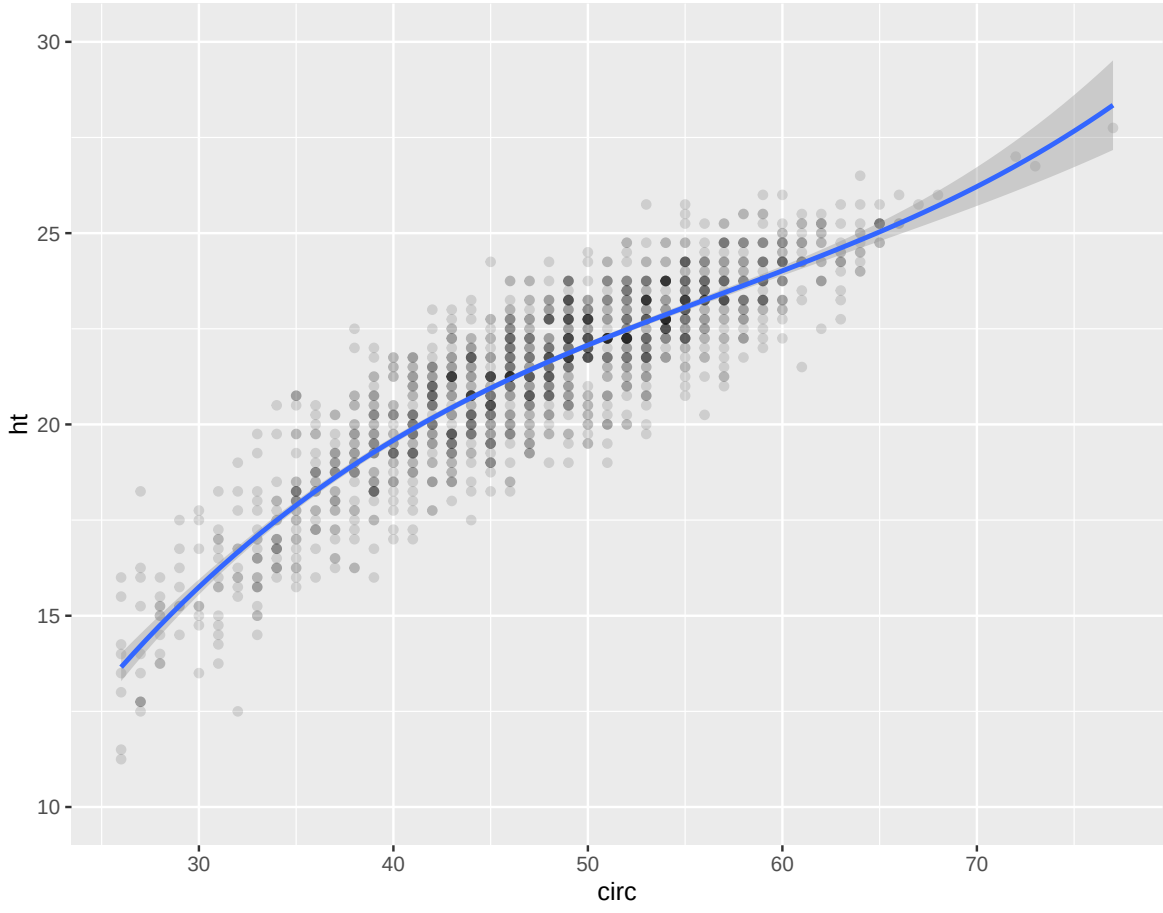
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

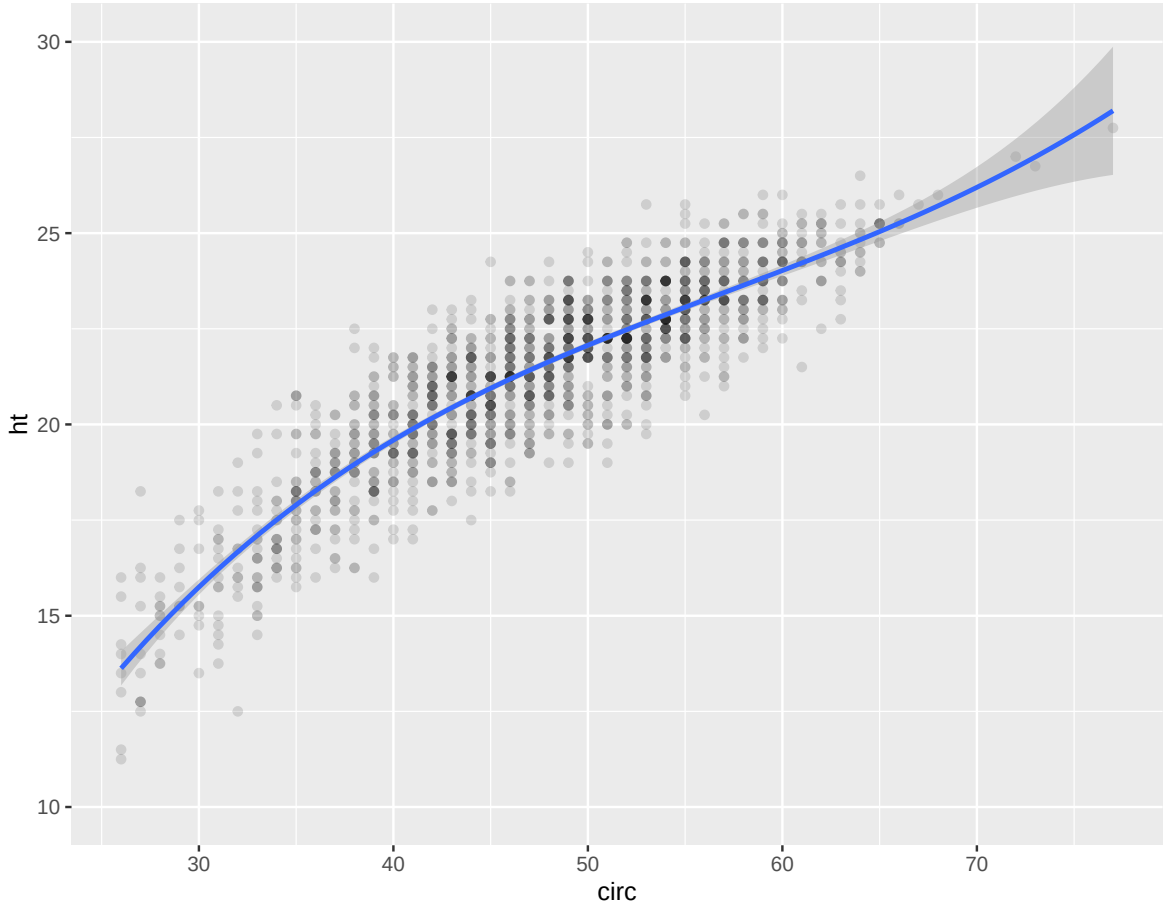
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

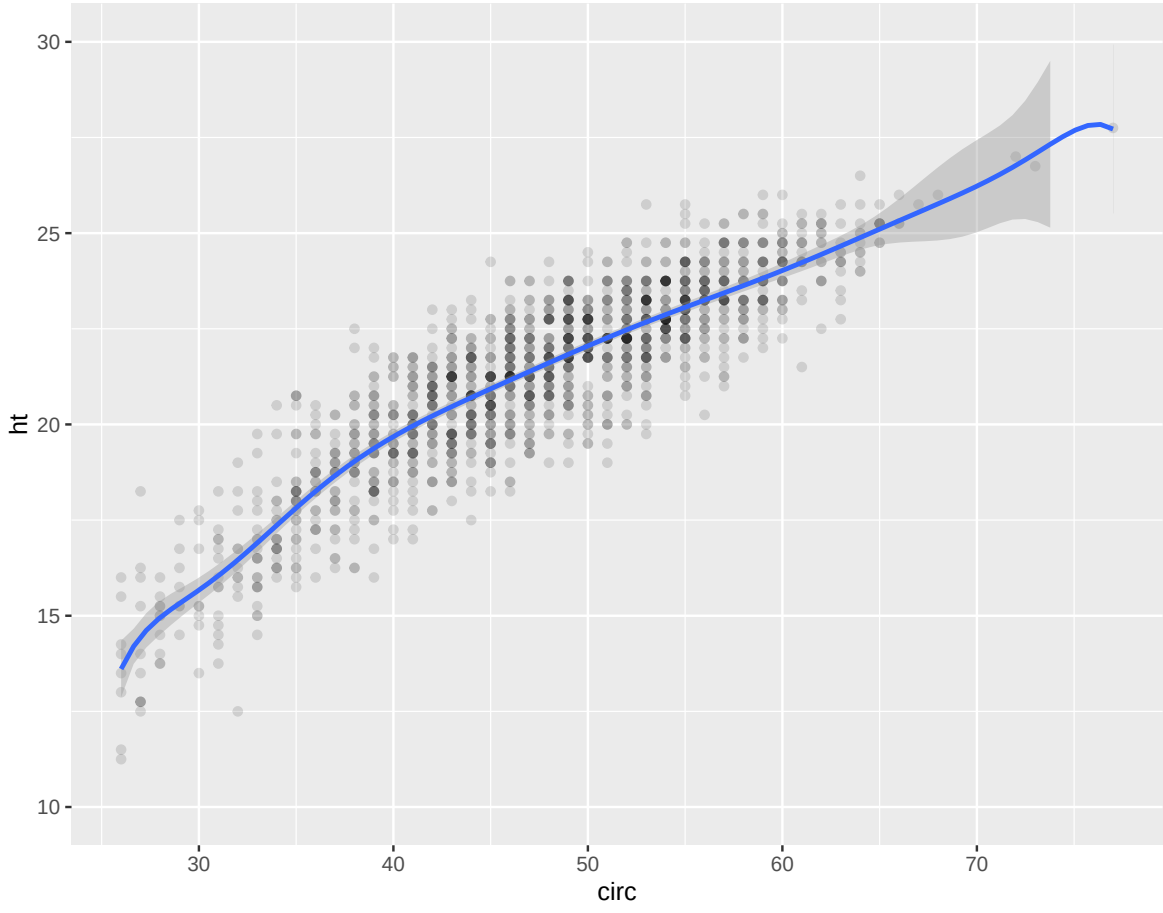
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

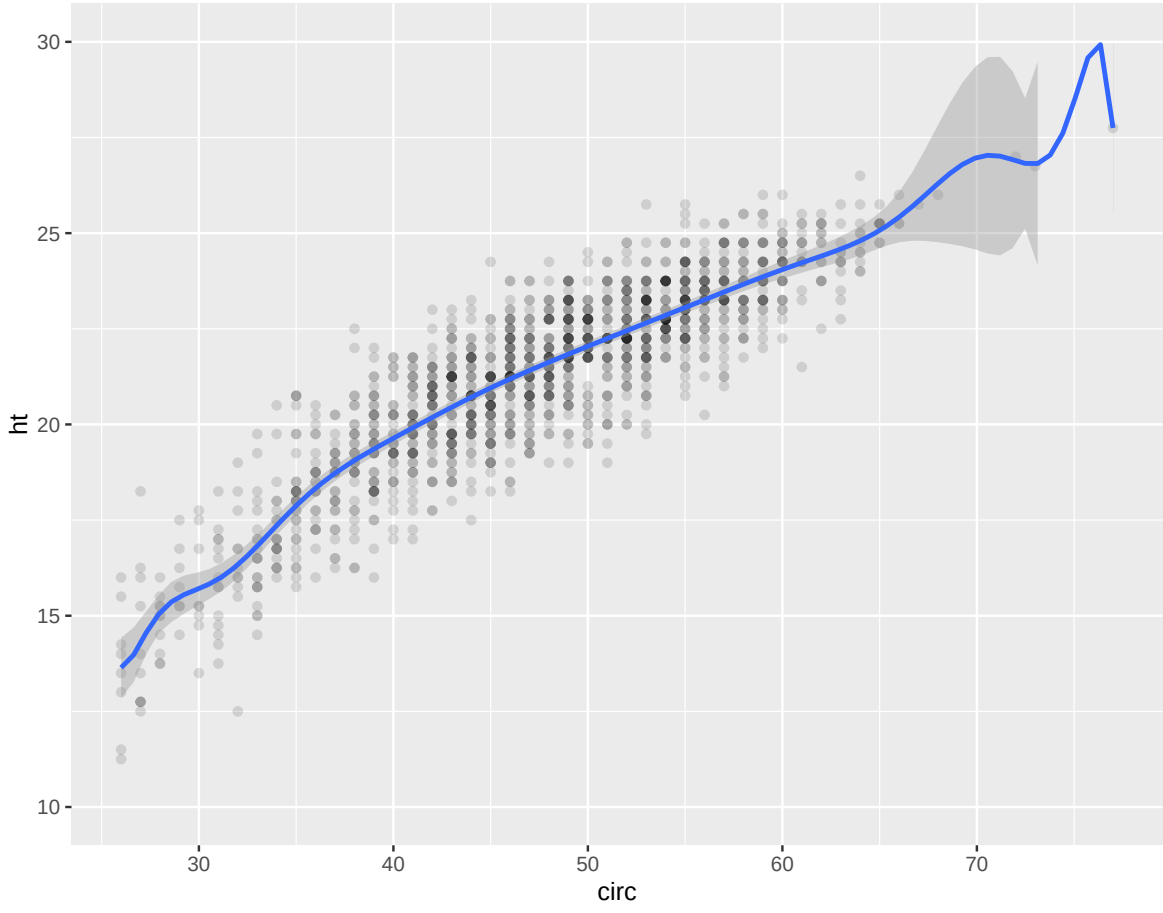
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

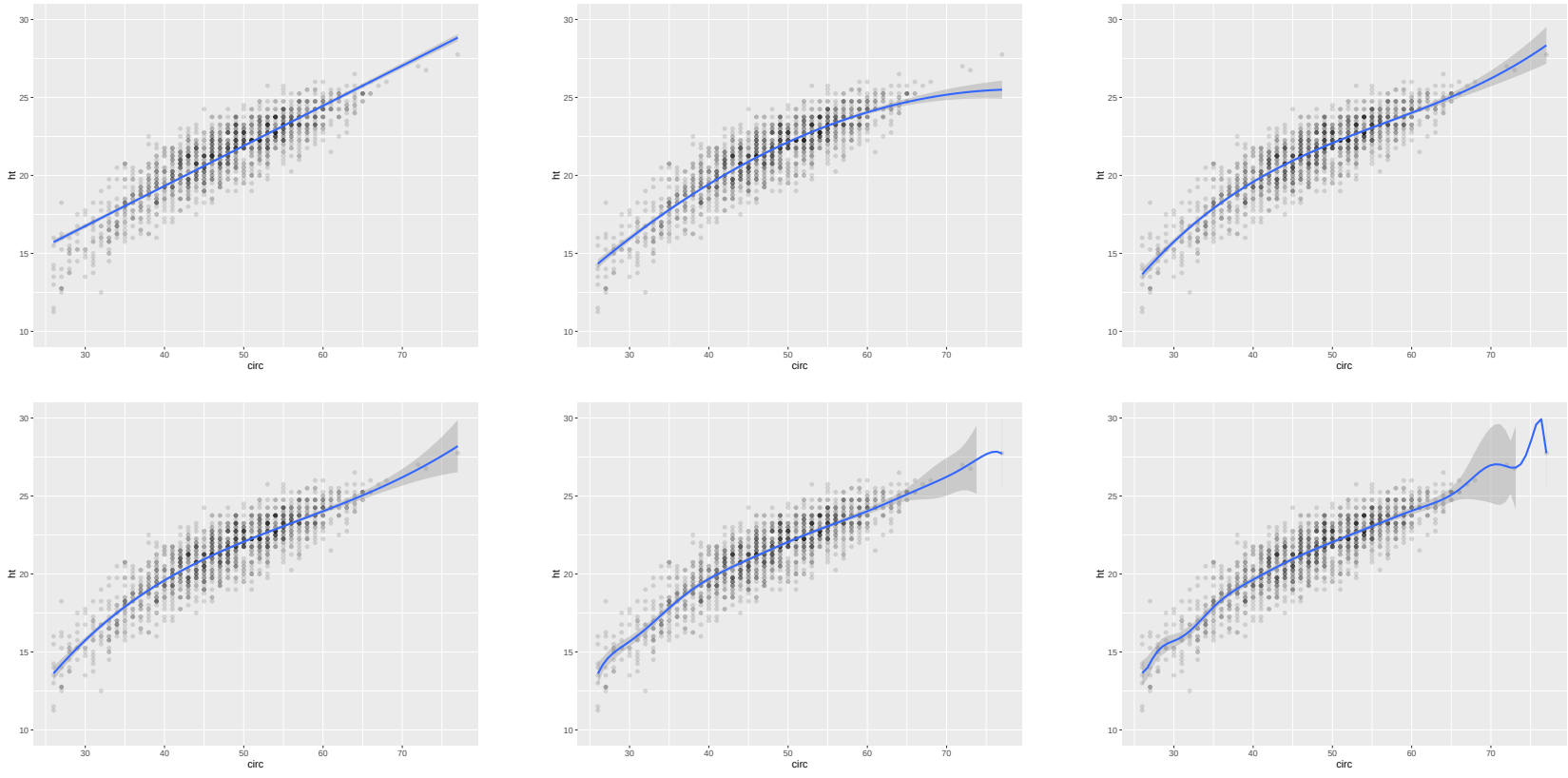
Which Degree?



Models

- Increasing degree = increasing complexity and better fit on the data

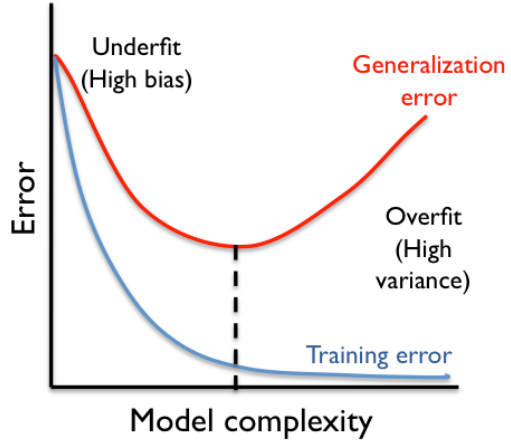
Which Degree?



Best Degree?

- How to choose among those solution?

Over-fitting Issue



Error behavior

- Empirical risk (error made on the training set) decays when the complexity of the model increases.
- Quite different behavior when the error is computed on new observations (true risk / generalization error).

- Overfit for complex models: parameters learned are too specific to the learning set!
- General situation! (Think of polynomial fit...)
- Need to use an other criterion than the training error!

Cross Validation and Penalization

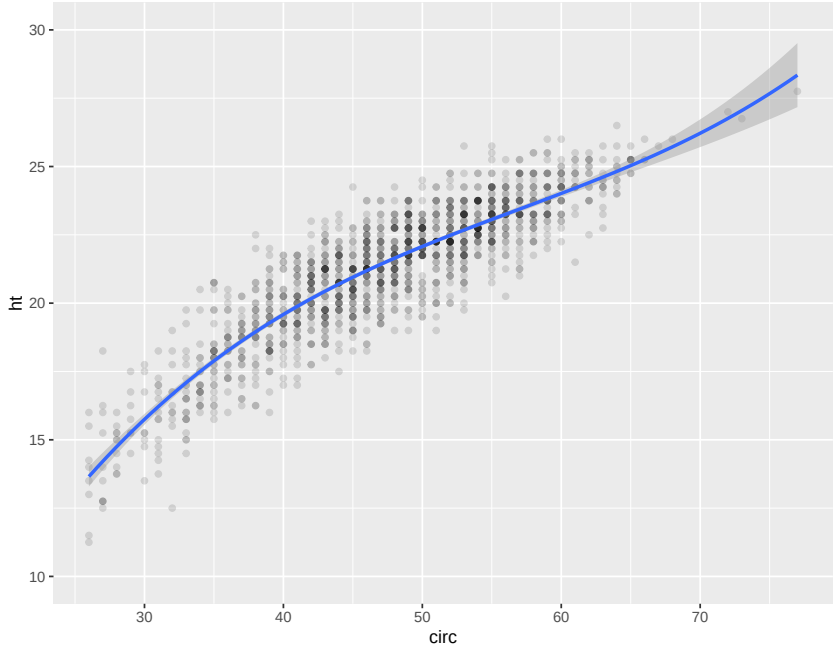
Two directions

- **How to estimate** the generalization error in a different way?
- Find a way to **correct** the empirical error?

Two Approaches

- **Cross validation:** Estimate the error on a different dataset:
  - Very efficient (and almost always used in practice!)
  - Need more data for the error computation.
- **Penalization approach:** Correct the optimism of the empirical error:
  - Require to find the correction (penalty).

Univariate Regression



Questions

- How to build a model?
- How to fit a model to the data?
- How to assess its quality?
- How to select a model among a collection?
- How to guaranty the quality of the selected model?

Outline

Supervised Learning

Supervised Learning Framework

- Input measurement $\underline{X} \in \mathcal{X}$
- Output measurement $Y \in \mathcal{Y}$.
- $(\underline{X}, Y) \sim \mathbb{P}$ with $\mathbb{P}$ unknown.
- **Training data** : $\mathcal{D}_n = \{(\underline{X}_1, Y_1), \ldots, (\underline{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbb{P}$)

- Often
  - $\underline{X} \in \mathbb{R}^d$ and $Y \in \{-1, 1\}$ (classification)
  - or $\underline{X} \in \mathbb{R}^d$ and $Y \in \mathbb{R}$ (regression).
- A **predictor** is a function in $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y} \text{ meas.}\}$

Goal

- Construct a **good** predictor $\widehat{f}$ from the training data.

- Need to specify the meaning of good.
- Classification and regression are almost the **same** problem!

Loss and Probabilistic Framework

Loss function for a generic predictor

- **Loss function**: $\ell(Y, f(\underline{X}))$ measures the goodness of the prediction of Y by $f(\underline{X})$
- Examples:
  - Prediction loss: $\ell(Y, f(\underline{X})) = \mathbf{1}_{Y \neq f(\underline{X})}$
  - Quadratic loss: $\ell(Y, f(\underline{X})) = |Y - f(\underline{X})|^2$

Risk function

- Risk measured as the average loss for a new couple:

$$\mathcal{R}(f) = \mathbb{E}_{(X,Y)\sim\mathbb{P}}\left[\ell(Y, f(\underline{X}))\right]$$

- Examples:
  - Prediction loss: $\mathbb{E}\left[\ell(Y, f(\underline{X}))\right] = \mathbb{P}\left(Y \neq f(\underline{X})\right)$
  - Quadratic loss: $\mathbb{E}\left[\ell(Y, f(\underline{X}))\right] = \mathbb{E}\left[|Y - f(\underline{X})|^2\right]$

- **Beware:** As $\widehat{f}$ depends on $\mathcal{D}_n$, $\mathcal{R}(\widehat{f})$ is a random variable!

Best Solution

- The best solution f^* (which is independent of $\mathcal{D}_n$) is

$$f^* = \arg\min_{f \in \mathcal{F}} R(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(Y, f(\underline{X}))\right] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\left[\mathbb{E}_{Y|\underline{X}}\left[\ell(Y, f(\underline{X}))\right]\right]$$

Bayes Predictor (explicit solution)

- In binary classification with $0-1$ loss:

$$f^*(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ & \Leftrightarrow \mathbb{P}(Y=+1|\underline{X}) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

- In regression with the quadratic loss

$$f^*(\underline{X}) = \mathbb{E}[Y|\underline{X}]$$

Issue: Solution requires to **know** $\mathbb{E}[Y|\underline{X}]$ for all values of $\underline{X}$!

Goal

Machine Learning

- Learn a rule to construct a **predictor** $\widehat{f} \in \mathcal{F}$ from the training data $\mathcal{D}_n$ s.t. **the risk** $\mathcal{R}(\widehat{f})$ is **small on average** or with high probability with respect to $\mathcal{D}_n$.

- In practice, the rule should be an algorithm!

Canonical example: Empirical Risk Minimizer

- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the empirical loss

$$\widehat{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(\underline{X}_i))$$

- Examples:
  - Linear regression
  - Linear discrimination with

$$\mathcal{S} = \{\underline{x} \mapsto \operatorname{sign}\{\underline{x}^\top \beta + \beta^{(0)}\} \,/\, \beta \in \mathbb{R}^d, \beta^{(0)} \in \mathbb{R}\}$$

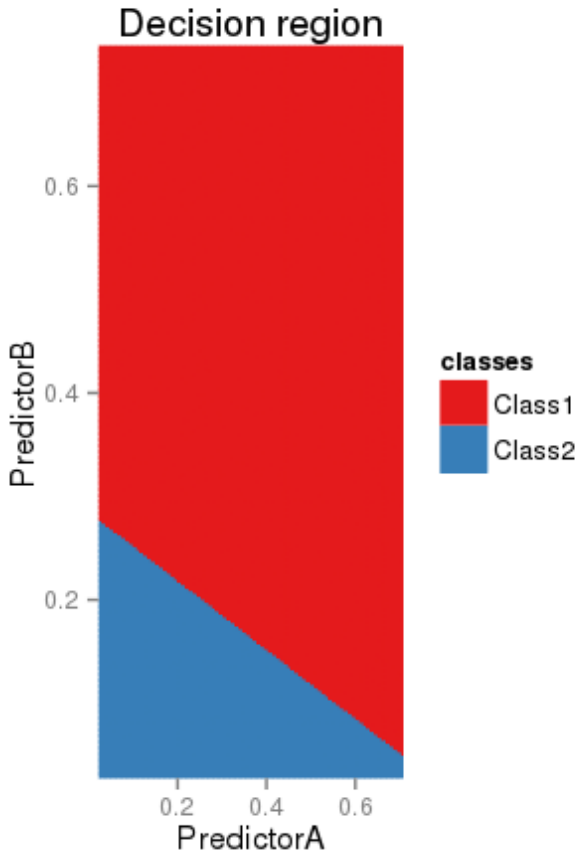
Example: TwoClass Dataset

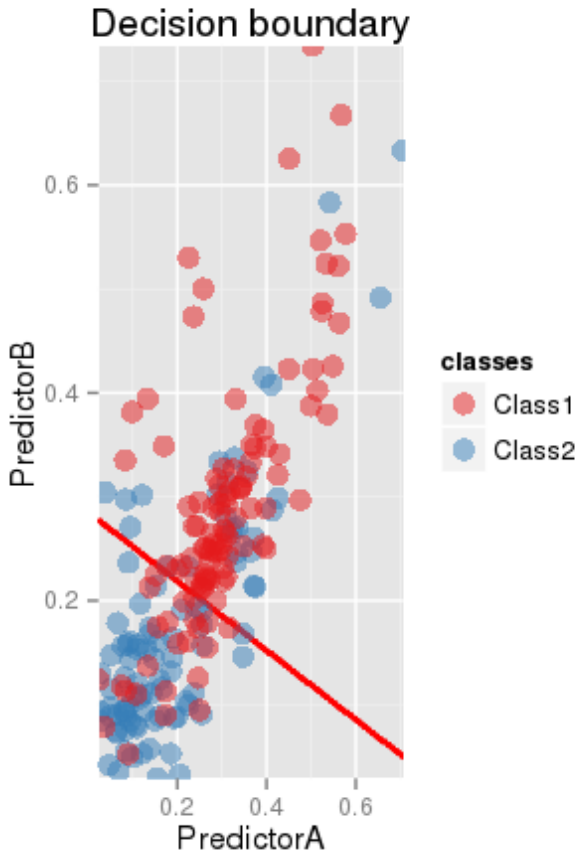
Synthetic Dataset

- Two features/covariates.
- Two classes.

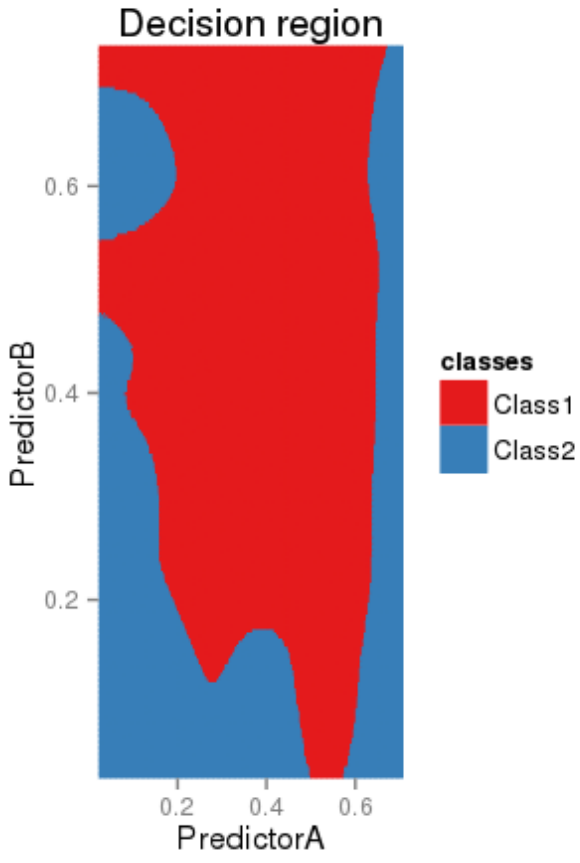
- Dataset from *Applied Predictive Modeling*, M. Kuhn and K. Johnson, Springer
- Numerical experiments with `R` and the `caret` package.

Example: Linear Discrimination

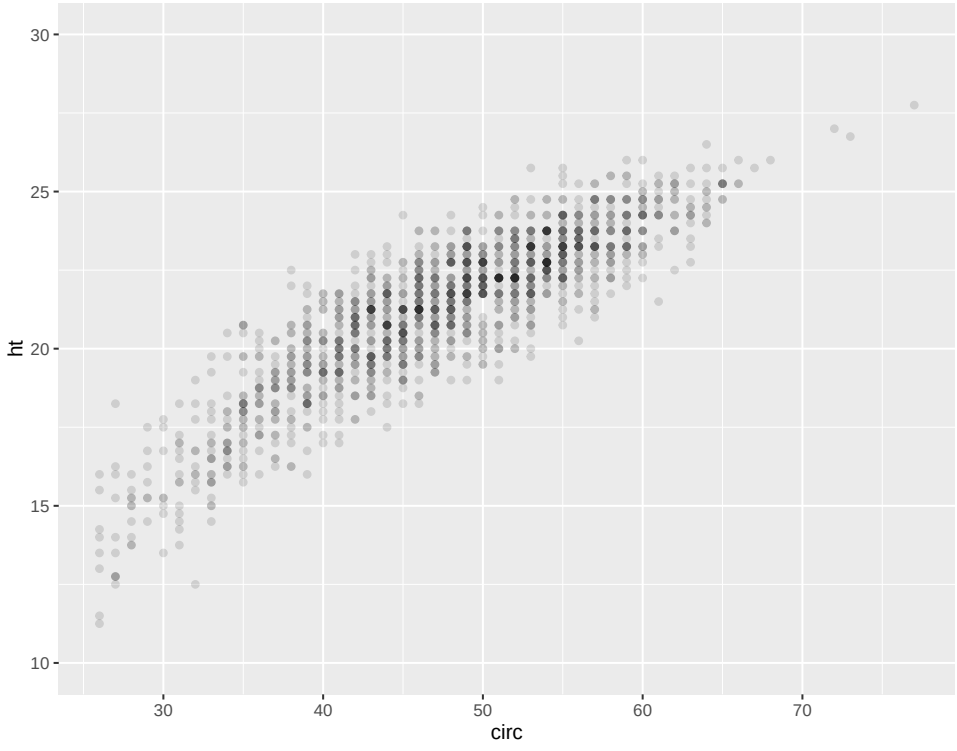




Example: More Complex Model



Eucalyptus

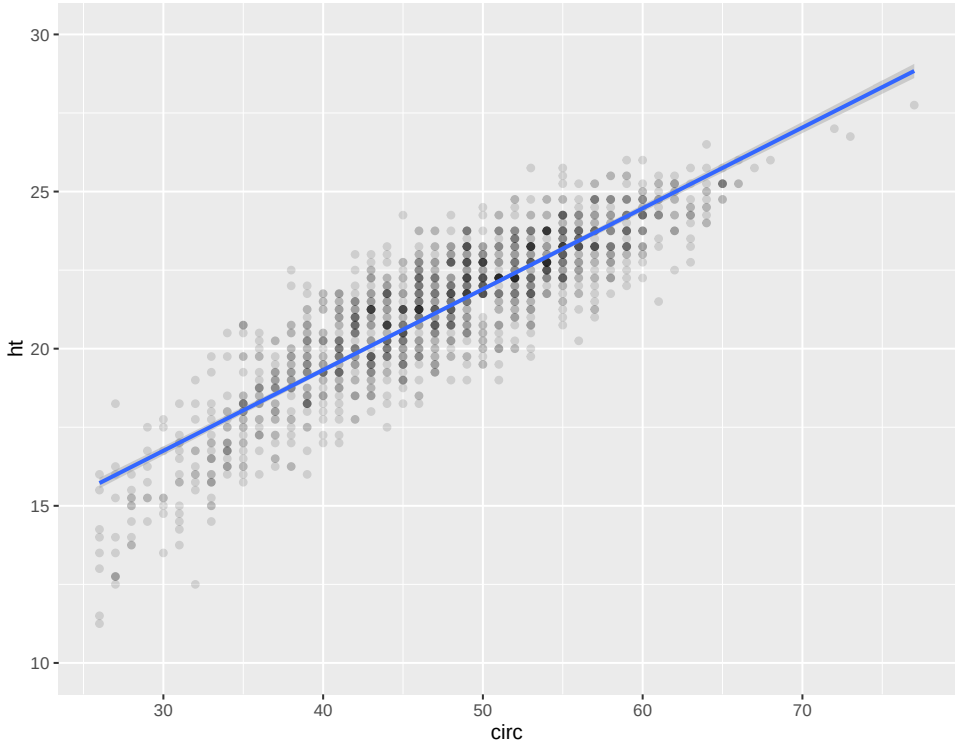


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

- Can we predict the height from the circumference?

Eucalyptus

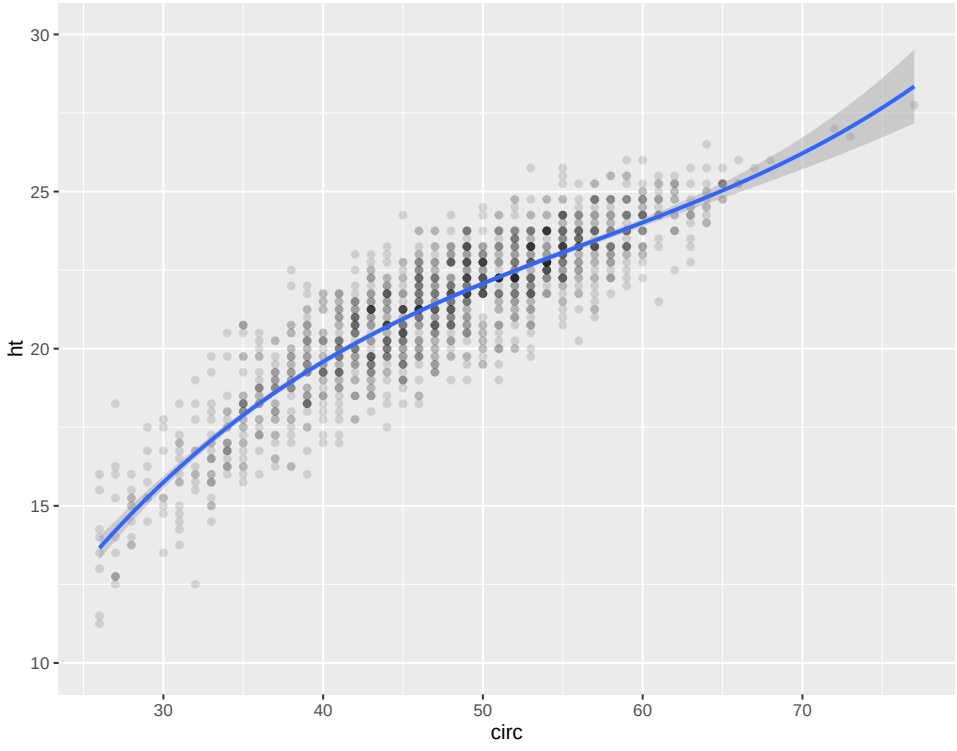


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

- Can we predict the height from the circumference?
  - by a **line**?

Eucalyptus

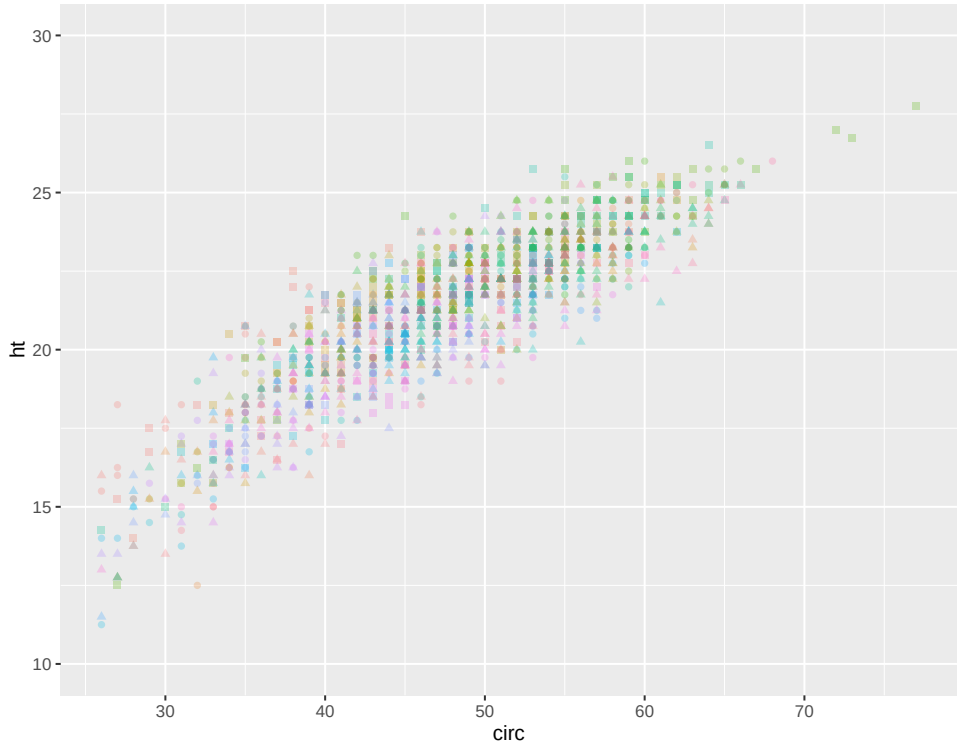


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

- Can we predict the height from the circumference?
  - by a **line**? by a more complex formula?

Eucalyptus

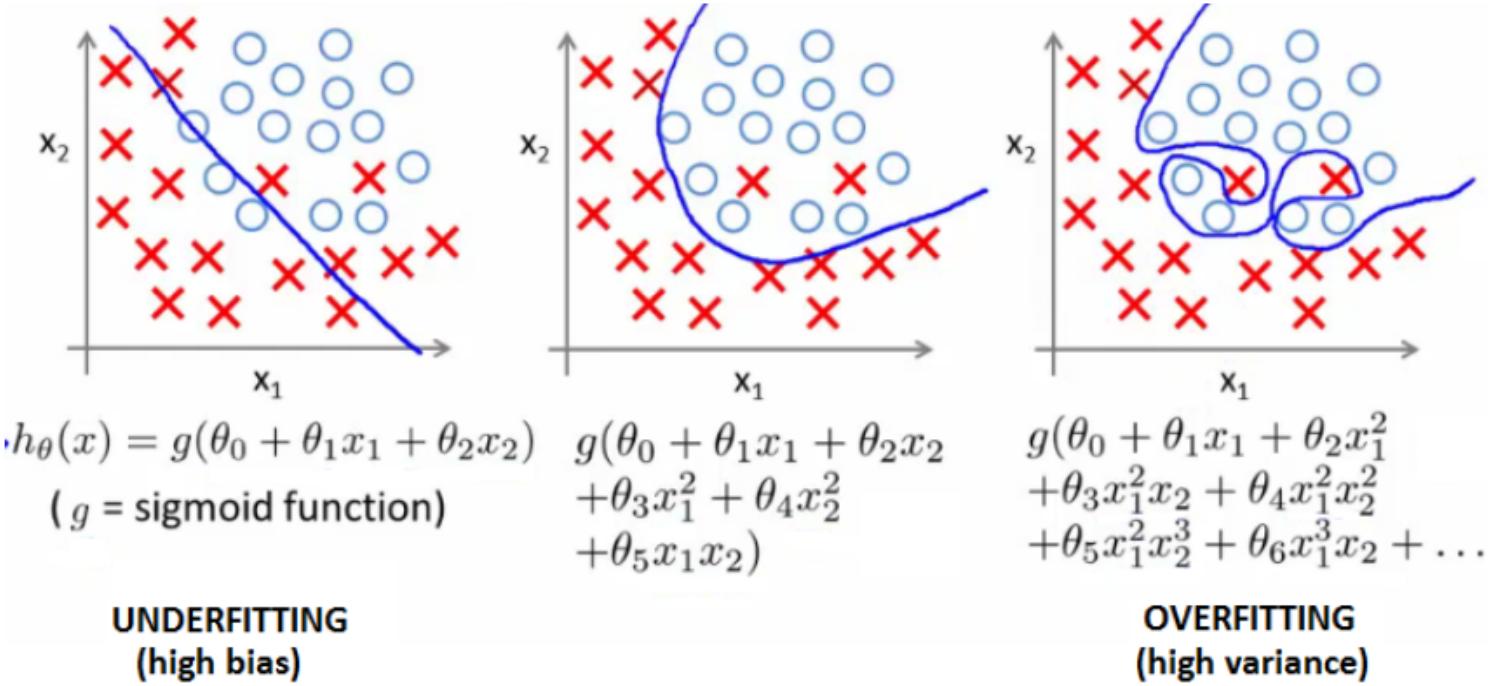


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference, block, clone / Y: height

- Can we predict the height from the circumference?
  - by a **line**? by a more complex formula?
  - by also taking account of the block and the clone type?

Under-fitting / Over-fitting Issue

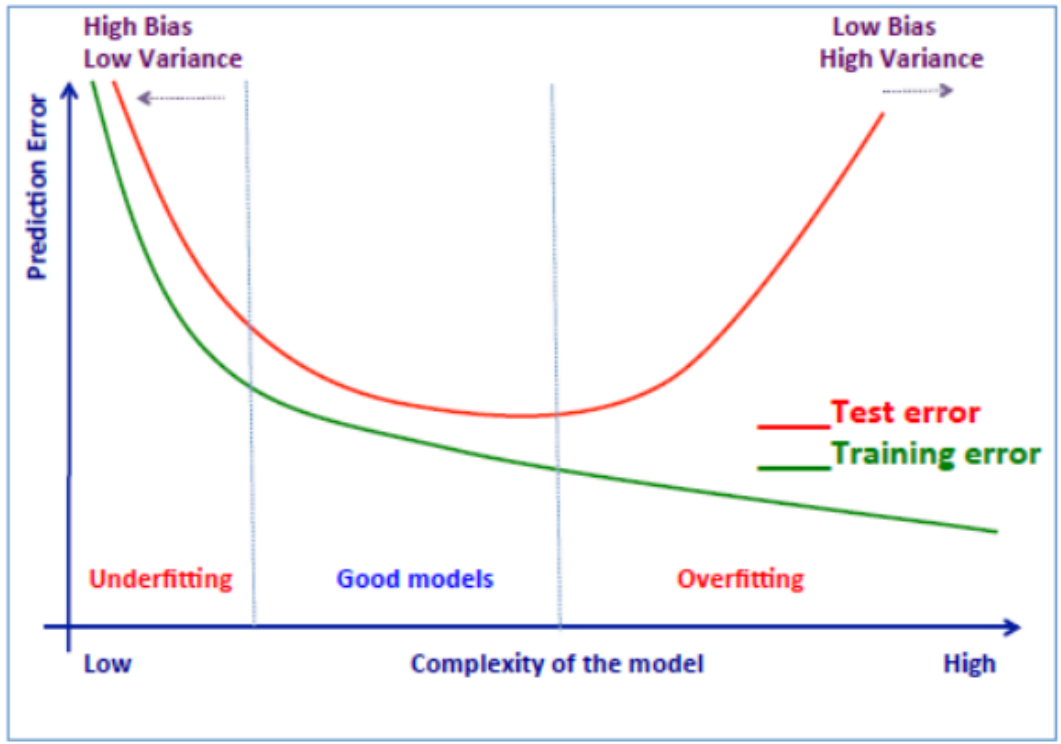


Model Complexity Dilemna

- What is best a simple or a complex model?
- Too simple to be good? Too complex to be learned?

Source: A. Ng

Under-fitting / Over-fitting Issue



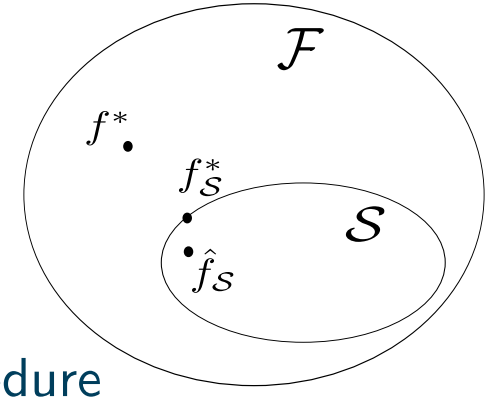
Under-fitting / Over-fitting

- **Under-fitting:** simple model are too simple.
- **Over-fitting:** complex model are too specific to the training set.

Source: Unknown

Bias-Variance Dilemma

- General setting:
  - $\mathcal{F} = \{\text{measurable functions } \mathcal{X} \to \mathcal{Y}\}$
  - Best solution: $f^* = \text{argmin}_{f \in \mathcal{F}} \mathcal{R}(f)$
  - Class $\mathcal{S} \subset \mathcal{F}$ of functions
  - Ideal target in $\mathcal{S}$: $f_{\mathcal{S}}^* = \text{argmin}_{f \in \mathcal{S}} \mathcal{R}(f)$
  - Estimate in $\mathcal{S}$: $\widehat{f}_{\mathcal{S}}$ obtained with some procedure



Approximation error and estimation error (Bias/Variance)

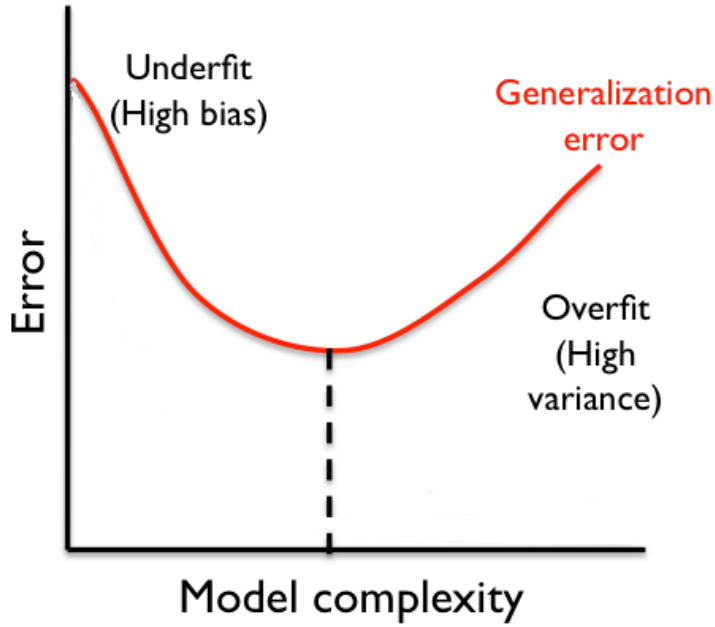
$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^*) = \underbrace{\mathcal{R}(f_{\mathcal{S}}^*) - \mathcal{R}(f^*)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f_{\mathcal{S}}^*)}_{\text{Estimation error}}$$

- Approx. error can be large if the model $\mathcal{S}$ is not suitable.
- Estimation error can be large if the model is complex.

Agnostic approach

- No assumption (so far) on the law of (X, Y).

Under-fitting / Over-fitting Issue



- Different behavior for different model complexity
- **Low complexity model** are easily learned but the approximation error (**bias**) may be large (**Under-fit**).
- **High complexity model** may contain a good ideal target but the estimation error (**variance**) can be large (**Over-fit**)

Bias-variance trade-off $\Longleftrightarrow$ avoid **overfitting** and **underfitting**

- **Rk:** Better to think in term of method (including feature engineering and specific algorithm) rather than only of model.

Source: Unknown

Theoretical Analysis

Statistical Learning Analysis

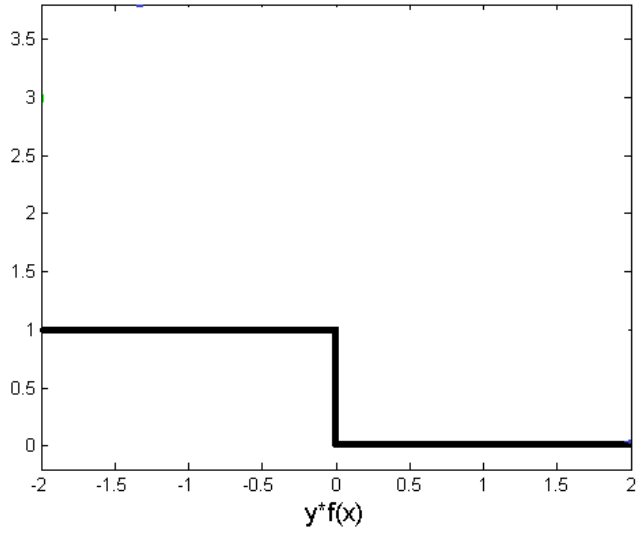
- Error decomposition:

$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(f^\star_{\mathcal{S}}) - \mathcal{R}(f^\star)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star_{\mathcal{S}})}_{\text{Estimation error}}$$

- Bound on the approximation term: approximation theory.
- Probabilistic bound on the estimation term: probability theory!
- **Goal: Agnostic bounds**, i.e. bounds that do not require assumptions on $\mathbb{P}$! (Statistical Learning?)

- Often need mild assumptions on $\mathbb{P}$... (Nonparametric Statistics?)

Binary Classification Loss Issue

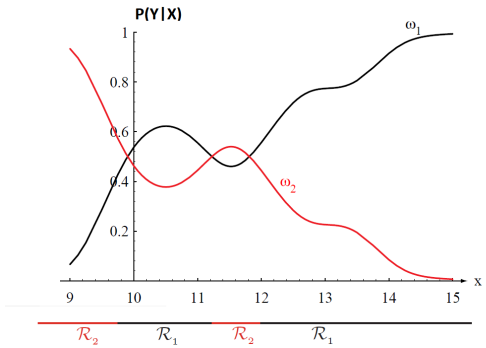


Empirical Risk Minimizer

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

- Classification loss: $\ell^{0/1}(y, f(\underline{x})) = \mathbf{1}_{y \neq f(\underline{x})}$
- Not convex and not smooth!

Probabilistic Point of View
Ideal Solution and Estimation



- The best solution f^* (which is independent of $\mathcal{D}_n$) is

$$f^* = \arg\min_{f \in \mathcal{F}} R(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(Y, f(\underline{X}))\right] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\left[\mathbb{E}_{Y|\underline{X}}\left[\ell(Y, f(\underline{x}))\right]\right]$$

Bayes Predictor (explicit solution)

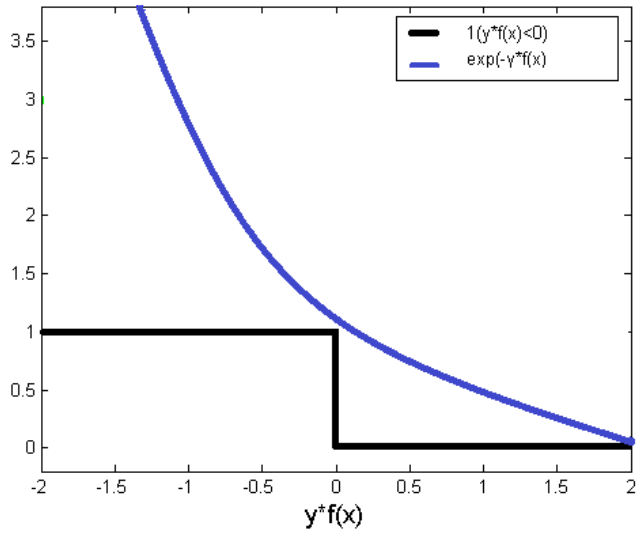
In binary classification with $0-1$ loss:

$$f^*(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- **Issue:** Solution requires to **know** $\mathbb{E}[Y|\underline{X}]$ for all values of $\underline{X}$!
- **Solution:** Replace it by an estimate.

Source: A. Fermin

Optimization Point of View
Loss Convexification



Minimizer of the risk

$$\widehat{f} = \operatorname*{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(X_i))$$

- **Issue:** Classification loss is not convex or smooth.
- **Solution:** Replace it by a convex majorant.

Probabilistic and Optimization Framework

How to find a good function f with a *small* risk

$$R(f) = \mathbb{E}\left[\ell(Y, f(\underline{X}))\right] \quad ?$$

Canonical approach: $\widehat{f}_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\underline{X}_i))$

Problems

- How to choose $\mathcal{S}$?
- How to compute the minimization?

A Probabilistic Point of View

Solution: For $\underline{X}$, estimate $Y|\underline{X}$ plug this estimate in the Bayes classifier: **(Generalized) Linear Models, Kernel methods, k-nn, Naive Bayes, Tree, Bagging...**

An Optimization Point of View

Solution: If necessary replace the loss ℓ by an upper bound ℓ' and minimize the empirical loss: **SVR, SVM, Neural Network,Tree, Boosting...**

Outline

Best Solution

- The best solution f^* (which is independent of $\mathcal{D}_n$) is

$$f^* = \arg\min_{f \in \mathcal{F}} R(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(Y, f(\underline{X}))\right] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\left[\mathbb{E}_{Y|\underline{X}}\left[\ell(Y, f(\underline{X})\right]\right]$$

Bayes Predictor (explicit solution)

- In binary classification with $0-1$ loss:

$$f^*(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ & \Leftrightarrow \mathbb{P}(Y=+1|\underline{X}) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

- In regression with the quadratic loss

$$f^*(\underline{X}) = \mathbb{E}[Y|\underline{X}]$$

Issue: Explicit solution requires to **know** $Y|\underline{X}$ (or $\mathbb{E}[Y|\underline{X}]$) for all values of $\underline{X}$!

Plugin Predictor

- **Idea:** Estimate $Y|\underline{X}$ by $\widehat{Y|\underline{X}}$ and plug it the Bayes classifier.

Plugin Bayes Predictor

- In binary classification with $0-1$ loss:

$$\widehat{f}(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{\mathbb{P}(Y=+1|\underline{X})} \geq \widehat{\mathbb{P}(Y=-1|\underline{X})} \\ & \Leftrightarrow \widehat{\mathbb{P}(Y=+1|\underline{X})} \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

- In regression with the quadratic loss

$$\widehat{f}(\underline{X}) = \mathbb{E}\left[\widehat{Y|\underline{X}}\right]$$

- **Rk:** Direct estimation of $\mathbb{E}[Y|\underline{X}]$ by $\widehat{\mathbb{E}[Y|\underline{X}]}$ also possible...

Plugin Predictor

- How to estimate $Y|\underline{X}$?

Three main heuristics

- **Fully Generative modeling:** Estimate the law of $(\underline{X}, Y)$ and use the **Bayes formula** to deduce an estimate of $Y|\underline{X}$: *LDA/QDA, Naive Bayes...*
- **Parametric Conditional modeling:** Estimate the law of $Y|\underline{X}$ by a **parametric** law $\mathcal{L}_\theta(\underline{X})$: *(generalized) linear regression...*
- **Non Parametric Conditional modeling:** Estimate the law of $Y|\underline{X}$ by a **non parametric** estimate: *kernel methods, loess, nearest neighbors...*

- **Rk:** Direct estimation of $\mathbb{E}[Y|\underline{X}]$ by $\widehat{\mathbb{E}[Y|\underline{X}]}$ also possible...

Plugin Classifier

- **Input**: a data set $\mathcal{D}_n$
  Learn $Y|\underline{X}$ or equivalently $\mathbb{P}(Y = k|\underline{X})$ (using the data set) and plug this estimate in the Bayes classifier

- **Output**: a classifier $\widehat{f} : \mathbb{R}^d \to \{-1, 1\}$

$$\widehat{f}(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(\widehat{Y = 1}|\underline{X}) \geq \mathbb{P}(\widehat{Y = -1}|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- Can we guaranty that the classifier is good if $Y|\underline{X}$ is well estimated?

Classification Risk Analysis

Theorem

- If $\widehat{f} = \text{sign}(2\widehat{p}_{+1} - 1)$ then

$$\mathbb{E}\left[\ell^{0,1}(Y, \widehat{f}(\underline{X}))\right] - \mathbb{E}\left[\ell^{0,1}(Y, f^\star(\underline{X}))\right]$$
$$\leq \mathbb{E}\left[\|\widehat{Y|\underline{X}} - Y|\underline{X}\|_1\right]$$
$$\leq \left(\mathbb{E}\left[2\text{KL}(Y|\underline{X}, \widehat{Y|\underline{X}}\right]\right)^{1/2}$$

- If one estimates $\mathbb{P}(Y = 1|\underline{X})$ well then one estimates $f^\star$ well!
- Link between a *conditional density estimation* task and a *classification* one!
- **Rk:** In general, the conditional density estimation task is more complicated as one should be good for all values of $\mathbb{P}(Y = 1|\underline{X})$ while the classification task focus on values around $1/2$ for the $0/1$ loss!
- In **regression**, (often) direct control of the quadratic loss...

Outline

Fully Generative Modeling

- **Idea:** If one knows the law of $(\underline{X}, Y)$ everything is easy!

Bayes formula

- With a slight abuse of notation,

$$\begin{aligned}\mathbb{P}(Y|\underline{X}) &= \frac{\mathbb{P}((\underline{X}, Y))}{\mathbb{P}(\underline{X})} \\ &= \frac{\mathbb{P}(\underline{X}|Y)\,\mathbb{P}(Y)}{\mathbb{P}(\underline{X})}\end{aligned}$$

- **Generative Modeling:**
  - Propose a model for $(\underline{X}, Y)$ (or equivalently $\underline{X}|Y$ and Y),
  - Estimate it as a density estimation problem,
  - Plug the estimate in the Bayes formula
  - Plug the conditional estimate in the Bayes *classifier*.
- **Rk:** Require to estimate $(\underline{X}, Y)$ rather than only $Y|\underline{X}$!
- Great flexibility in the model design but may lead to complex computation.

Fully Generative Modeling

- Simpler setting in classification!

Bayes formula

$$\mathbb{P}(Y = k|\underline{X}) = \frac{\mathbb{P}(\underline{X}|Y = k)\,\mathbb{P}(Y = k)}{\mathbb{P}(\underline{X})}$$

- Binary Bayes classifier (the best solution)

$$f^*(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y = 1|\underline{X}) \geq \mathbb{P}(Y = -1|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

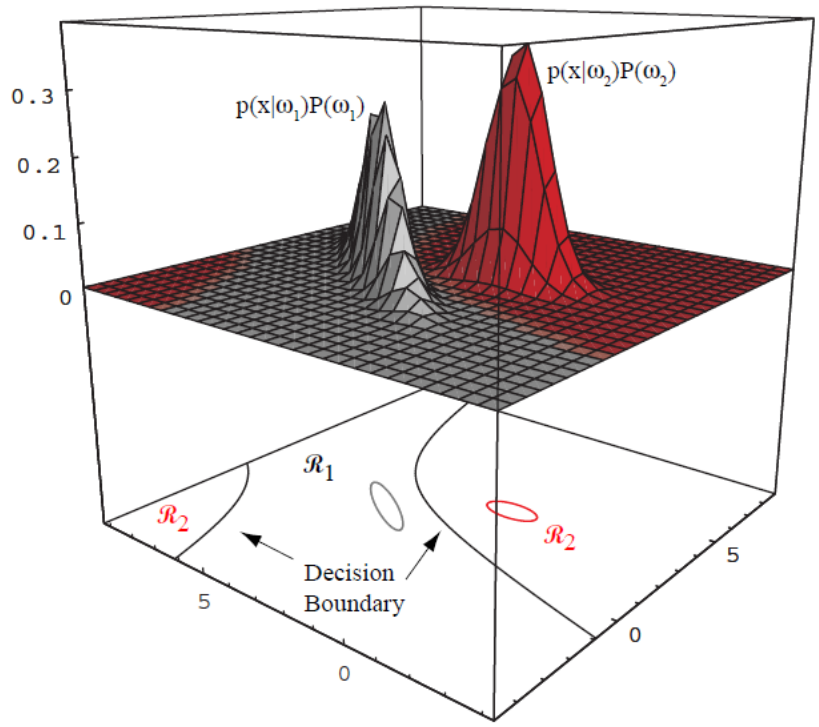
- **Heuristic**: Estimate those quantities and plug the estimations.
- By using different models/estimators for $\mathbb{P}(\underline{X}|Y)$, we get different classifiers.
- **Rk:** No need to renormalize by $\mathbb{P}(\underline{X})$ to take the decision!

Discriminant Analysis (Gaussian model)

- The densities are modeled as multivariate normal, i.e.,
$$\mathbb{P}(\underline{X}|Y=k) \sim \mathcal{N}_{\mu_k,\Sigma_k}$$
- Discriminant functions:
$$\boldsymbol{g_k(\underline{X}) = \ln(\mathbb{P}(\underline{X}|Y=k)) + \ln(\mathbb{P}(Y=k))}$$
$$g_k(\underline{X}) = -\frac{1}{2}(\underline{X}-\mu_k)^\top \Sigma_k^{-1}(\underline{X}-\mu_k) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\mathbb{P}(Y=k))$$
- QDA (different Σ_k in each class) and LDA ($\Sigma_k = \Sigma$ for all k)

- **Beware: this model can be false but the methodology remains valid!**

Discriminant Analysis

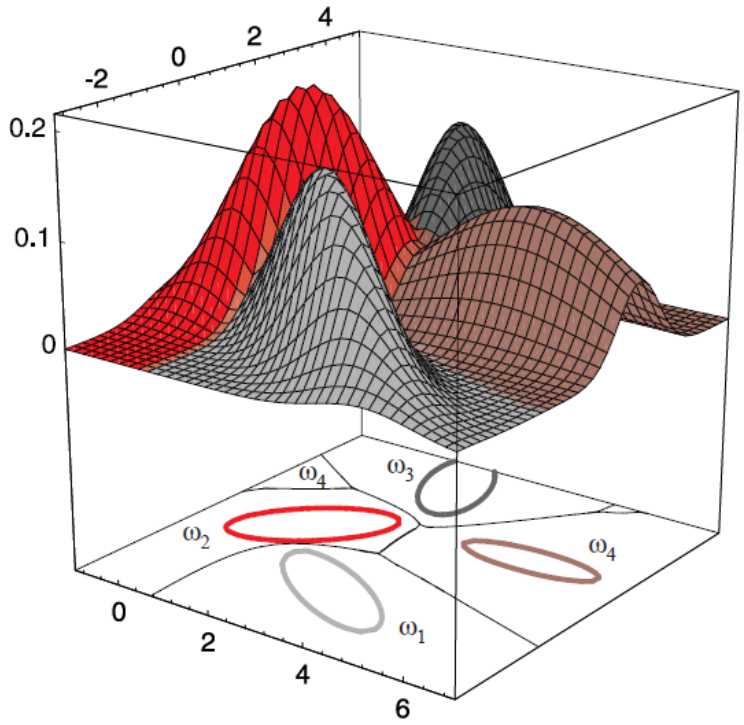


Quadratic Discriminant Analysis

- The probability densities are Gaussian
- The effect of any decision rule is to divide the feature space into some decision regions $\mathcal{R}_1, \mathcal{R}_2$
- The regions are separated by decision boundaries

Source: A. Fermin

Discriminant Analysis



Quadratic Discriminant Analysis

- The probability densities are Gaussian
- The effect of any decision rule is to divide the feature space into some decision regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_c$
- The regions are separated by decision boundaries

Source: A. Fermin

Discriminant Analysis

Estimation

In practice, we will need to estimate μ_k, Σ_k and $\mathbb{P}_k := \mathbb{P}(Y = k)$

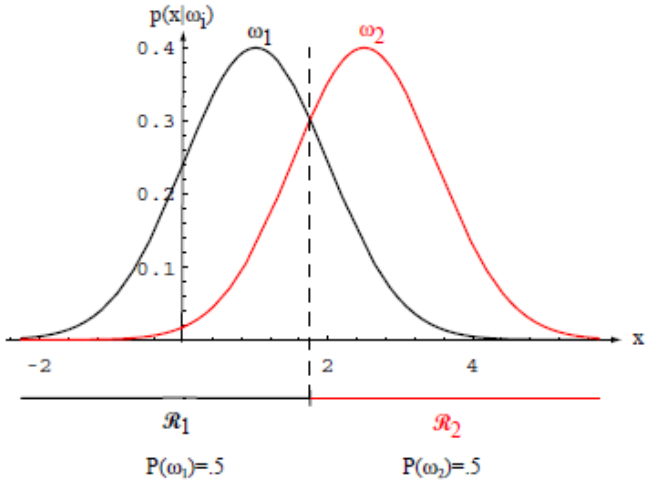
- The estimate proportion $\widehat{\mathbb{P}(Y = k)} = \frac{n_k}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{Y_i = k\}}$
- Maximum likelihood estimate of $\widehat{\mu_k}$ and $\widehat{\Sigma_k}$ (explicit formulas)

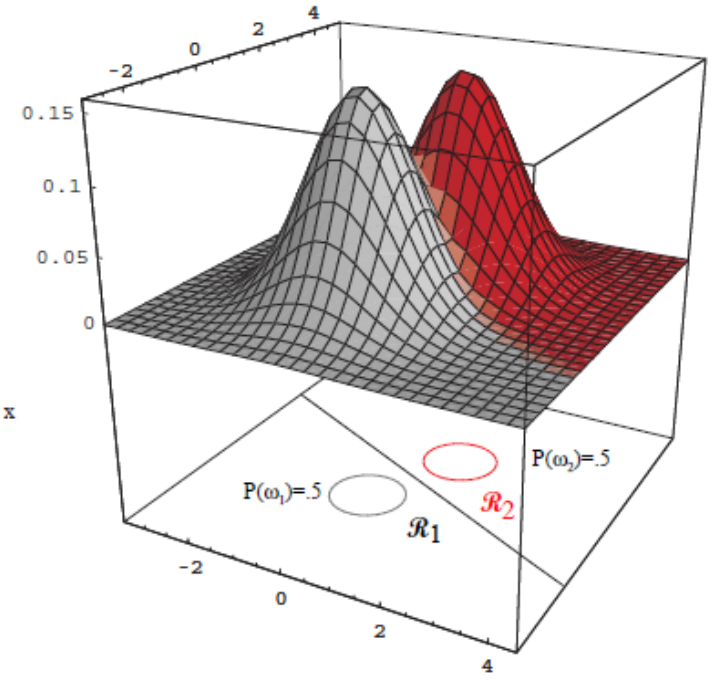
- DA classifier

$$\widehat{f}_G(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{g}_{+1}(\underline{X}) \geq \widehat{g}_{-1}(\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- Decision boundaries: quadratic = degree 2 polynomials.
- If one imposes $\Sigma_{-1} = \Sigma_1 = \Sigma$ then the decision boundaries is a linear hyperplane.

Discriminant Analysis



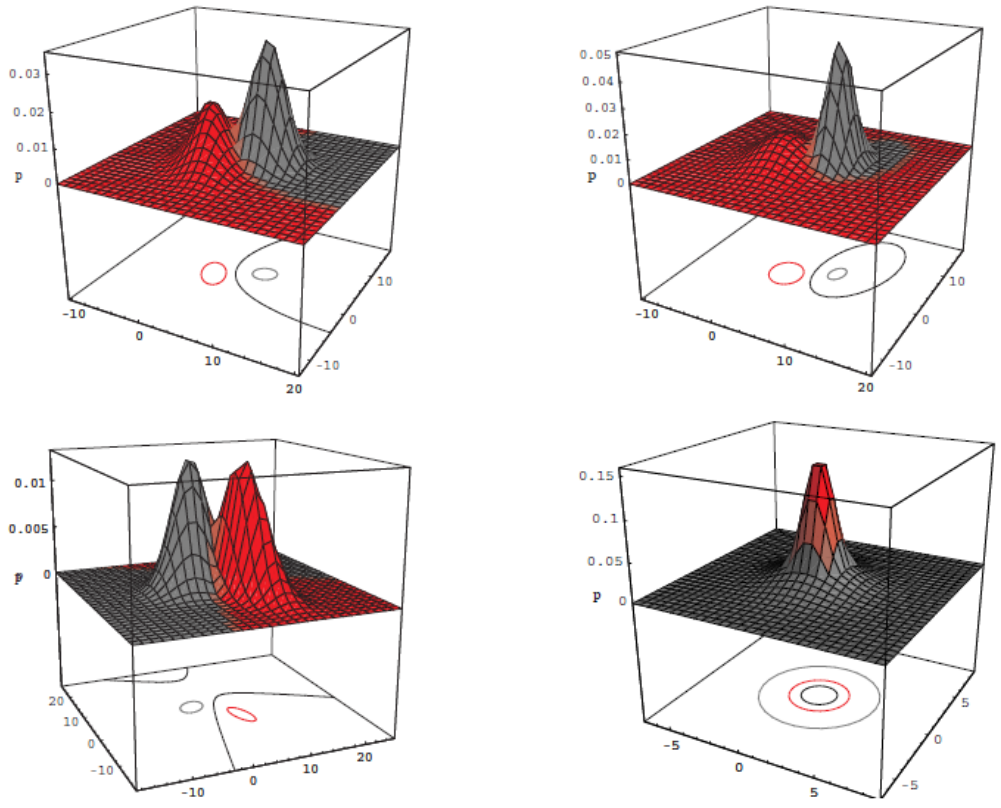


Linear Discriminant Analysis

- $\Sigma_{\omega_1} = \Sigma_{\omega_2} = \Sigma$
- The decision boundaries are linear hyperplanes

Source: A. Fermin

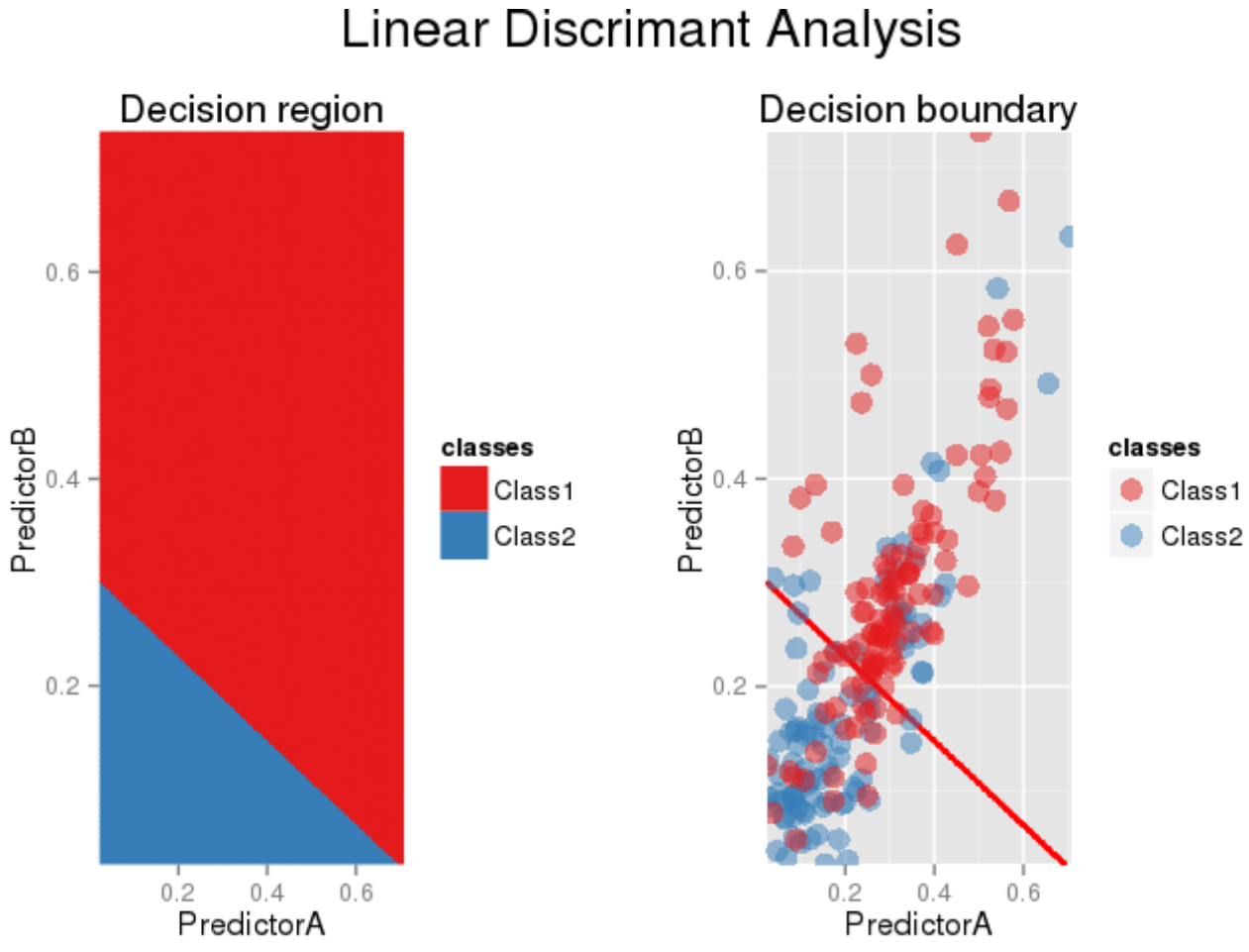
Discriminant Analysis



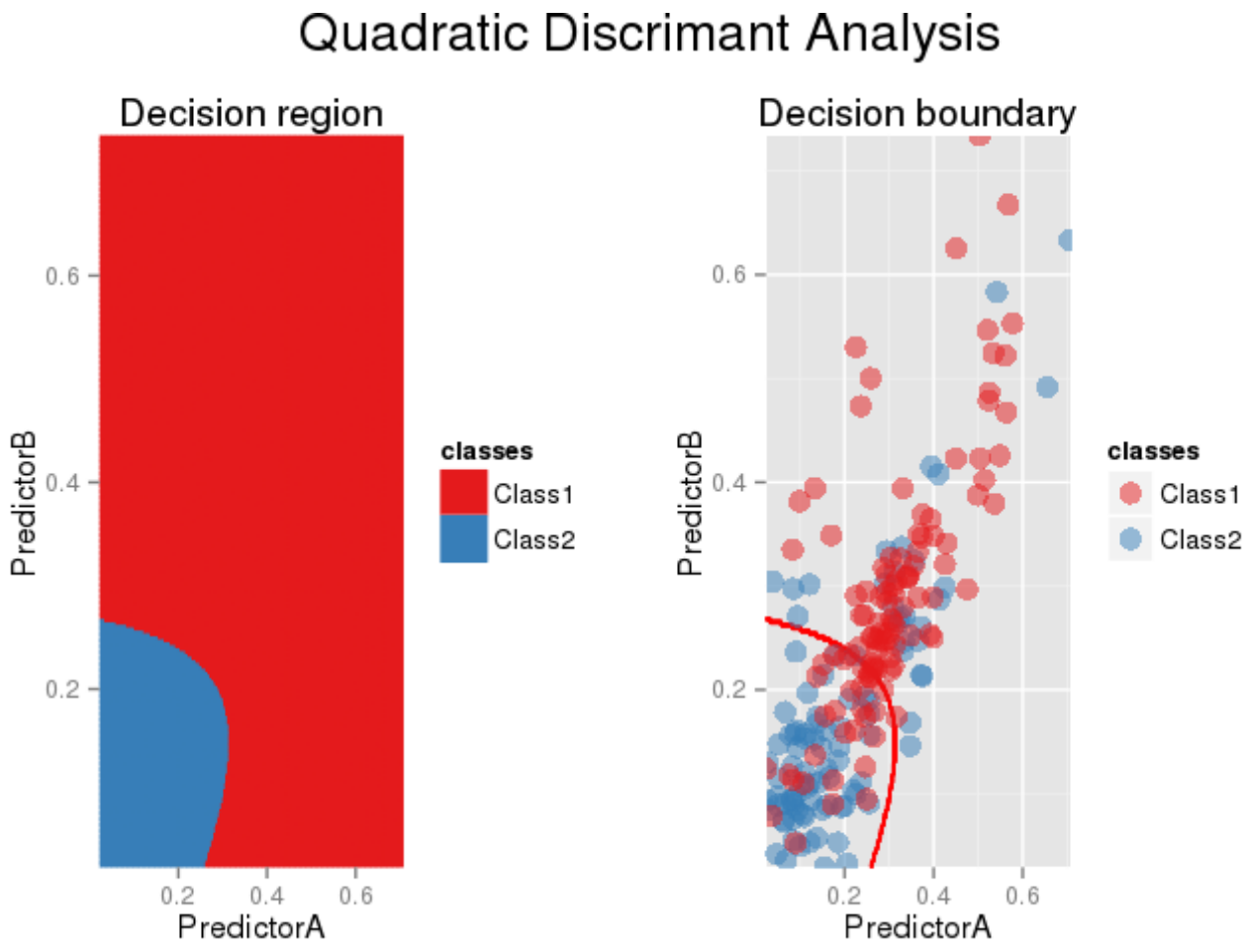
Quadratic Discriminant Analysis

- $\Sigma_{\omega_1} \neq \Sigma_{\omega_2}$
- Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general quadratics.

Example: LDA


Linear Discrimant Analysis
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: QDA


Quadratic Discrimant Analysis
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Naive Bayes

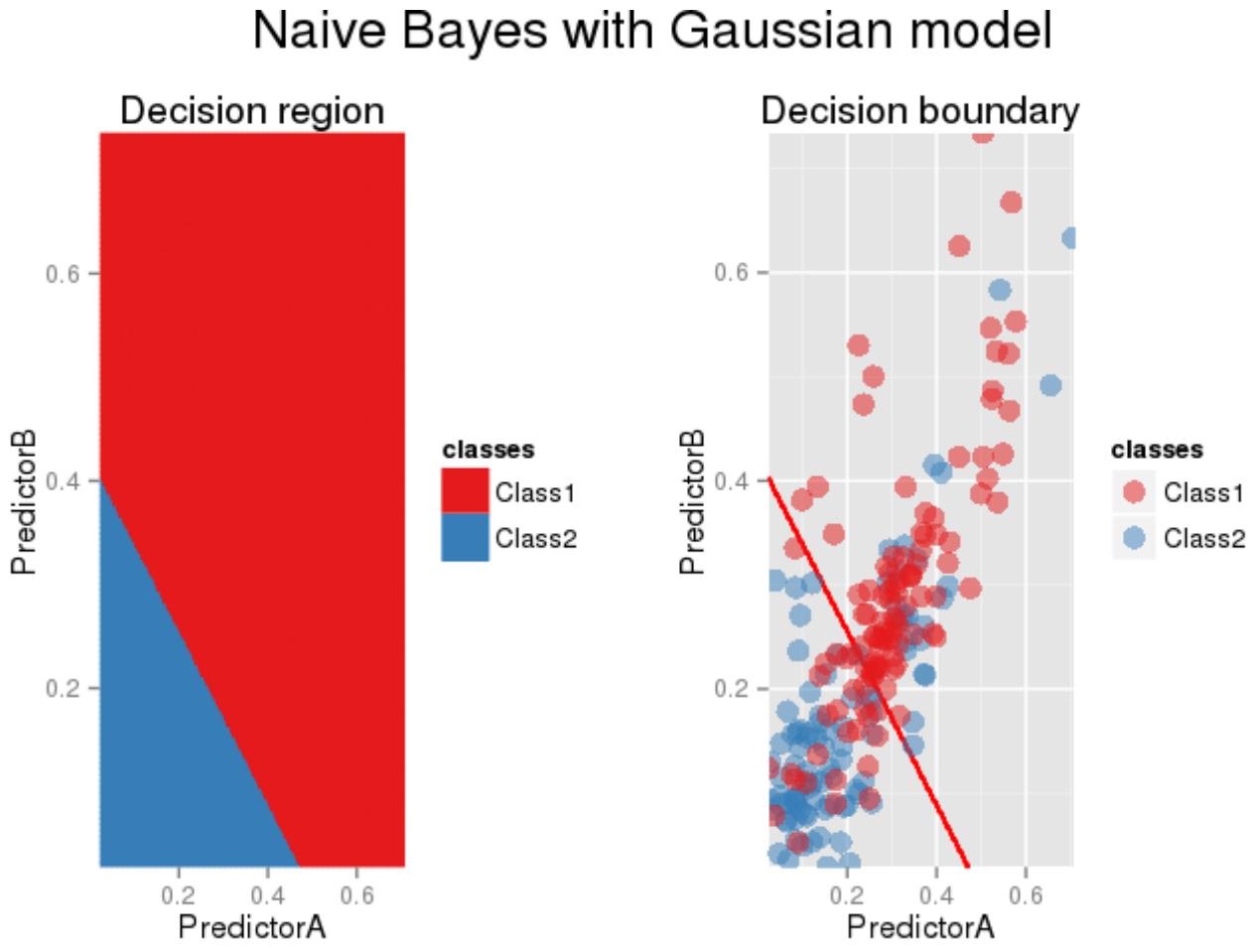
Naive Bayes

- Classical algorithm using a crude modeling for $\mathbb{P}(\underline{X}|Y)$:
  - Feature **independence** assumption:

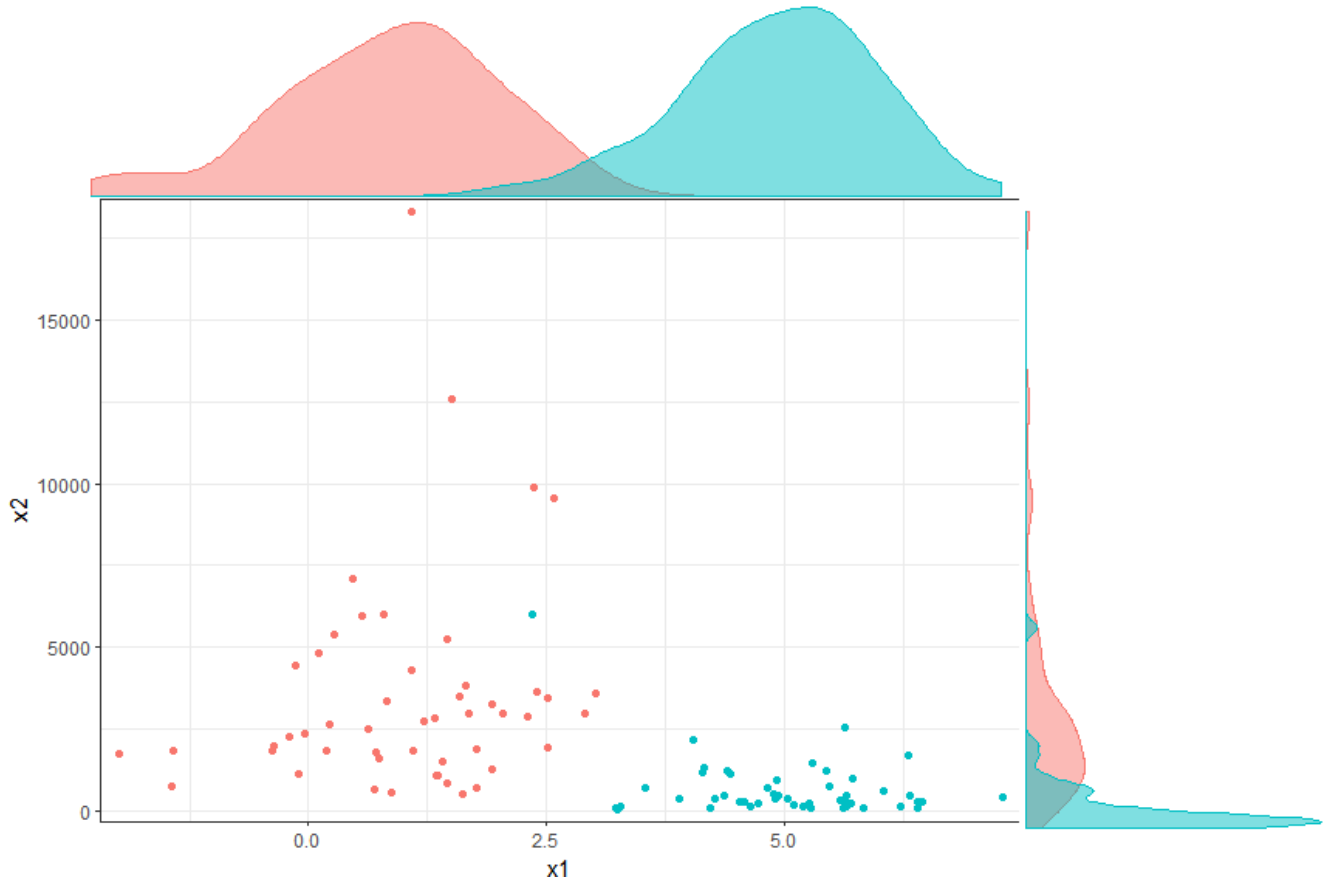
$$\mathbb{P}(\underline{X}|Y) = \prod_{l=1}^{d} \mathbb{P}\left(\underline{X}^{(l)} \middle| Y\right)$$

  - Simple featurewise model: binomial if binary, multinomial if finite and Gaussian if continuous
- If all features are continuous, similar to the previous Gaussian but with a **diagonal covariance matrix**!
- Very simple learning even in **very high dimension!**

Example: Naive Bayes

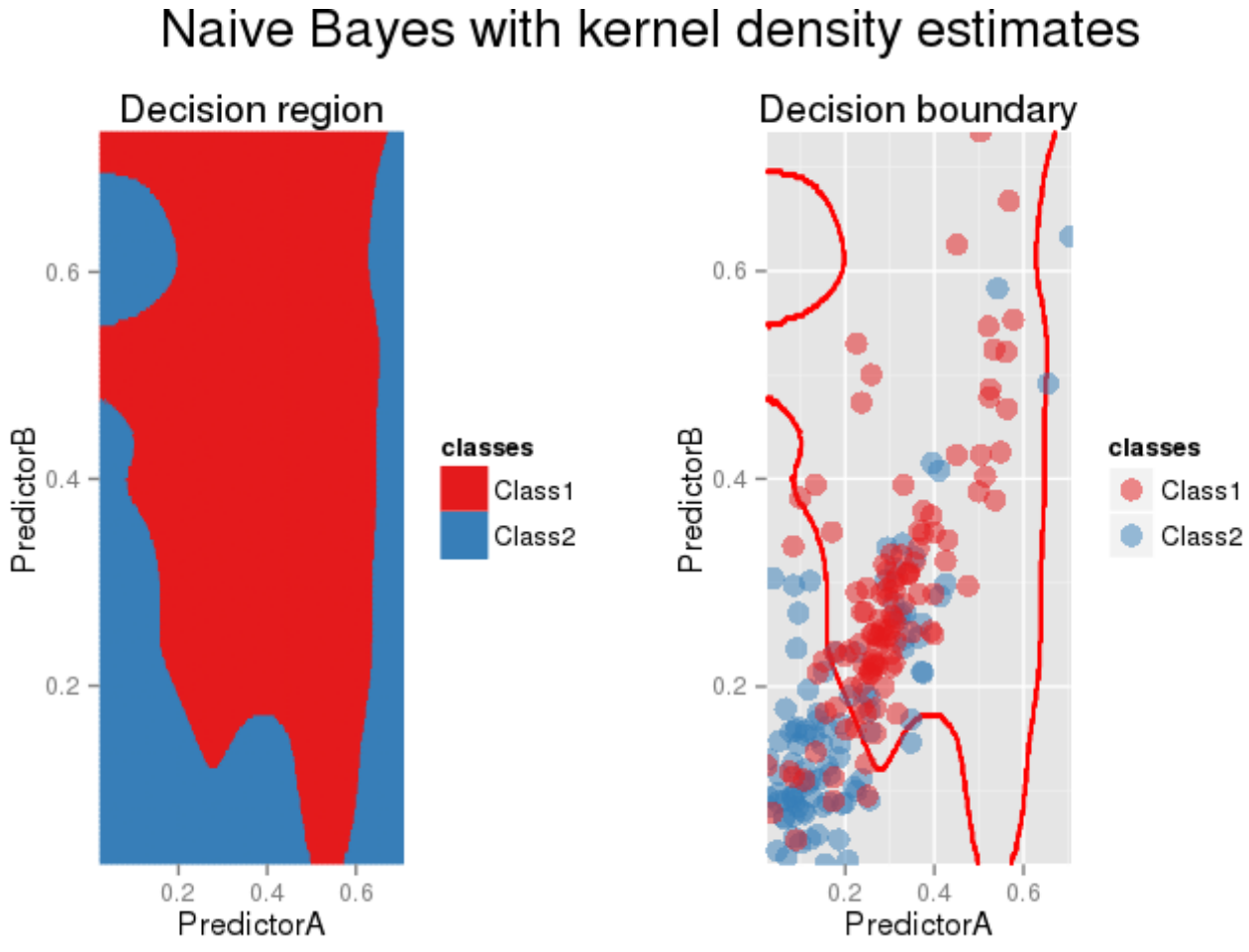


Naive Bayes with density estimation



Source: A. Fermin

Example: Naive Bayes



Other Models

- Other models of the world!

Bayesian Approach

- Generative Model plus prior on the parameters
- Inference thanks to the Bayes formula

Graphical Models

- Markov type models on Graphs

Gaussian Processes

- Multivariate Gaussian models

- ...

Outline

Parametric Conditional Density Models

- **Idea:** Estimate directly $Y|\underline{X}$ by a parametric conditional density $\mathbb{P}_\theta\left(Y|\underline{X}\right)$.

Maximum Likelihood Approach

- Classical choice for θ:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \log \mathbb{P}_\theta\left(Y_i|\underline{X}_i\right)$$

- **Goal:** *Minimize* the Kullback-Leibler divergence between the conditional law of $Y|\underline{X}$ and $\mathbb{P}_\theta\left(Y|\underline{X}\right)$

$$\mathbb{E}\left[\mathrm{KL}\left(Y|\underline{X}, \mathbb{P}_\theta\left(Y|\underline{X}\right)\right)\right]$$

- **Rk:** This is often not (exactly) the learning task!
- Large choice for the family $\{\mathbb{P}_\theta\left(Y|\underline{X}\right)\}$ but depends on $\mathcal{Y}$ (and $\mathcal{X}$).
- **Regression:** One can also model directly $\mathbb{E}\left[Y|\underline{X}\right]$ by $f_\theta(\underline{X})$ and estimate it with a least square criterion...

Linear Conditional Density Models

Linear Models

- **Classical choice:** $\theta = (\theta', \varphi)$

$$\mathbb{P}_\theta\left(Y|\underline{X}\right) = \mathbb{P}_{\underline{X}^\top\beta,\varphi}\left(Y\right)$$

- **Very strong assumption!**

- Classical examples:
  - Binary variable: logistic, probit...
  - Discrete variable: multinomial logistic regression...
  - Integer variable: Poisson regression...
  - Continuous variable: Gaussian regression...

Binary Classifier

Plugin Linear Discrimination

- Model $\mathbb{P}(Y = +1|\underline{X})$ by $h(\underline{x}^\top \beta + \beta^{(0)})$ with h non decreasing.
- $h(\underline{x}^\top \beta + \beta^{(0)}) > 1/2 \Leftrightarrow \underline{x}^\top \beta + \beta^{(0)} - h^{-1}(1/2) > 0$
- Linear Classifier: $\text{sign}(\underline{x}^\top \beta + \beta^{(0)} - h^{-1}(1/2))$

Plugin Linear Classifier Estimation

- Classical choice for h:

$$h(t) = \frac{e^t}{1+e^t} \qquad \text{logit or logistic}$$

$$h(t) = F_{\mathcal{N}}(t) \qquad \text{probit}$$

$$h(t) = 1 - e^{-e^t} \qquad \text{log-log}$$

- Choice of the *best* β from the data.

Maximum Likelihood Estimate

Probabilistic Model

- By construction, $Y|\underline{X}$ follows $\mathcal{B}(\mathbb{P}(Y = +1|\underline{X}))$
- Approximation of $Y|\underline{X}$ by $\mathcal{B}(h(\underline{x}^\top \beta + \beta^{(0)}))$
- *Natural* probabilistic choice for β: β minimizing the distance between $\mathcal{B}(h(\underline{x}^\top \beta))$ and $\mathcal{B}(\mathbb{P}(Y = 1|\underline{X}))$.

KL Distance

- *Natural* distance: Kullback-Leibler divergence

$$\begin{aligned}
&\mathrm{KL}(\mathcal{B}(\mathbb{P}(Y = 1|\underline{X})), \mathcal{B}(h(\underline{x}^\top \beta)) \\
&\quad = \mathbb{E}_{\underline{X}}\left[\mathrm{KL}(\mathcal{B}(\mathbb{P}(Y = 1|\underline{X})), \mathcal{B}(h(\underline{x}^\top \beta))\right] \\
&\quad = \mathbb{E}_{\underline{X}}\left[\mathbb{P}(Y = 1|\underline{X}) \log \frac{\mathbb{P}(Y = 1|\underline{X})}{h(\underline{x}^\top \beta)}\right. \\
&\qquad\qquad \left. + (1 - \mathbb{P}(Y = 1|\underline{X})) \log \frac{1 - \mathbb{P}(Y = 1|\underline{X})}{1 - h(\underline{x}^\top \beta)}\right]
\end{aligned}$$

Maximum Likelihood Estimate

log-likelihood

- KL:

$$\begin{aligned}
&\mathrm{KL}(\mathcal{B}(\mathbb{P}(Y=1|\underline{X})), \mathcal{B}(h(\underline{x}^\top\beta)) \\
&\quad = \mathbb{E}_{\underline{X}}\Big[\mathbb{P}(Y=1|\underline{X}) \log \frac{\mathbb{P}(Y=1|\underline{X})}{h(\underline{x}^\top\beta)} \\
&\qquad\qquad + (1-\mathbb{P}(Y=1|\underline{X})) \log \frac{1-\mathbb{P}(Y=1|\underline{X})}{1-h(\underline{x}^\top\beta)}\Big] \\
&\quad = \mathbb{E}_{\underline{X}}\Big[-\mathbb{P}(Y=1|\underline{X}) \log(h(\underline{x}^\top\beta)) \\
&\qquad\qquad - (1-\mathbb{P}(Y=1|\underline{X})) \log(1-h(\underline{x}^\top\beta))\Big] + C_{\underline{X},Y}
\end{aligned}$$

- Empirical counterpart = opposite of the log-likelihood:

$$-\frac{1}{n}\sum_{i=1}^{n}\Big(\mathbf{1}_{Y_i=1}\log(h(\underline{X}_i{}^\top\beta)) + \mathbf{1}_{Y_i=-1}\log(1-h(\underline{X}_i{}^\top\beta))\Big)$$

- Minimization of possible if h is regular...

Logistic Regression

Logistic Regression and Odd

- Logistic model: $h(t) = \frac{e^t}{1+e^t}$ (most *natural* choice...)
- The Bernoulli law $\mathcal{B}(h(t))$ satisfies then

$$\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1)} = e^t \Leftrightarrow \log \frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1)} = t$$

- Interpretation in term of odd.
- Logistic model: linear model on the logarithm of the odd.

Associated Classifier

- Plugin strategy:

$$f_\beta(\underline{X}) = \begin{cases} 1 & \text{if } \frac{e^{\underline{X}^\top \beta}}{1+e^{\underline{X}^\top \beta}} > 1/2 \Leftrightarrow \underline{X}^\top \beta > 0 \\ -1 & \text{otherwise} \end{cases}$$

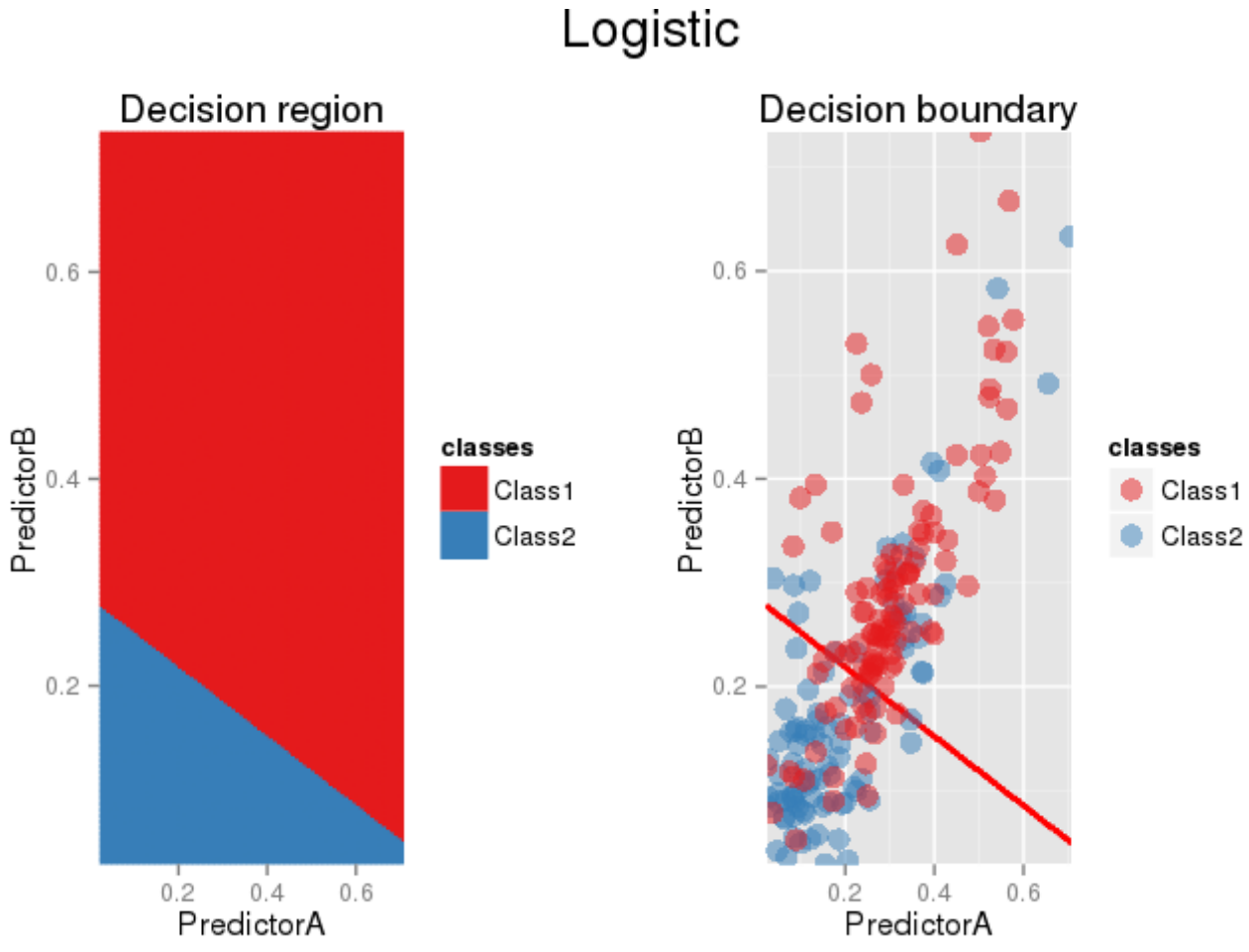
Likelihood Rewriting

- Opposite of the log-likelihood:

$$
\begin{aligned}
&-\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{Y_i=1}\log(h(\underline{X}_i{}^\top\beta))+\mathbf{1}_{Y_i=-1}\log(1-h(\underline{X}_i{}^\top\beta))\right)\\
&=-\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{Y_i=1}\log\frac{e^{\underline{X}_i{}^\top\beta}}{1+e^{\underline{X}^\top\beta}}+\mathbf{1}_{Y_i=-1}\log\frac{1}{1+e^{\underline{X}_i{}^\top\beta}}\right)\\
&=\frac{1}{n}\sum_{i=1}^{n}\log\left(1+e^{-Y_i(\underline{X}_i{}^\top\beta)}\right)
\end{aligned}
$$

- Convex and smooth function of β
- Easy optimization.

Example: Logistic



Feature Design

Transformed Representation

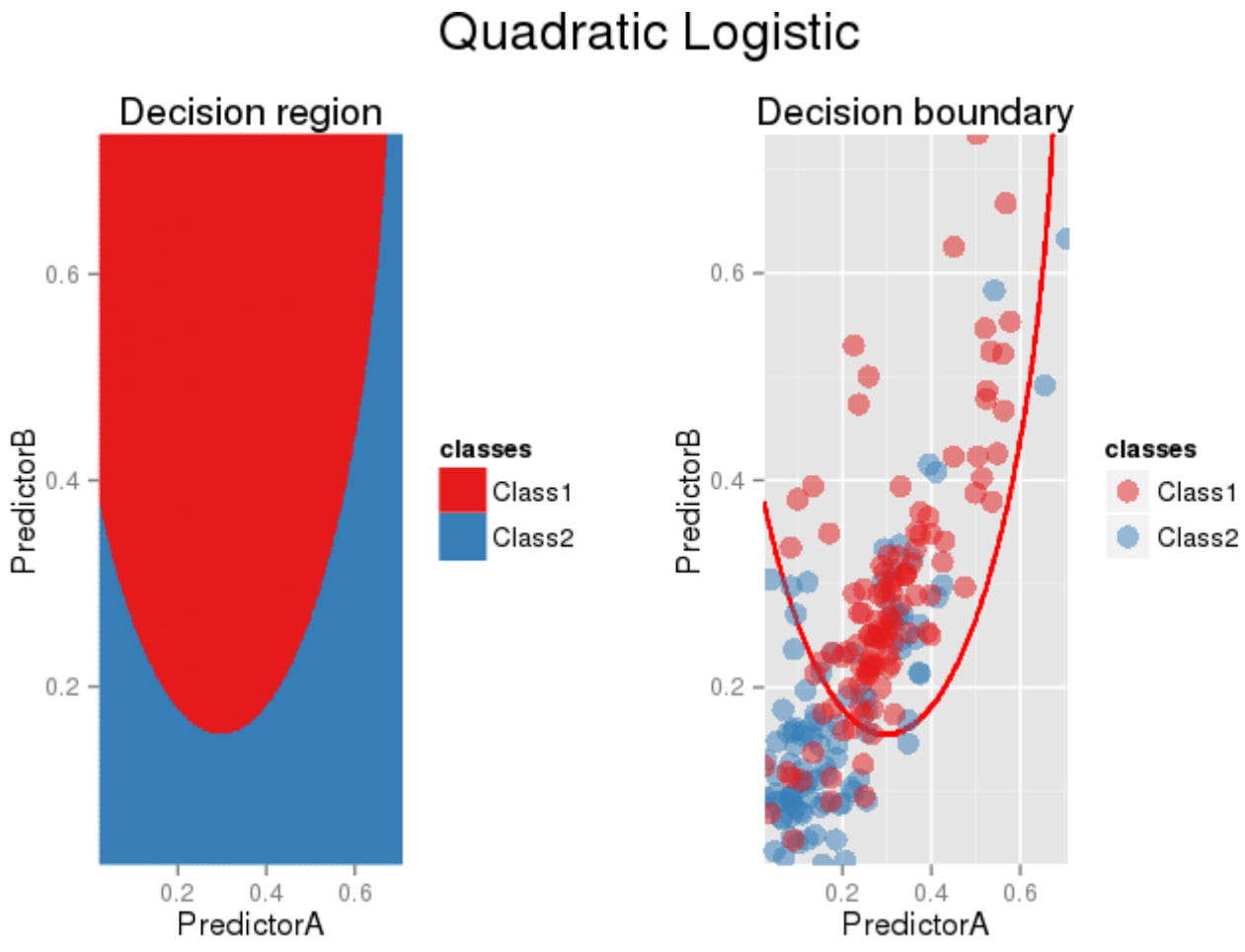
- From $\underline{X}$ to $\Phi(\underline{X})$!
- New description of $\underline{X}$ leads to a different **linear** model:

$$f_\beta(\underline{X}) = \Phi(\underline{X})^\top \beta$$

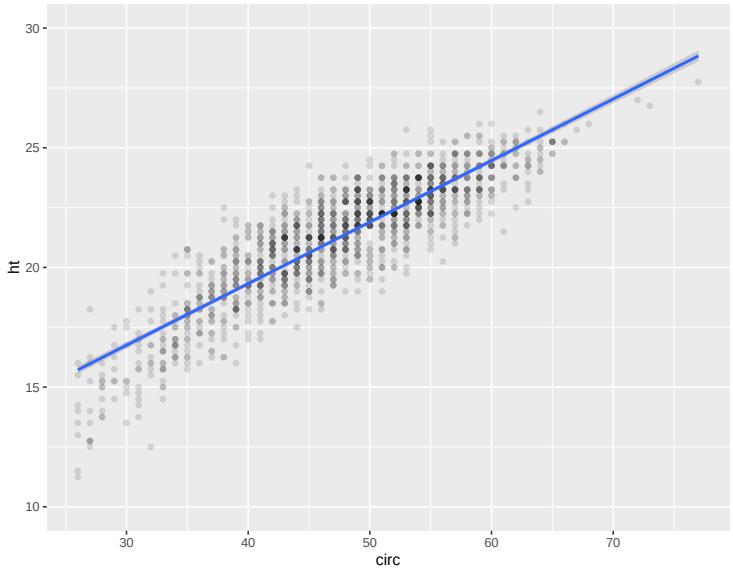
Feature Design

- Art of choosing Φ.
- Examples:
  - Renormalization, (domain specific) transform
  - Basis decomposition
  - Interaction between different variables...

Example: Quadratic Logistic



Gaussian Linear Regression



Gaussian Linear Model

- **Model:** $Y|\underline{X} \sim \mathcal{N}(\underline{x}^\top \beta, \sigma^2)$ plus independence
- Probably the most classical model of all time!
- Maximum Likelihood with explicit formulas for the two parameters.

- In regression, estimation of $\mathbb{E}[Y|\underline{X}]$ is sufficient: other/no model for the noise possible.

Extension of Gaussian Linear Regression

Generalized Linear Model

- Model entirely characterized by its mean (up to a scalar nuisance parameter) ($v(\mathbb{E}_\theta[Y]) = \theta$ with v invertible).
- Exponential family: Probability law family P_θ such that the density can be written

$$f(y, \theta, \varphi) = e^{\frac{y\theta - v(\theta)}{\varphi} + w(y,\varphi)}$$

  where φ is a nuisance parameter and w a function independent of θ.
- Examples:
  - Gaussian: $f(y, \theta, \varphi) = e^{\frac{y\theta - \theta^2/2}{\varphi} + \frac{y^2/2}{\varphi}}$
  - Bernoulli: $f(y, \theta) = e^{z\theta - \ln(1+e^\theta)}$ $(\theta = \ln p/(1-p))$
  - Poisson: $f(y, \theta) = e^{(y\theta - e^\theta) + \ln(y!)}$ $(\theta = \ln \lambda)$
- Linear Conditional model: $Y|\underline{X} \sim P_{\underline{x}^\top \beta}...$

- ML fit of the parameters

Outline

Non Parametric Conditional Estimation

- **Idea:** Estimate $Y|\underline{X}$ or $\mathbb{E}[Y|\underline{X}]$ directly without resorting to an explicit parametric model.

Non Parametric Conditional Estimation

- Two heuristics:
  - $Y|\underline{X}$ (or $\mathbb{E}[Y|\underline{X}]$) is almost constant (or simple) in a neighborhood of $\underline{X}$. (Kernel methods)
  - $Y|\underline{X}$ (or $\mathbb{E}[Y|\underline{X}]$) can be approximated by a model whose dimension depends on the complexity and the number of observation. (Quite similar to parametric model plus model selection...)

- Focus on **kernel methods**!

Kernel Methods

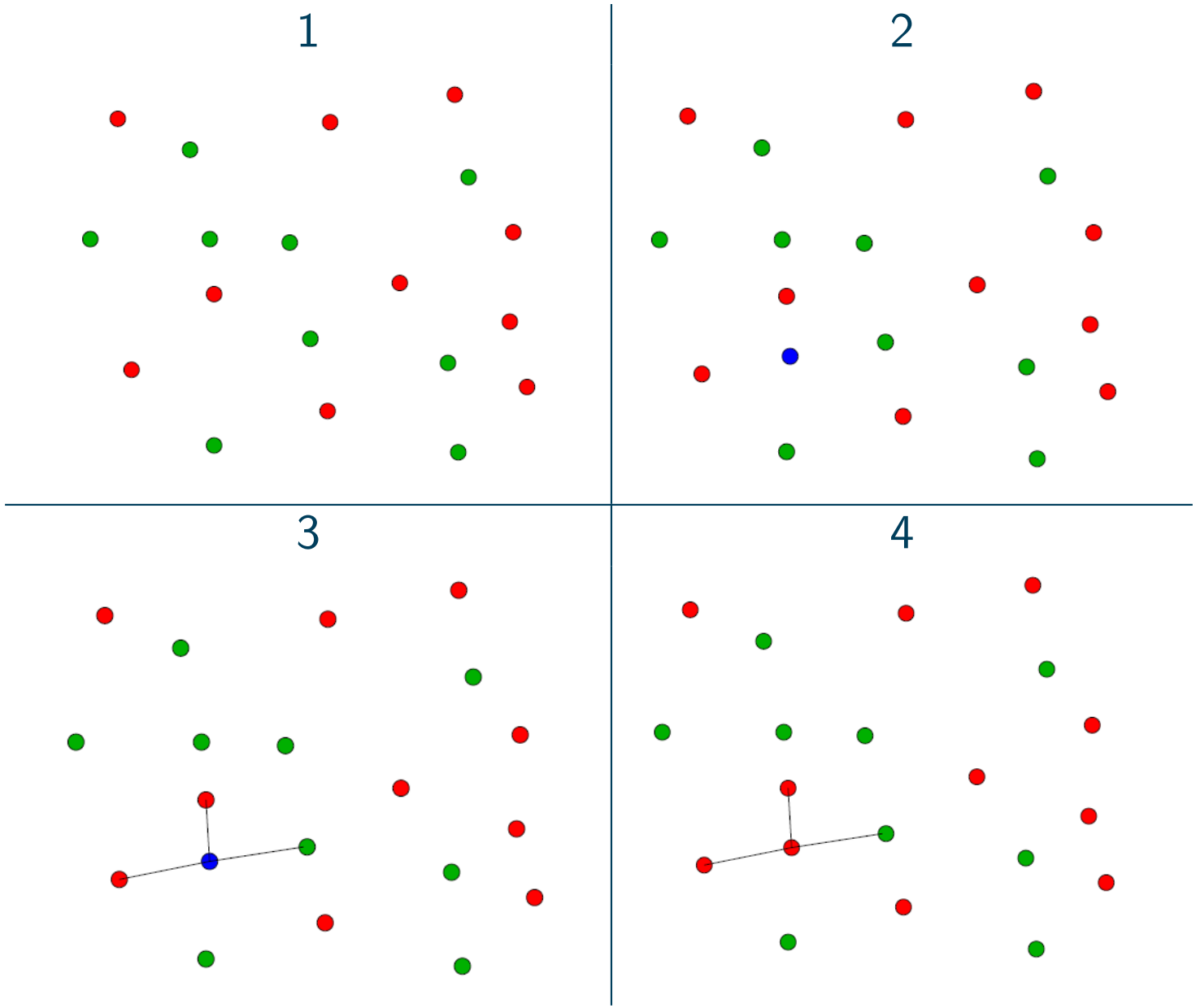
- **Idea:** The behavior of $Y|\underline{X}$ is locally *constant* or simple!

Kernel

- Choose a kernel K (think of a weighted neighborhood).
- For each $\widetilde{\underline{X}}$, compute a simple localized estimate of $Y|\underline{X}$
- Use this local estimate to take the decision

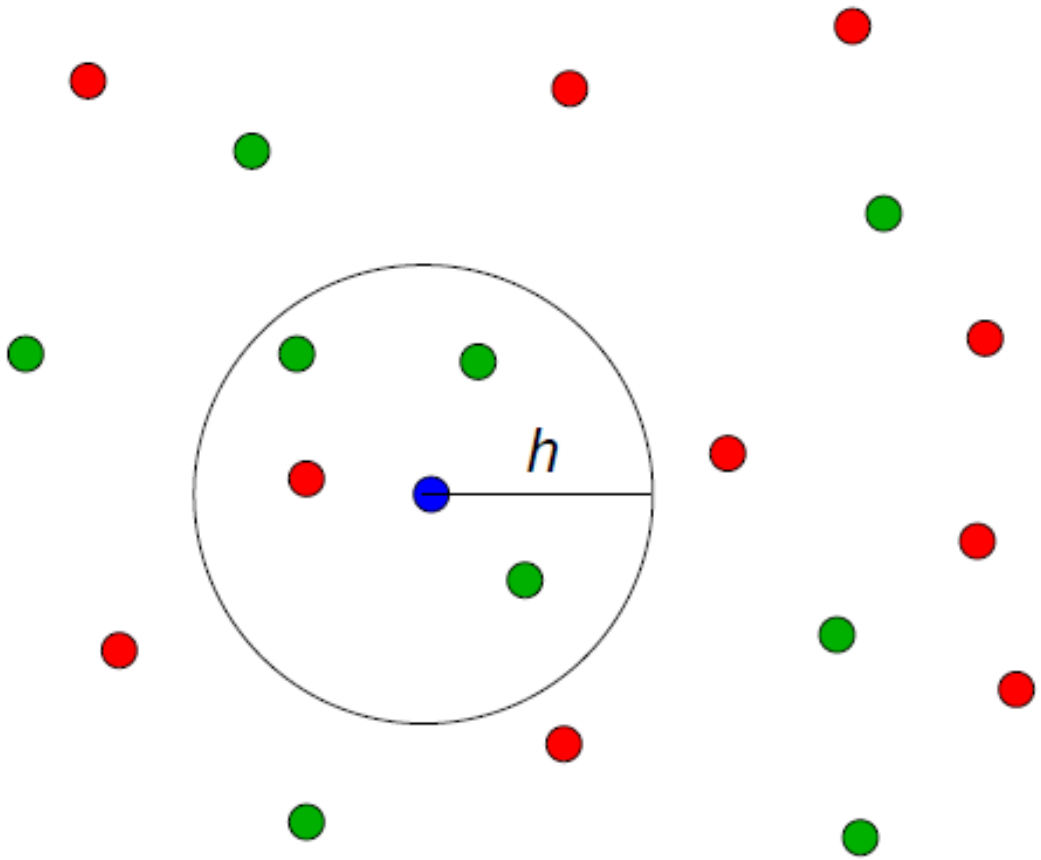
- In regression, estimation of $\mathbb{E}[Y|\underline{X}]$ is sufficient.

Example: k Nearest-Neighbors (with $k = 3$)



Source: Unknown

Example: k Nearest-Neighbors (with $k = 4$)



Source: Unknown

k Nearest-Neighbors

- Neighborhood $\mathcal{V}_{\underline{x}}$ of $\underline{x}$: k learning samples closest from $\underline{x}$.

k-NN as local conditional density estimate

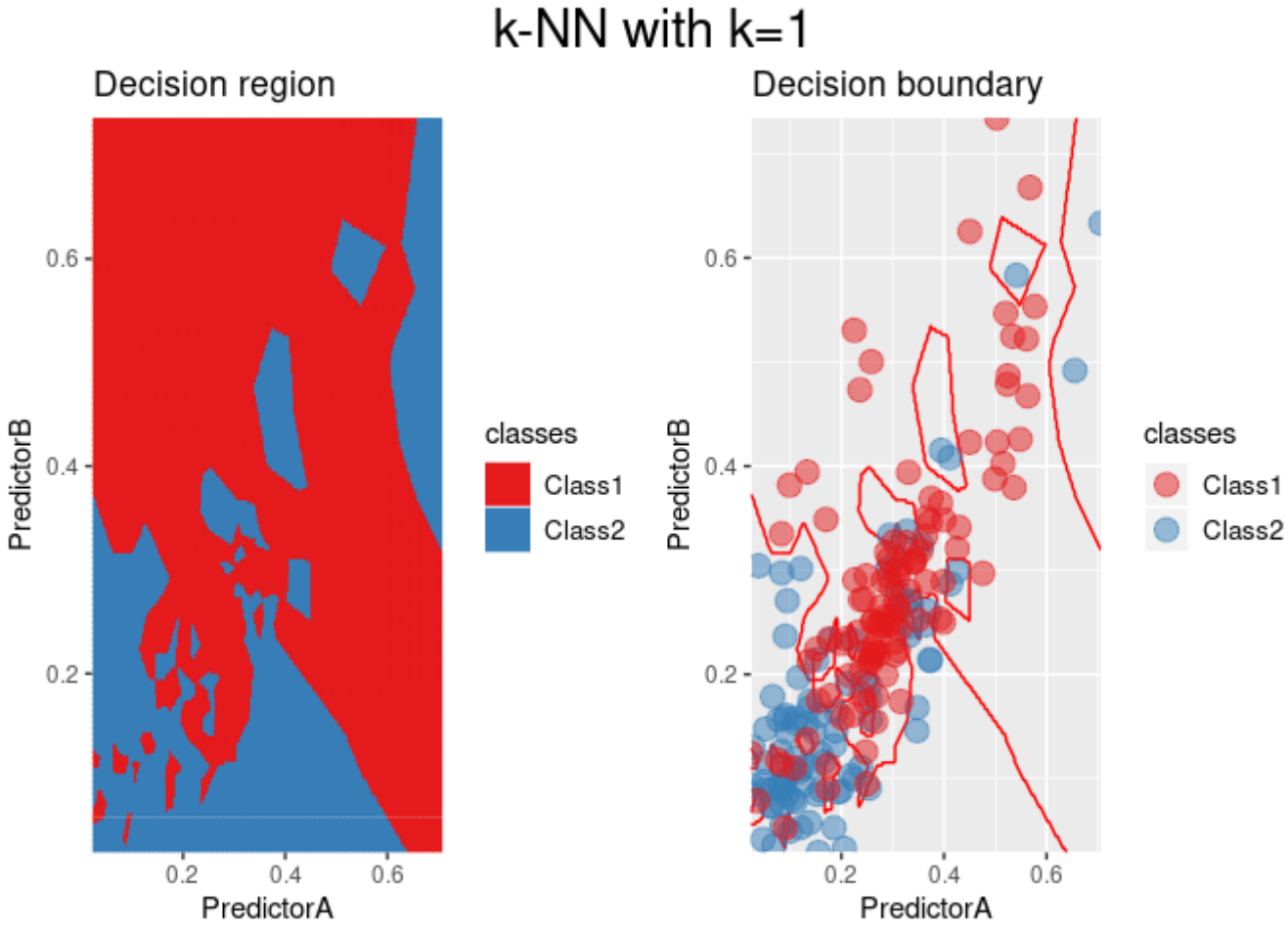
$$\widehat{\mathbb{P}(Y=1|\underline{X})} = \frac{\sum_{\underline{X}_i \in \mathcal{V}_{\underline{x}}} \mathbf{1}_{\{Y_i=+1\}}}{|\mathcal{V}_{\underline{X}}|}$$

- KNN Classifier:

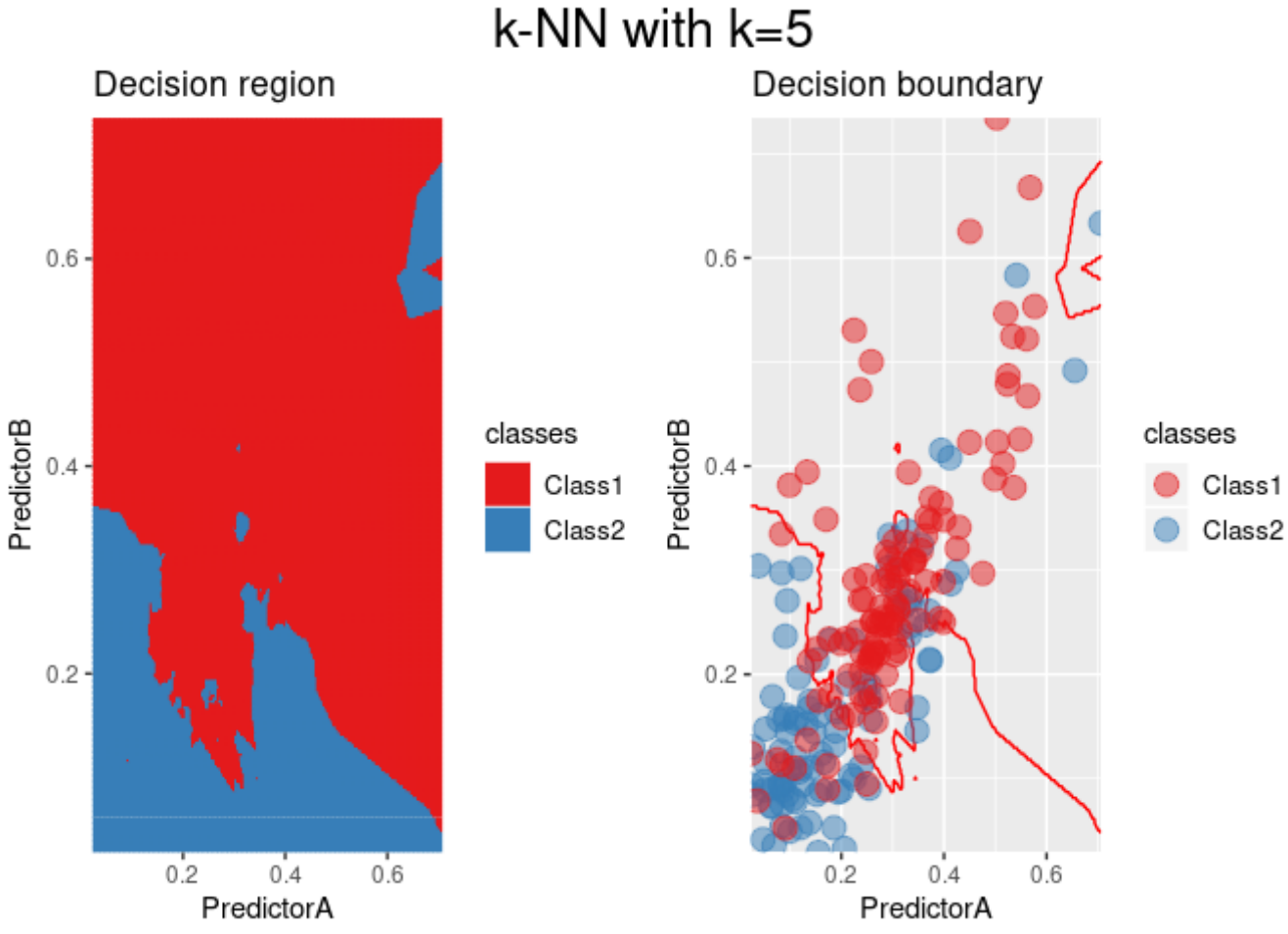
$$\widehat{f}_{KNN}(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{\mathbb{P}(Y=1|\underline{X})} \geq \widehat{\mathbb{P}(Y=-1|\underline{X})} \\ -1 & \text{otherwise} \end{cases}$$

- **Lazy learning**: all the computations have to be done at prediction time.
- **Remark:** You can also use your favorite kernel estimator...

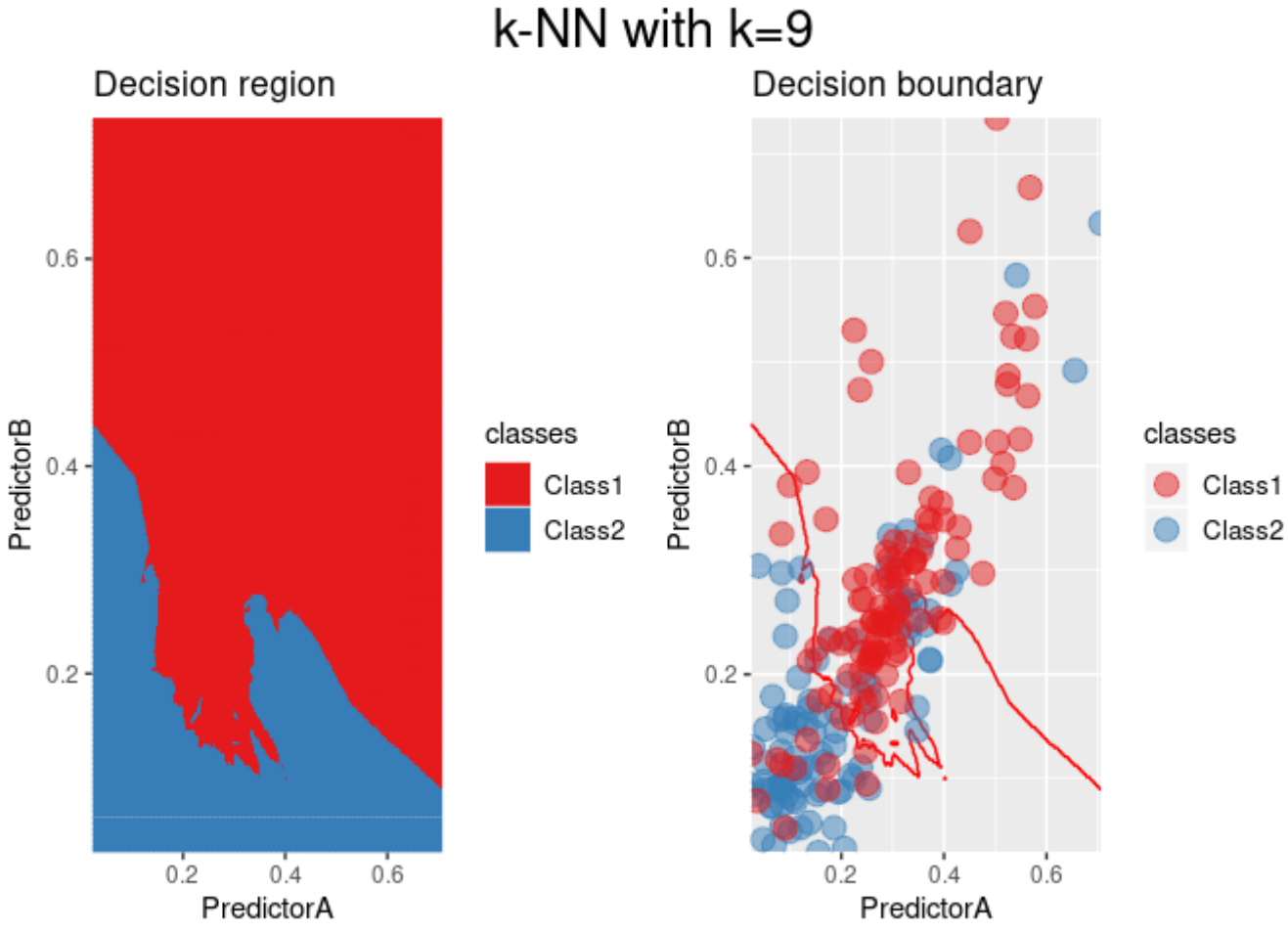
Example: KNN


k-NN with k=1
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

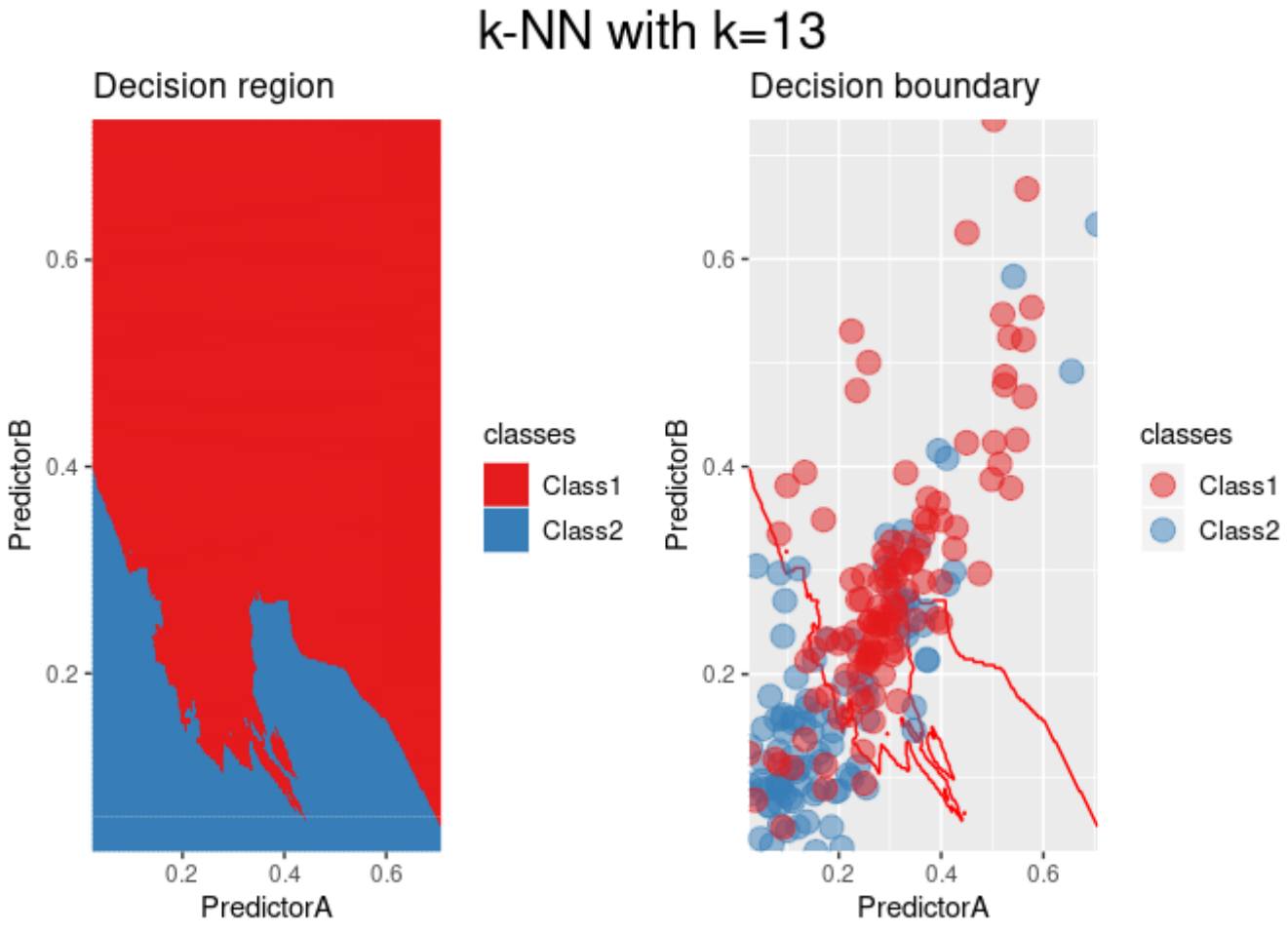
Example: KNN


k-NN with k=5
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

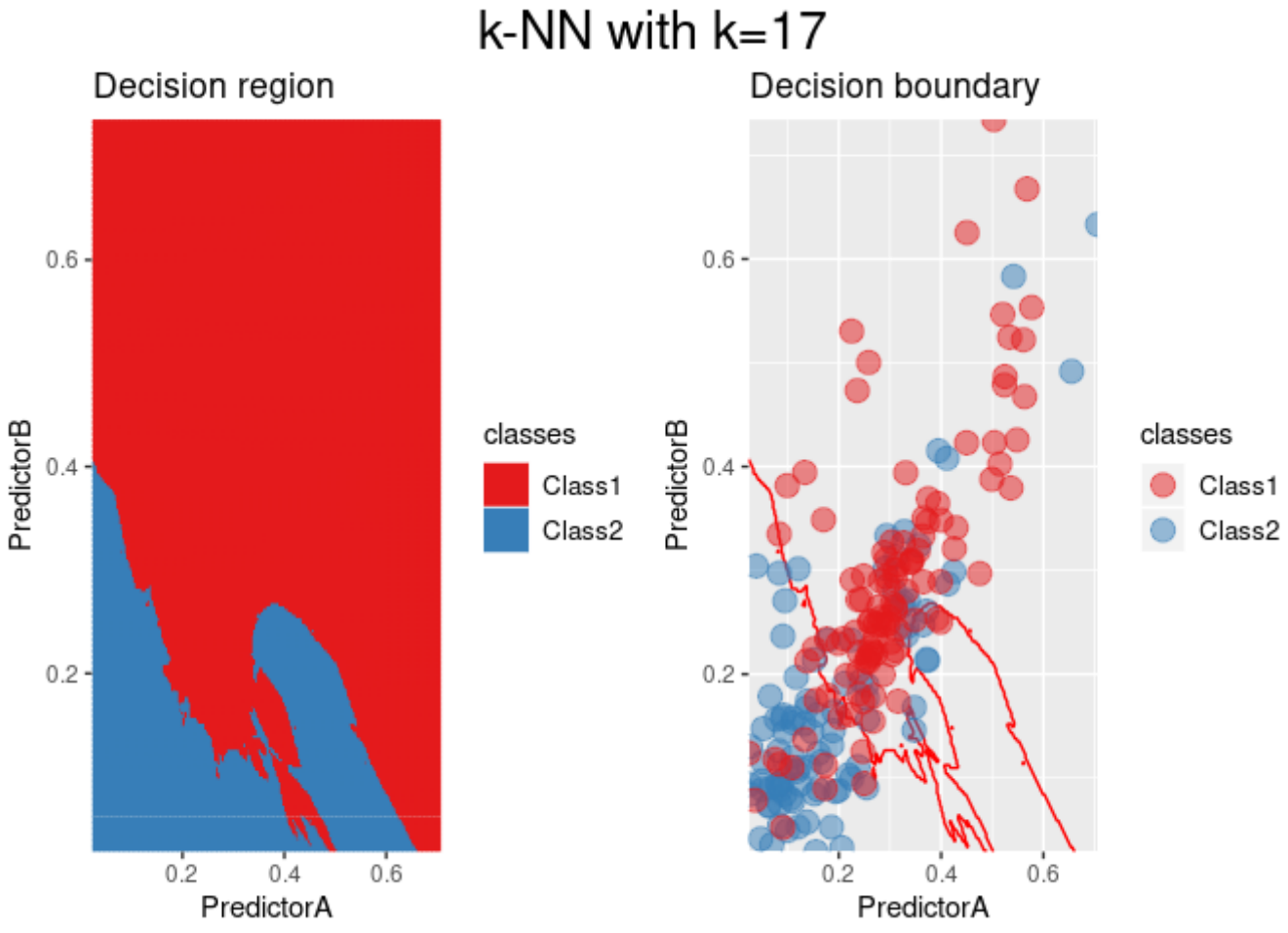
Example: KNN


k-NN with k=9
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

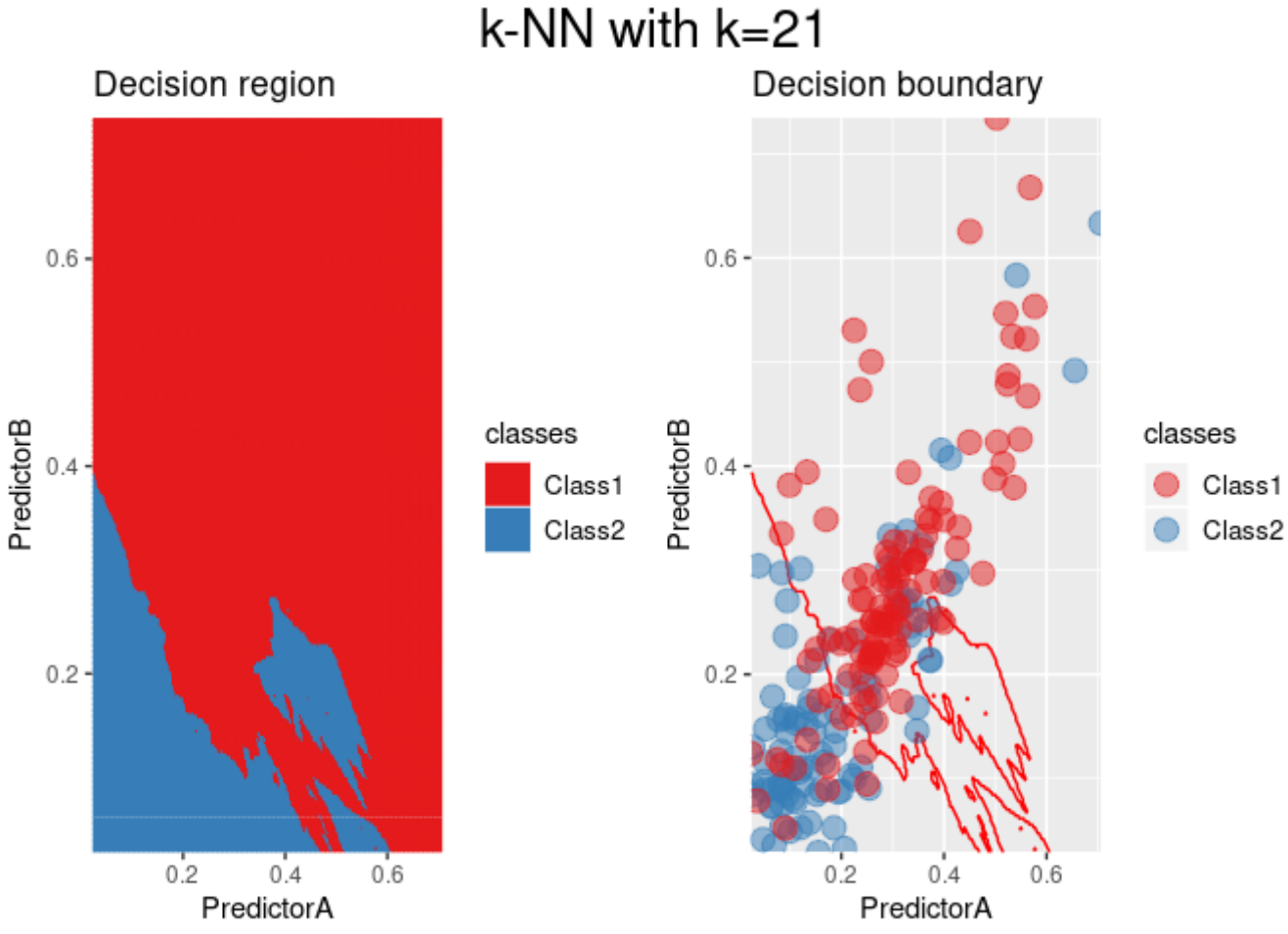
Example: KNN


k-NN with k=13
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

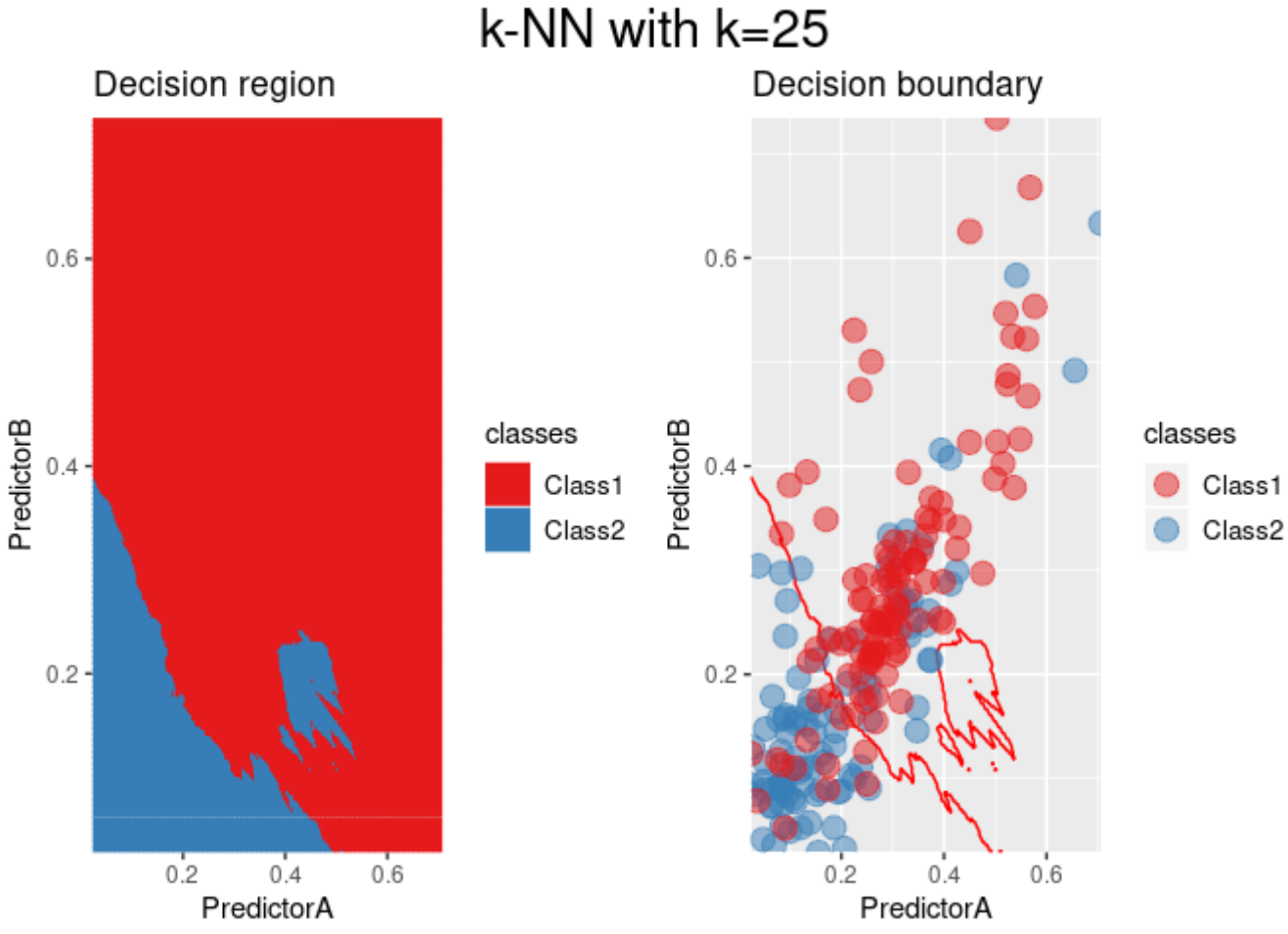
Example: KNN


k-NN with k=17
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

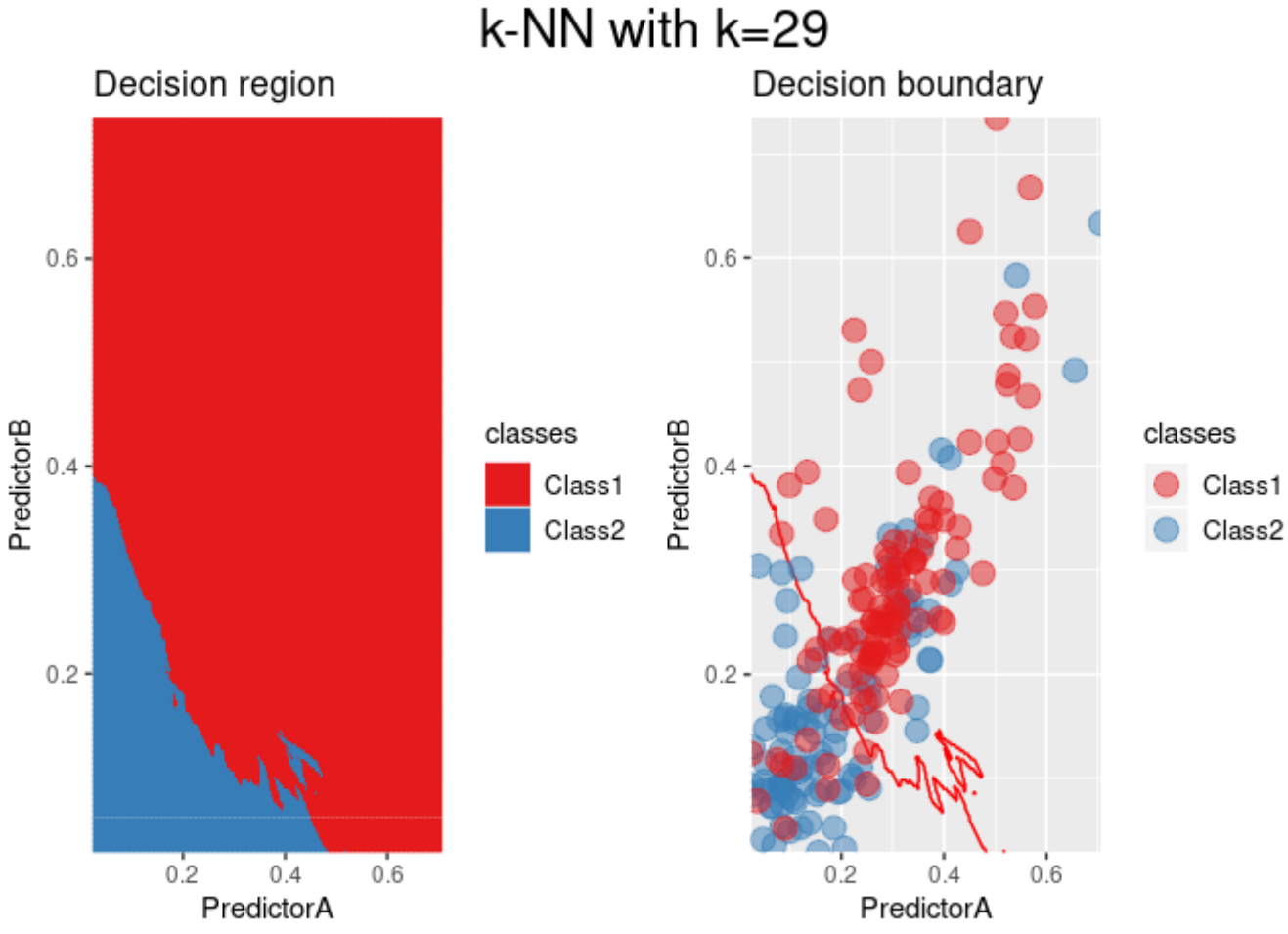
Example: KNN


k-NN with k=21
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

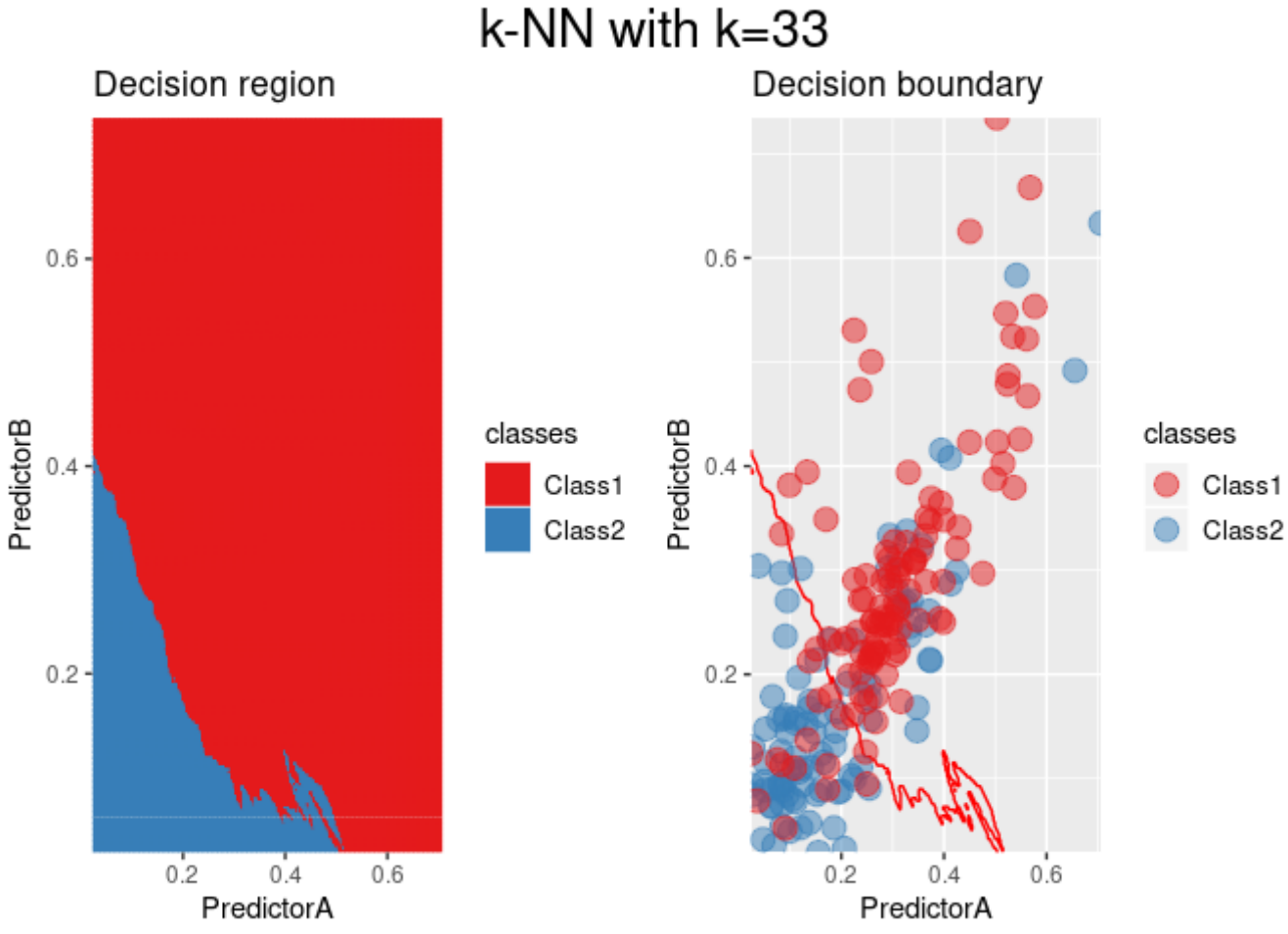
Example: KNN


k-NN with k=25
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

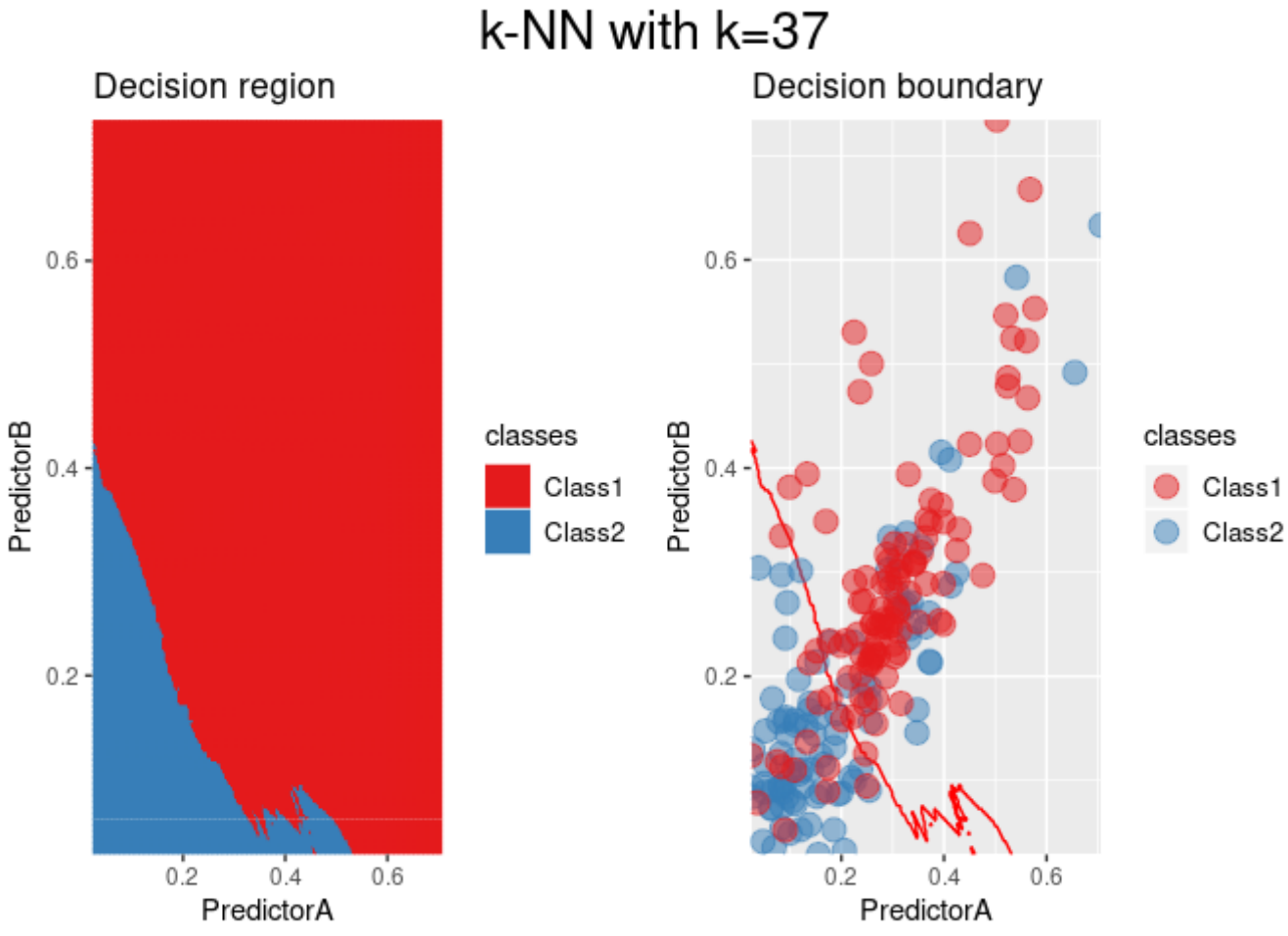
Example: KNN


k-NN with k=29
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

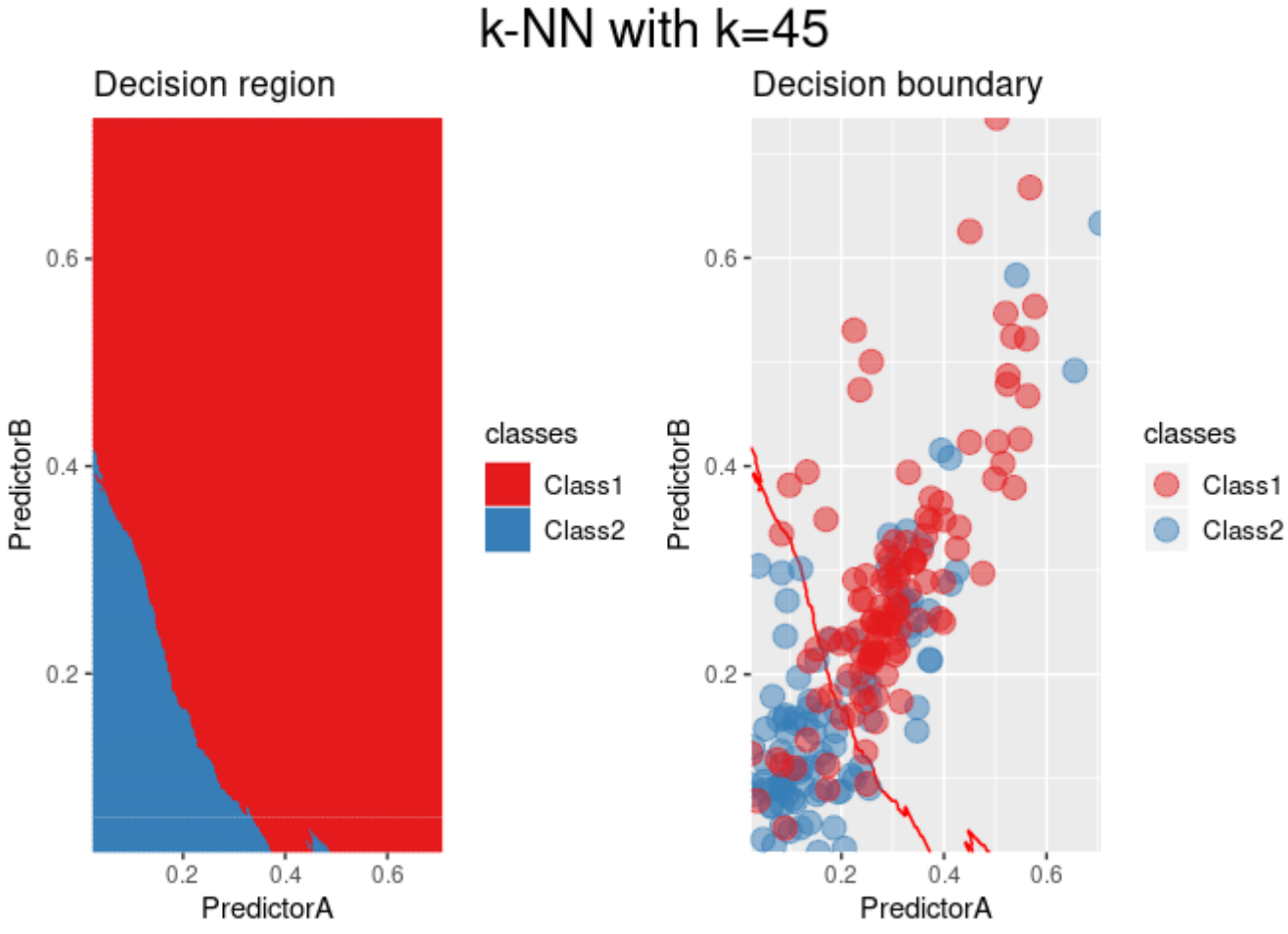
Example: KNN


k-NN with k=33
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

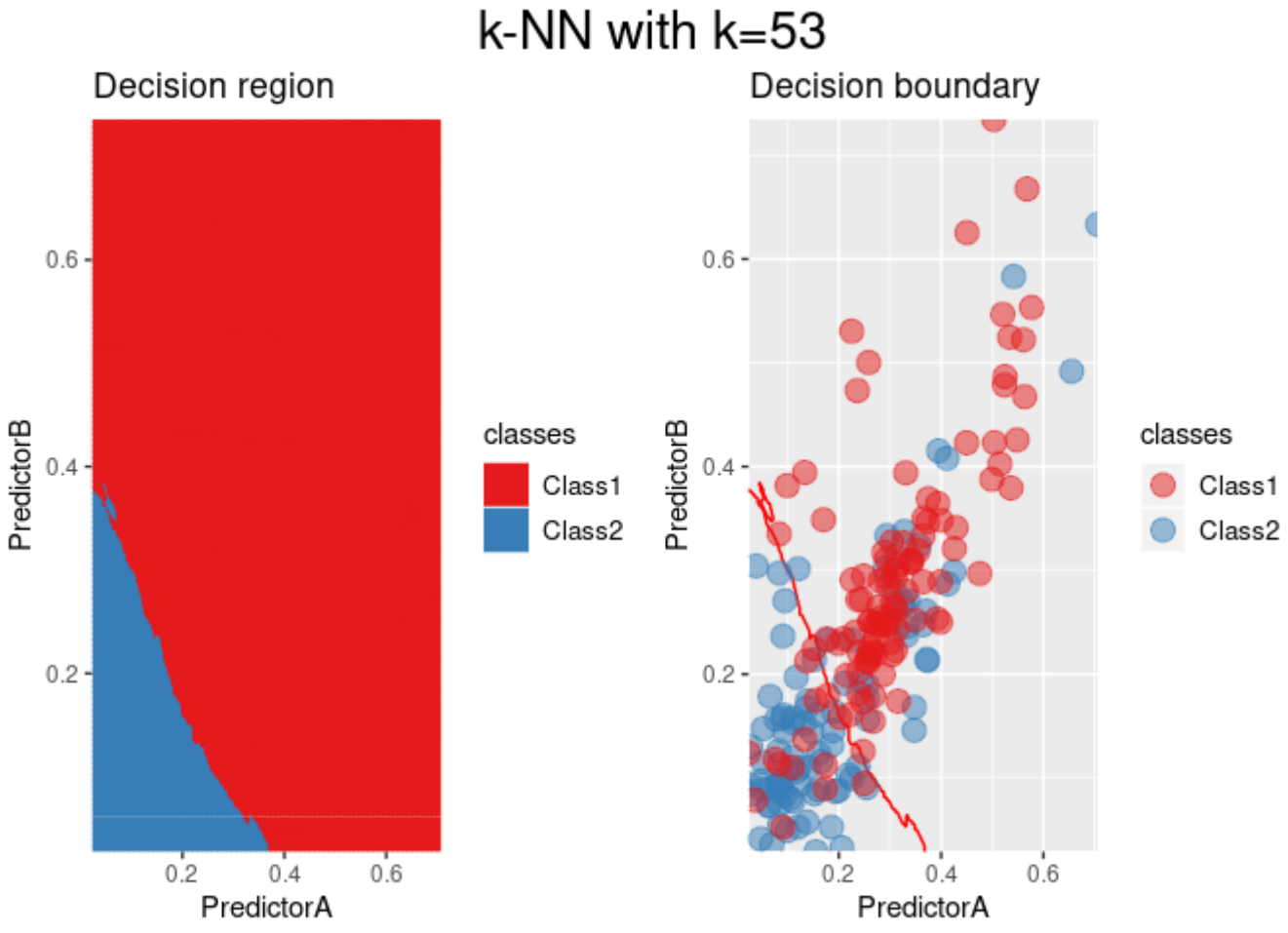
Example: KNN


k-NN with k=37
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

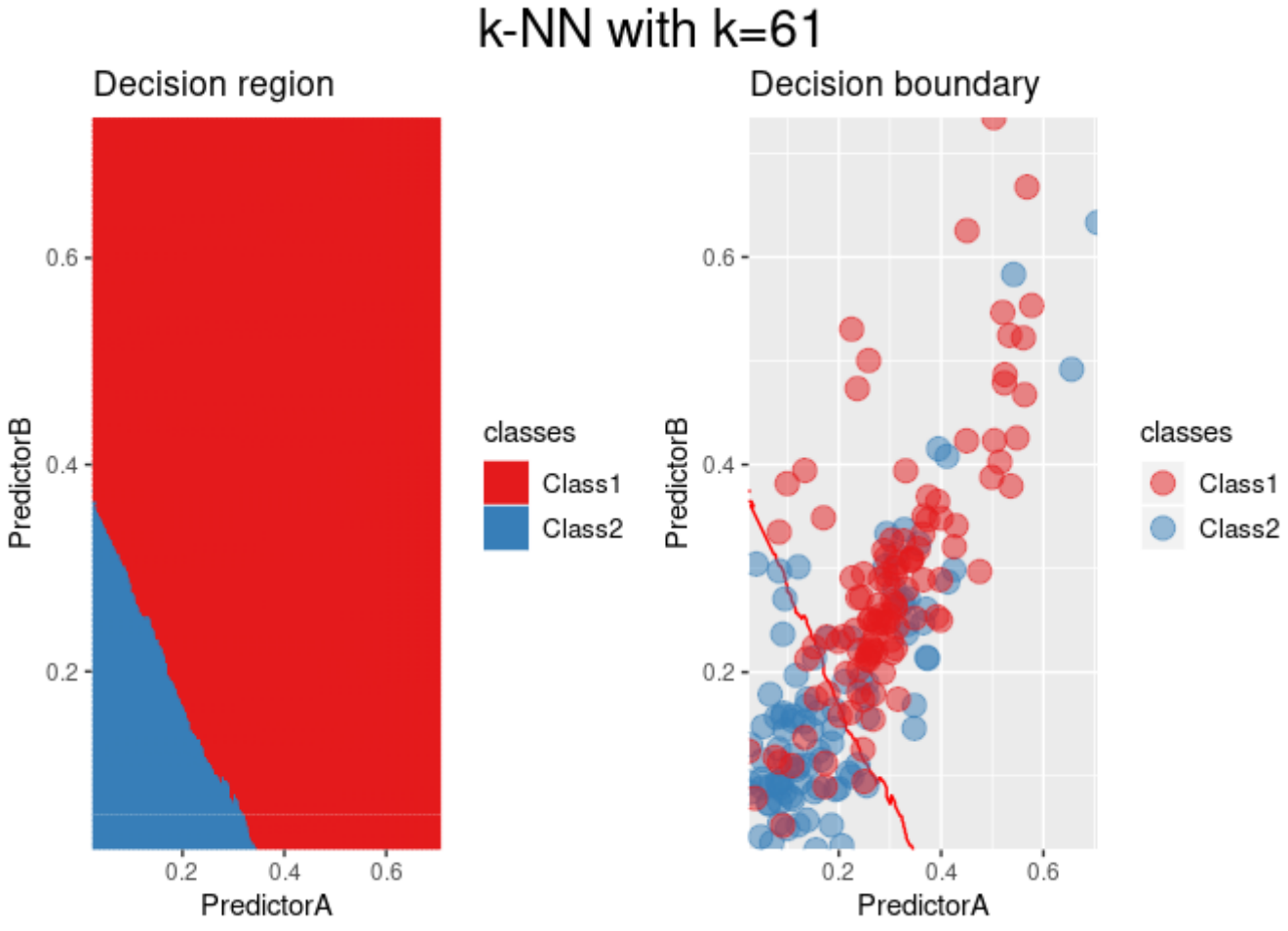
Example: KNN


k-NN with k=45
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

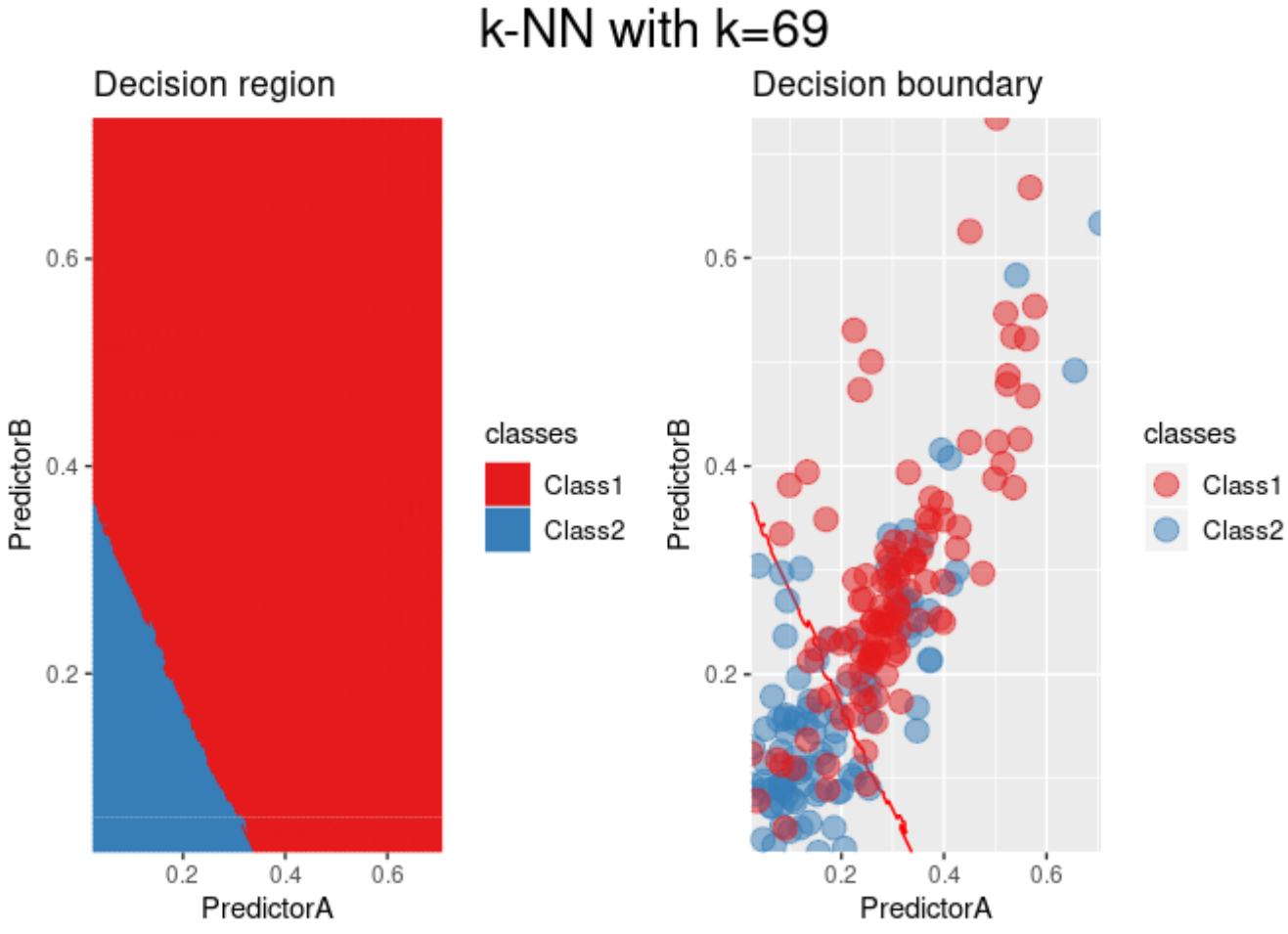
Example: KNN


k-NN with k=53
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

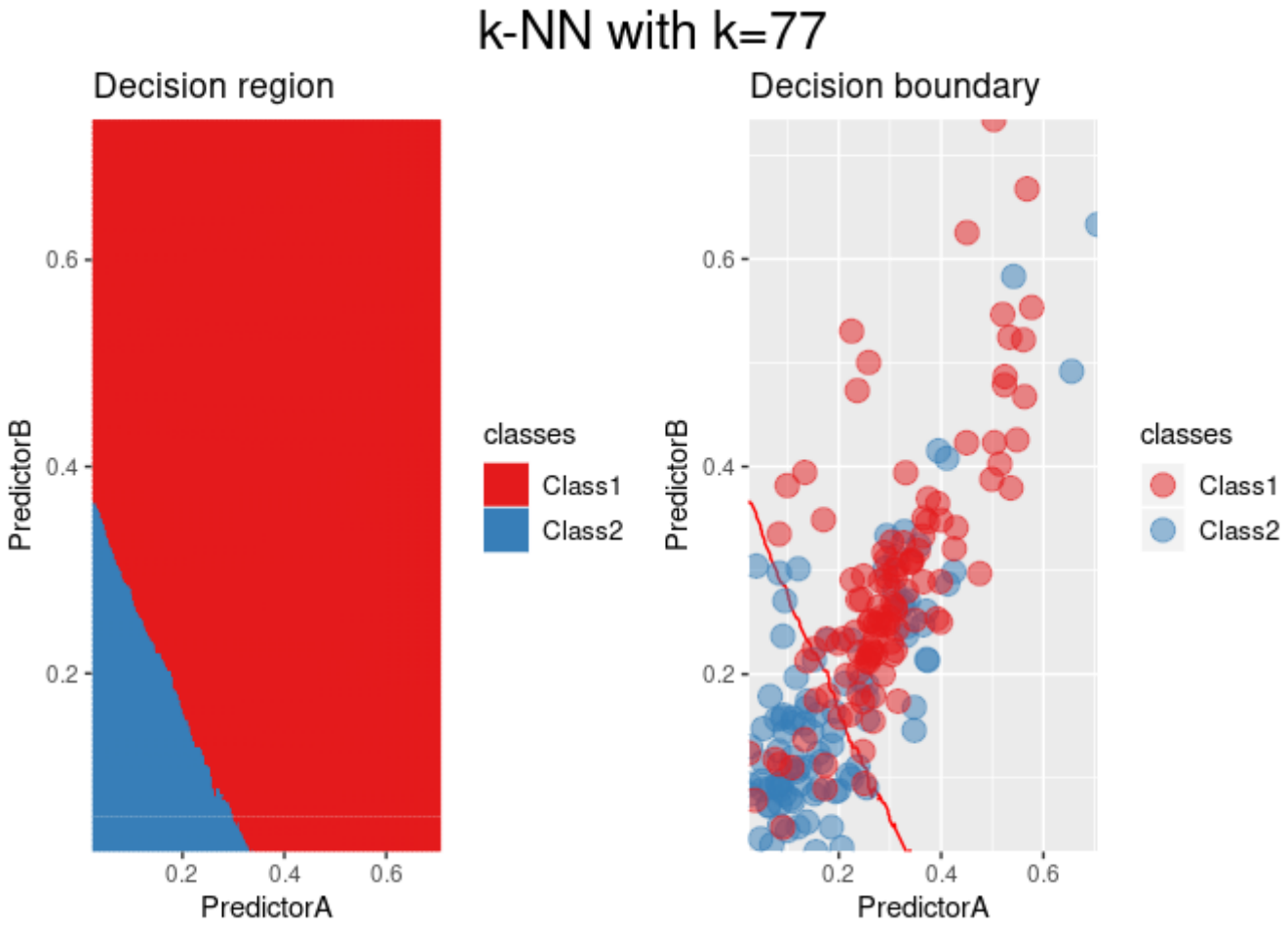
Example: KNN


k-NN with k=61
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

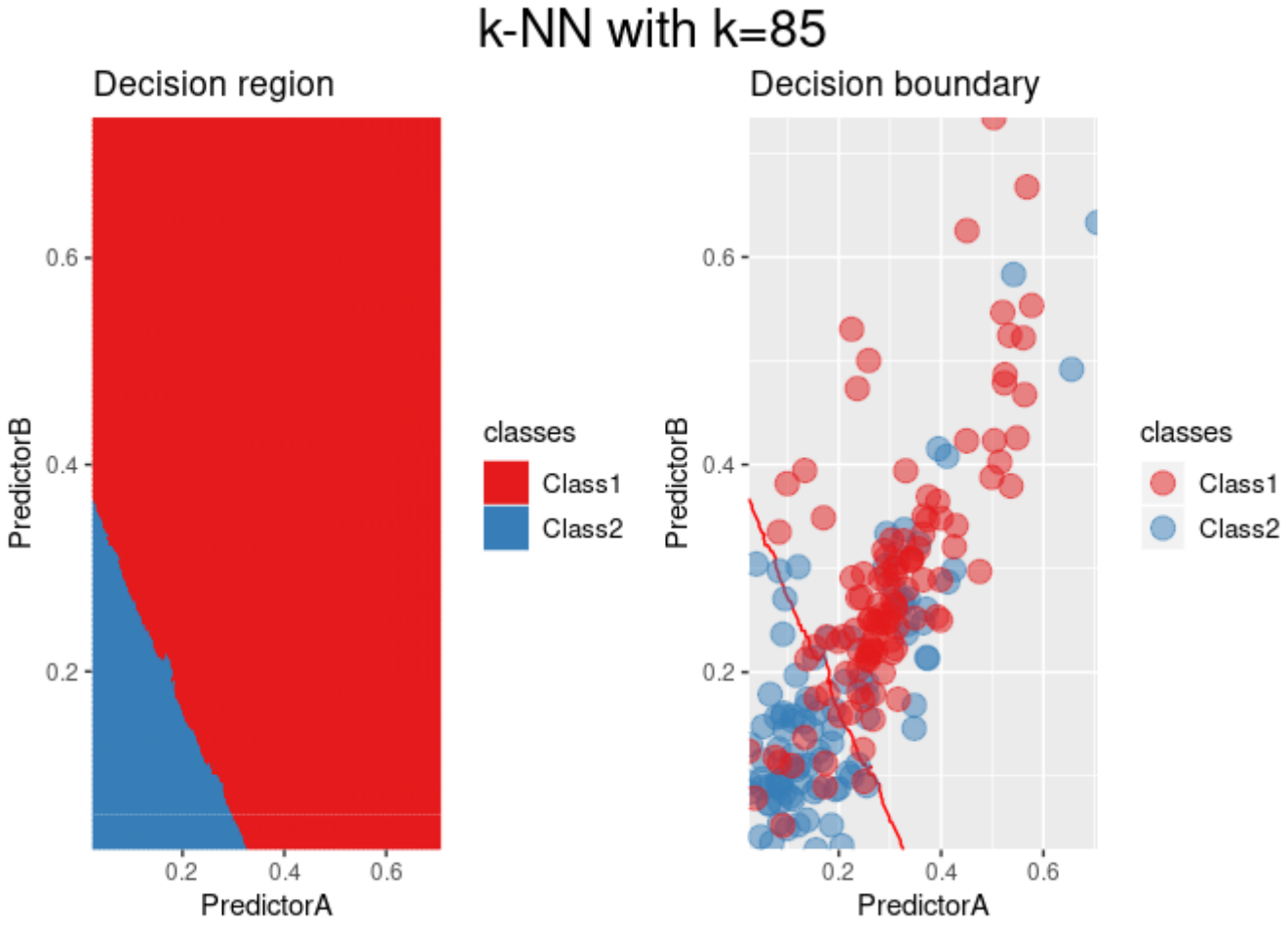
Example: KNN


k-NN with k=69
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

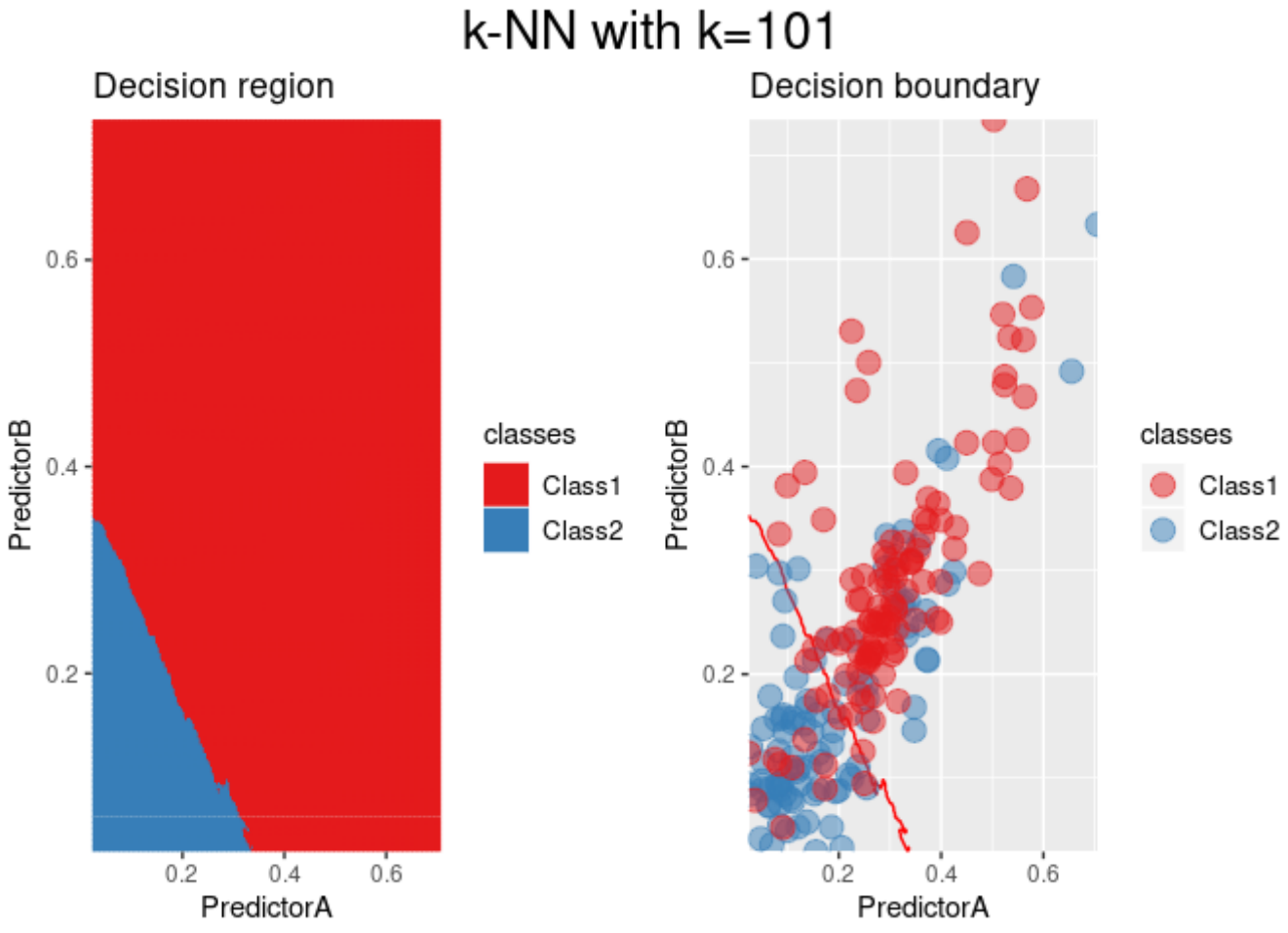
Example: KNN


k-NN with k=77
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

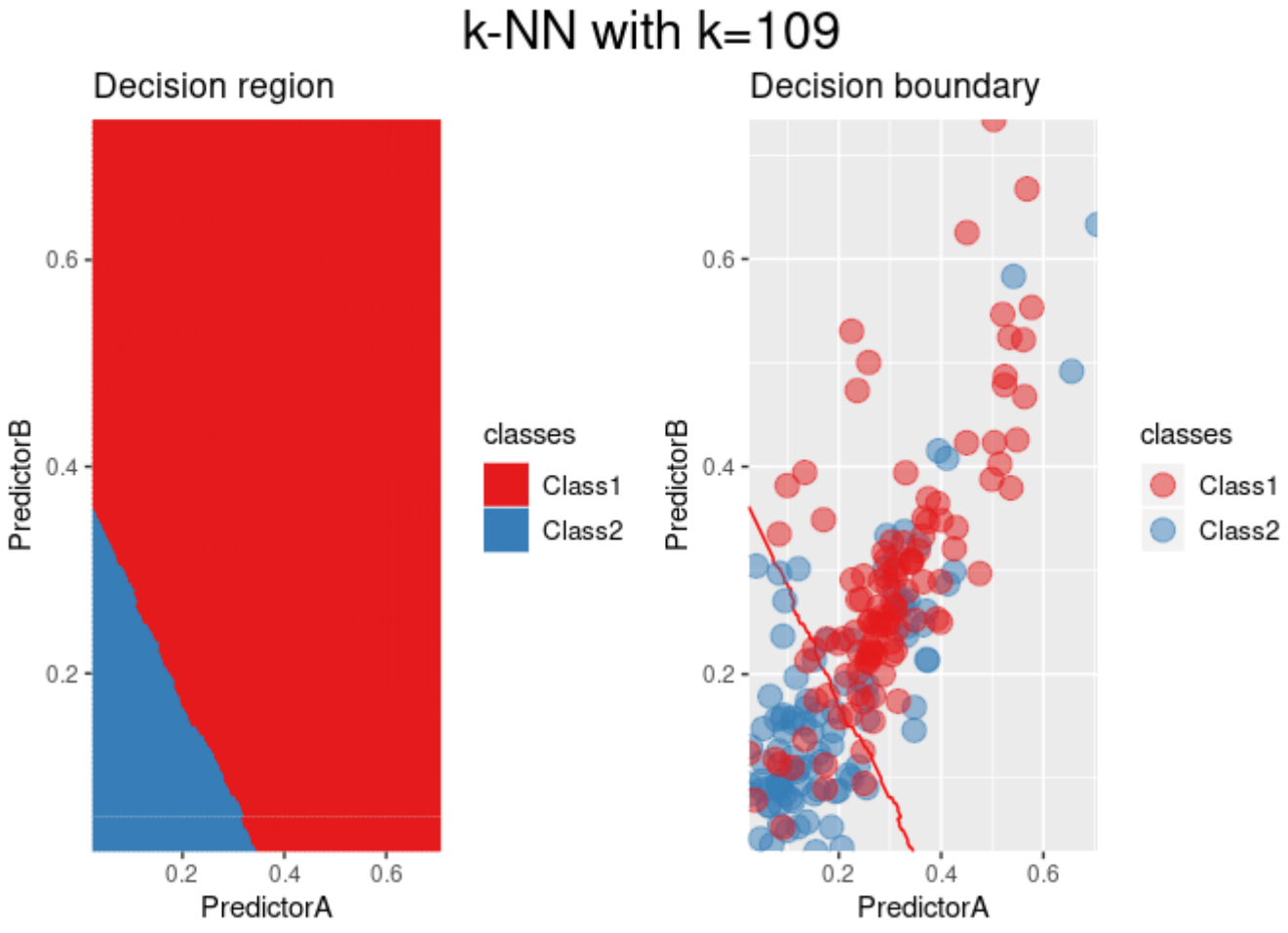
Example: KNN


k-NN with k=85
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

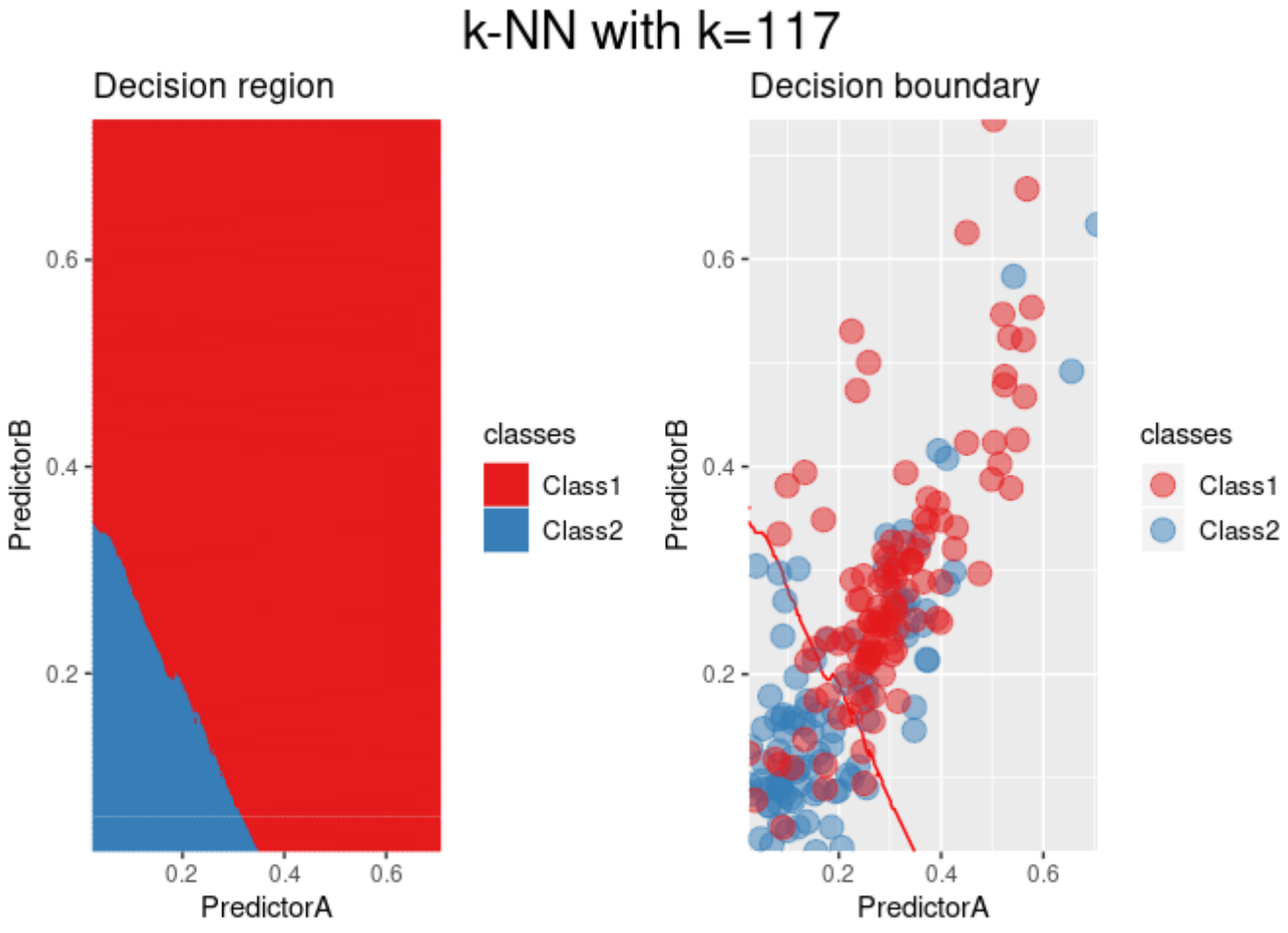
Example: KNN


k-NN with k=101
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

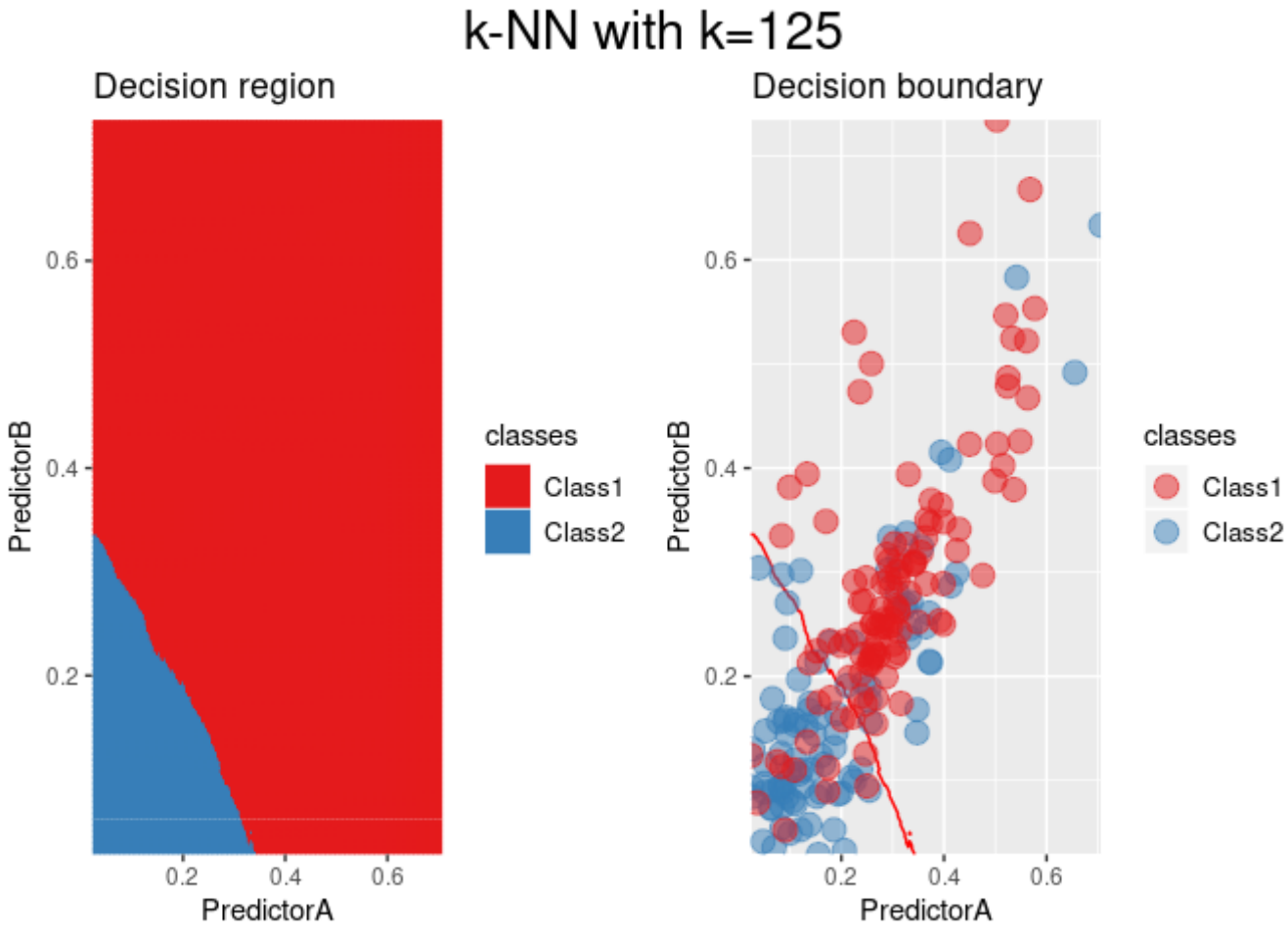
Example: KNN


k-NN with k=109
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

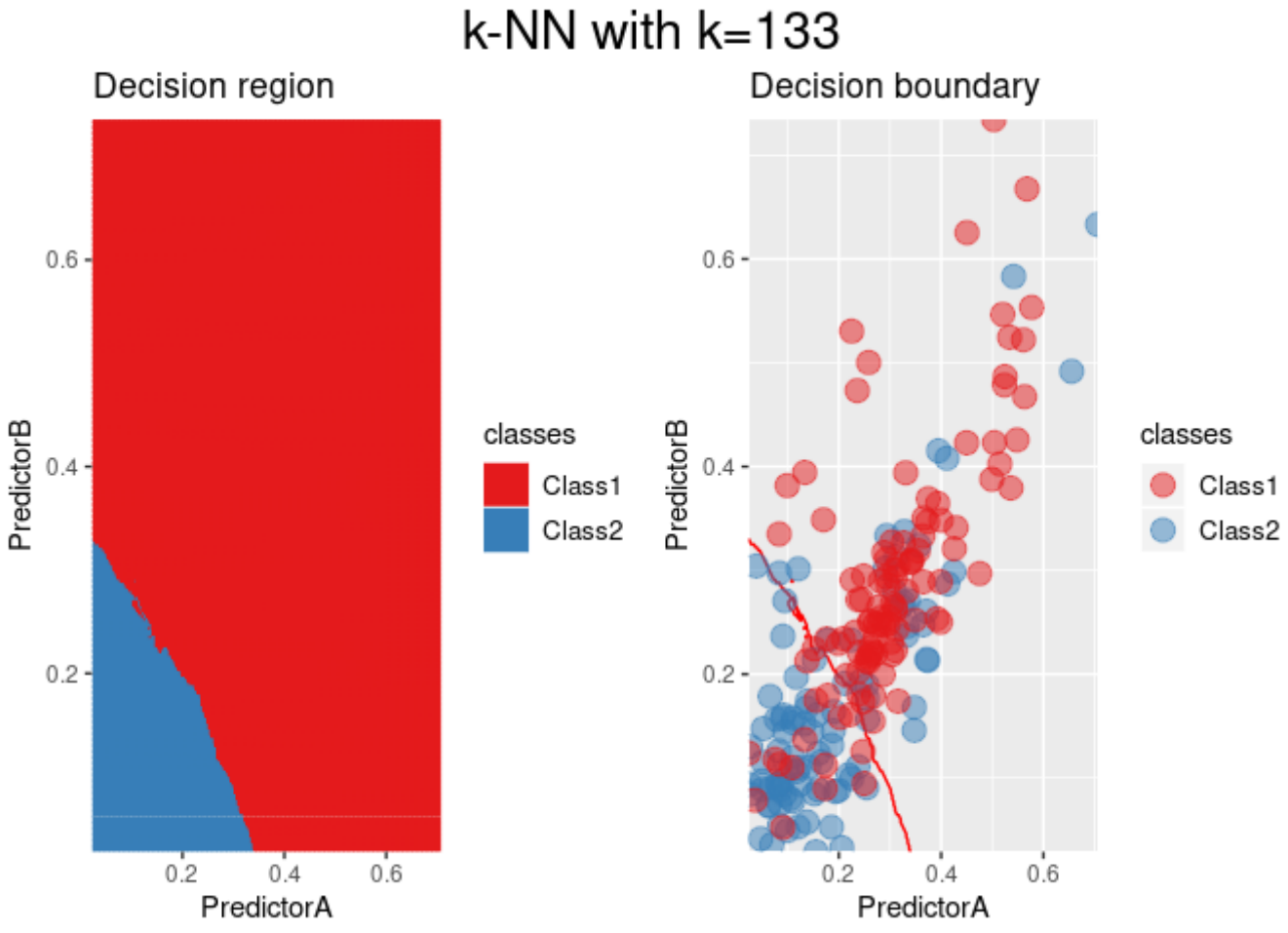
Example: KNN


k-NN with k=117
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

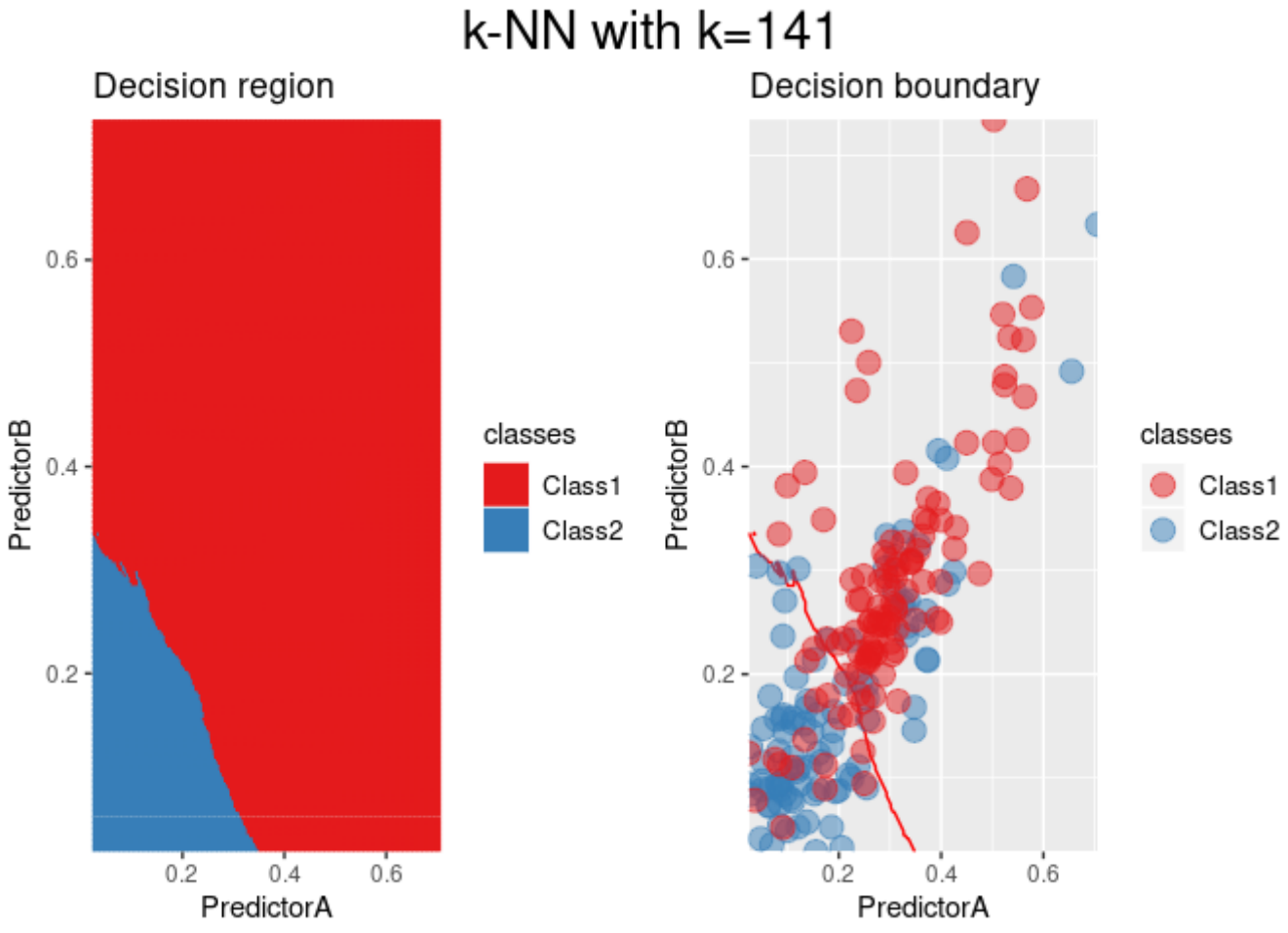
Example: KNN


k-NN with k=125
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

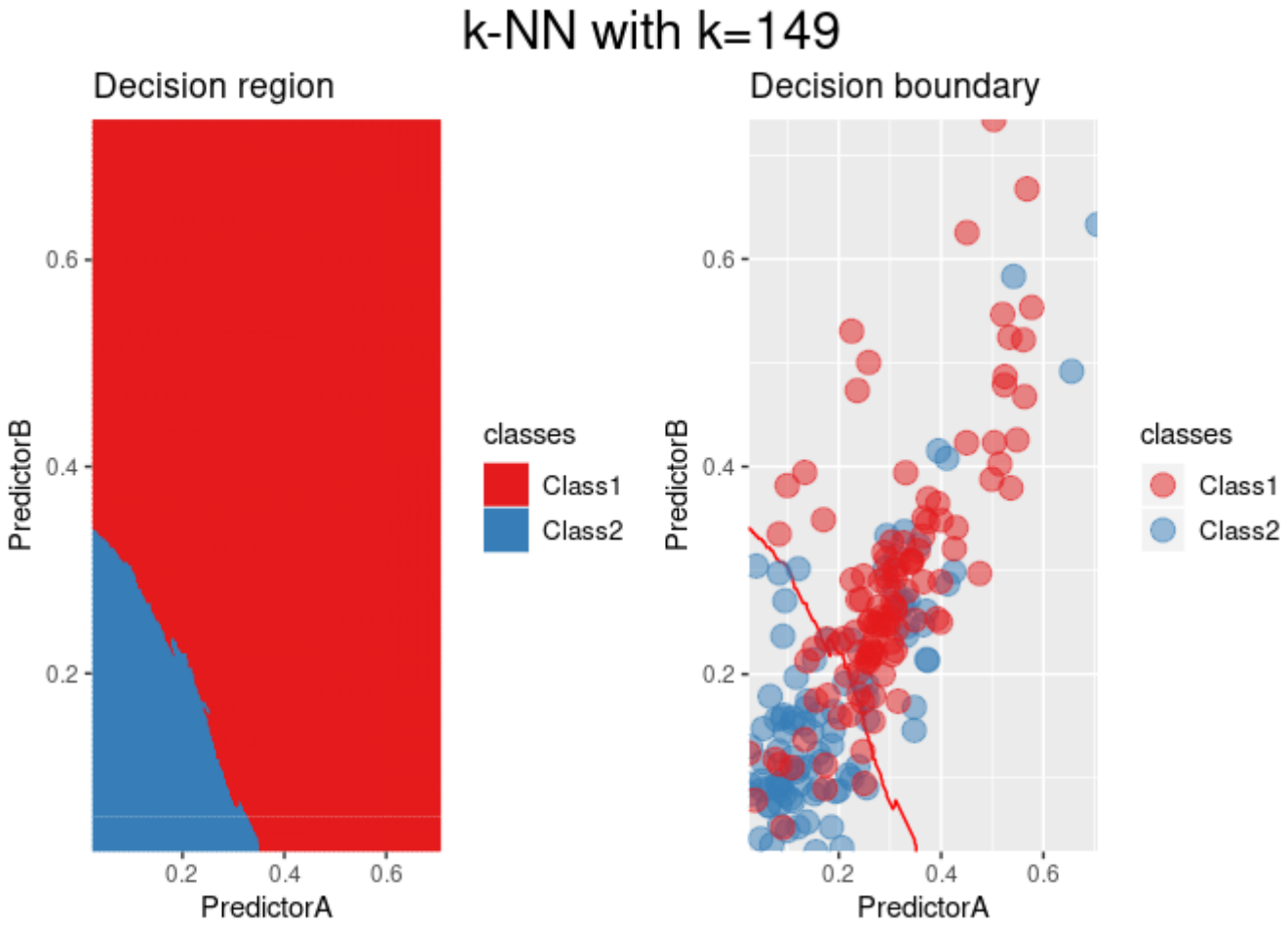
Example: KNN


k-NN with k=133
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

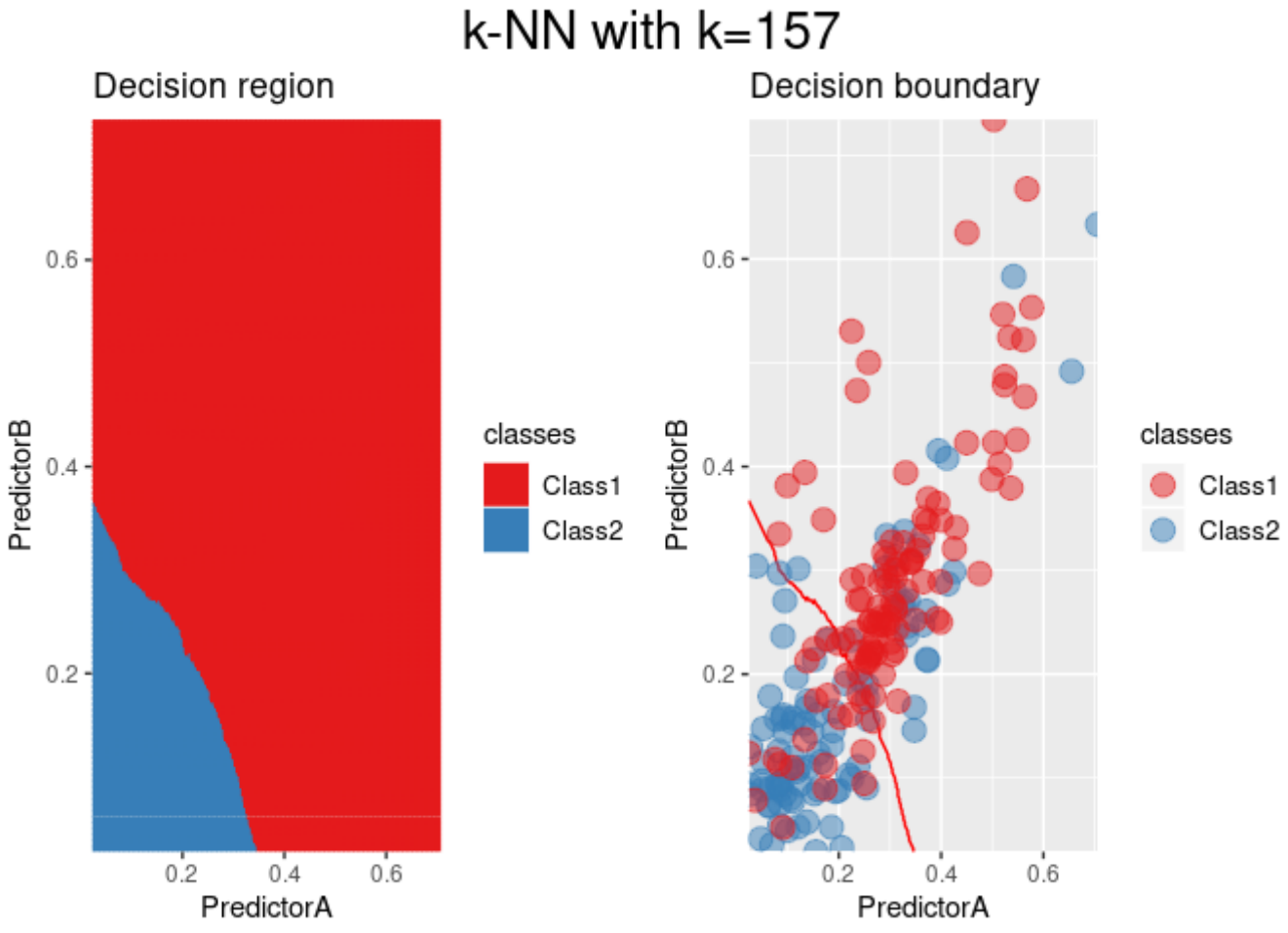
Example: KNN


k-NN with k=141
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

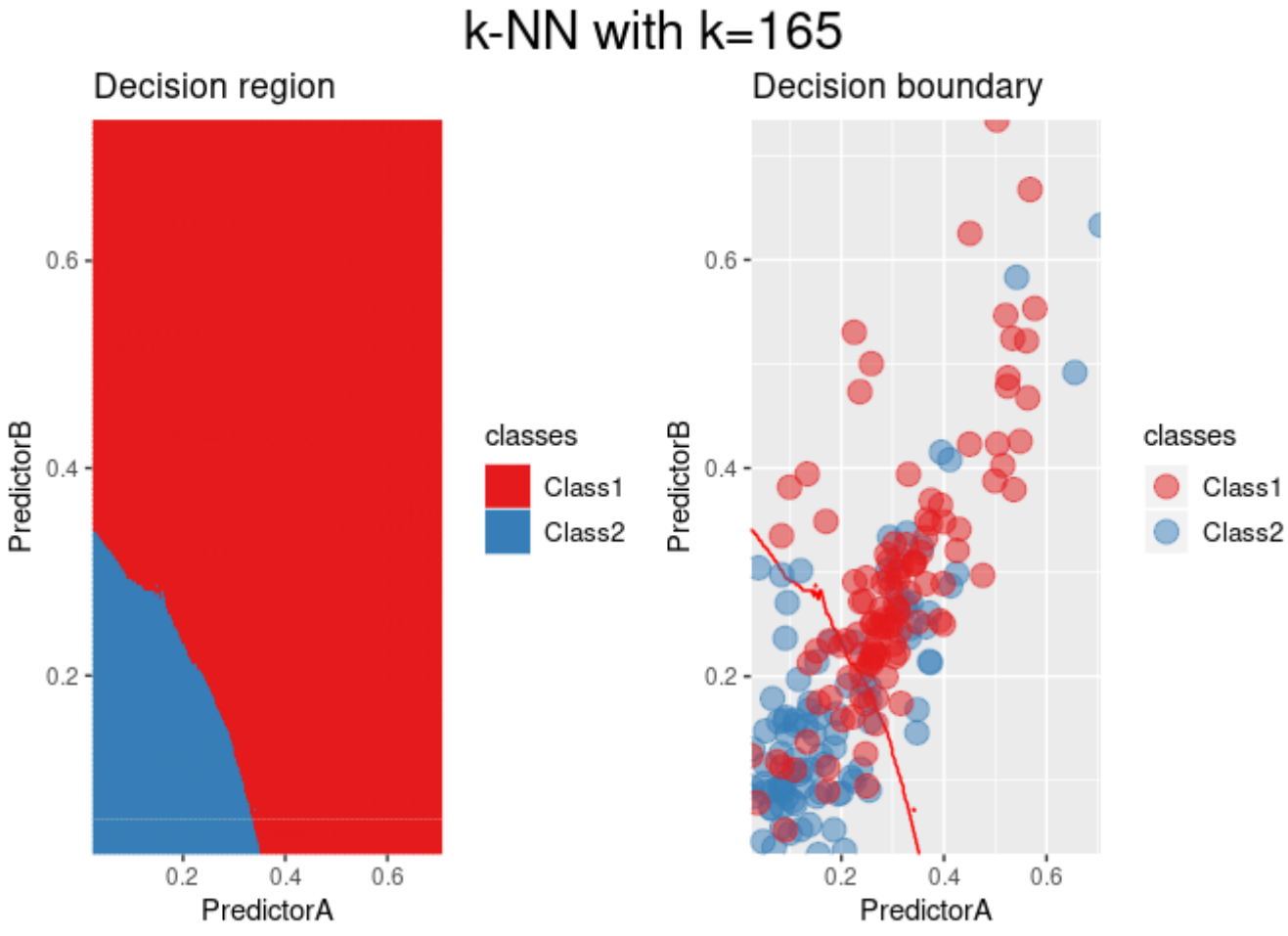
Example: KNN


k-NN with k=149
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

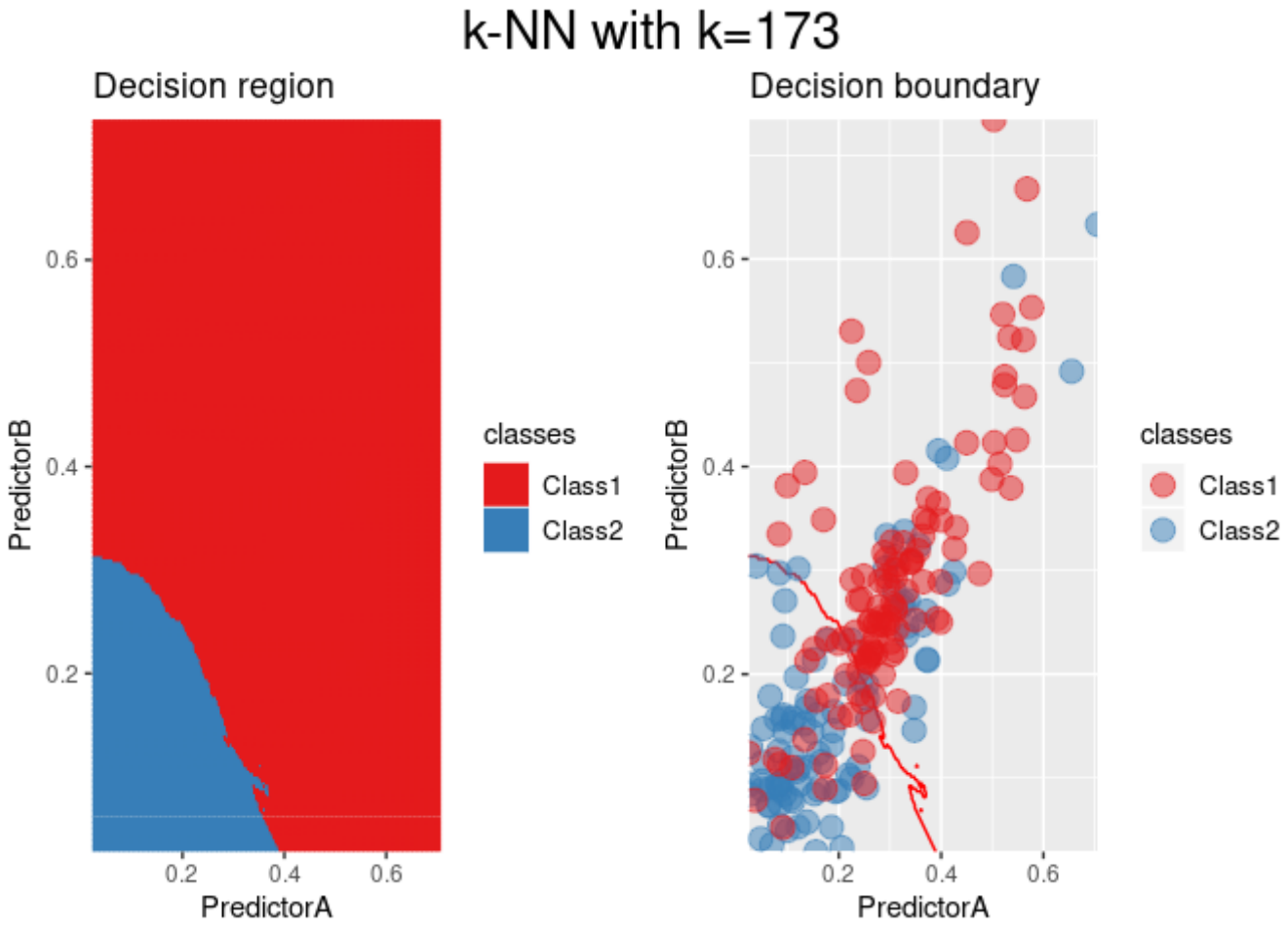
Example: KNN


k-NN with k=157
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

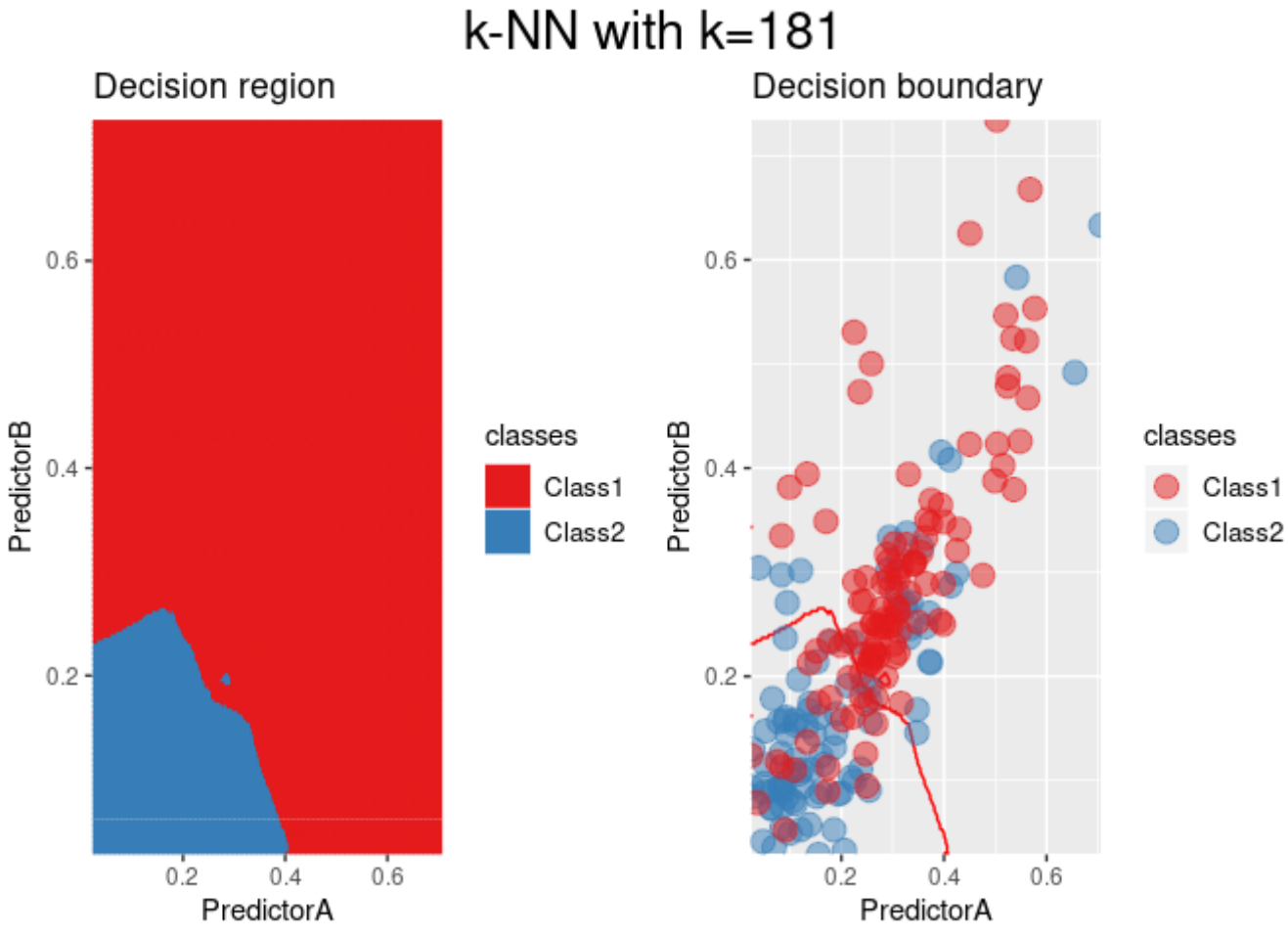
Example: KNN


k-NN with k=165
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

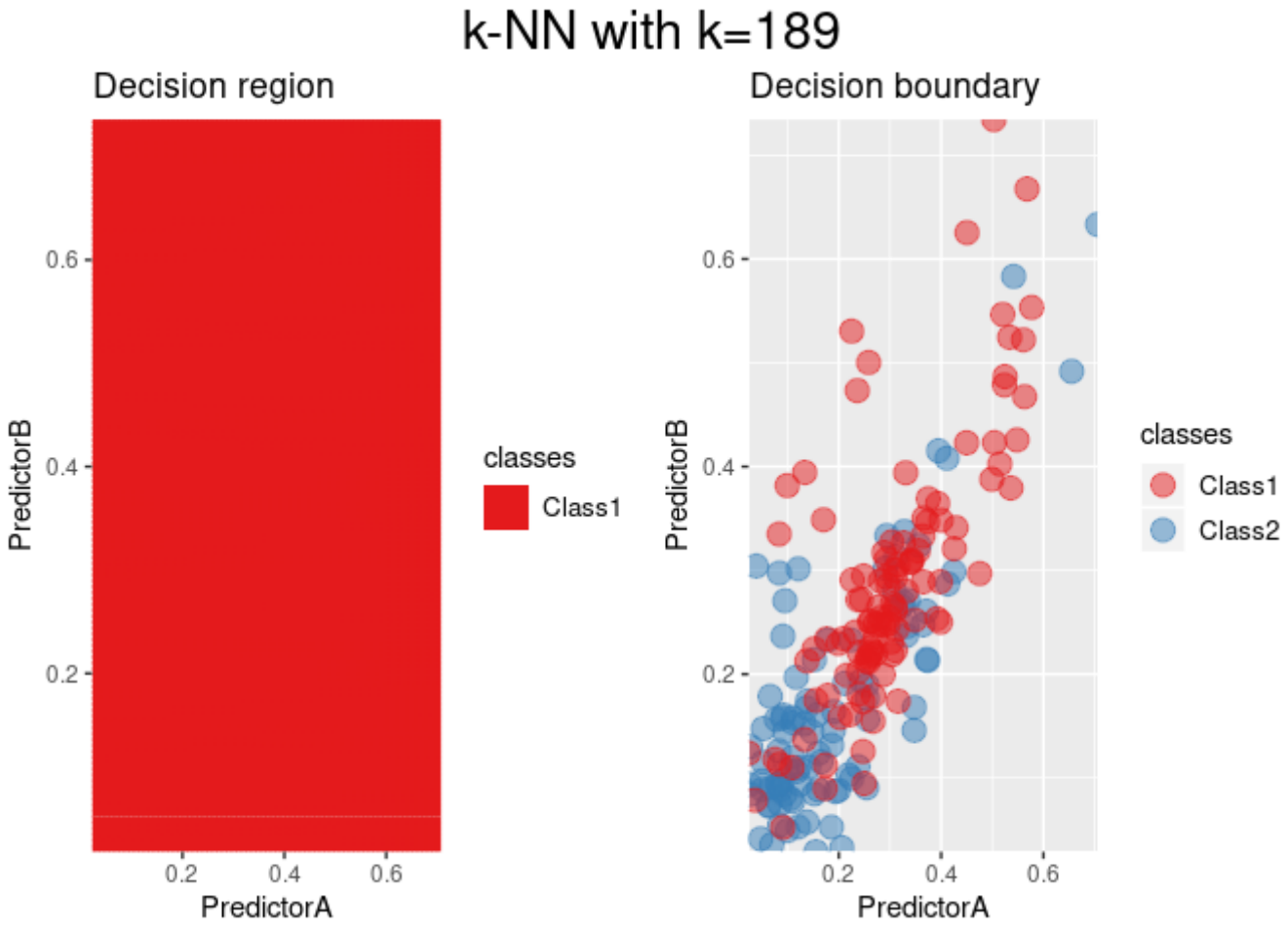
Example: KNN


k-NN with k=173
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

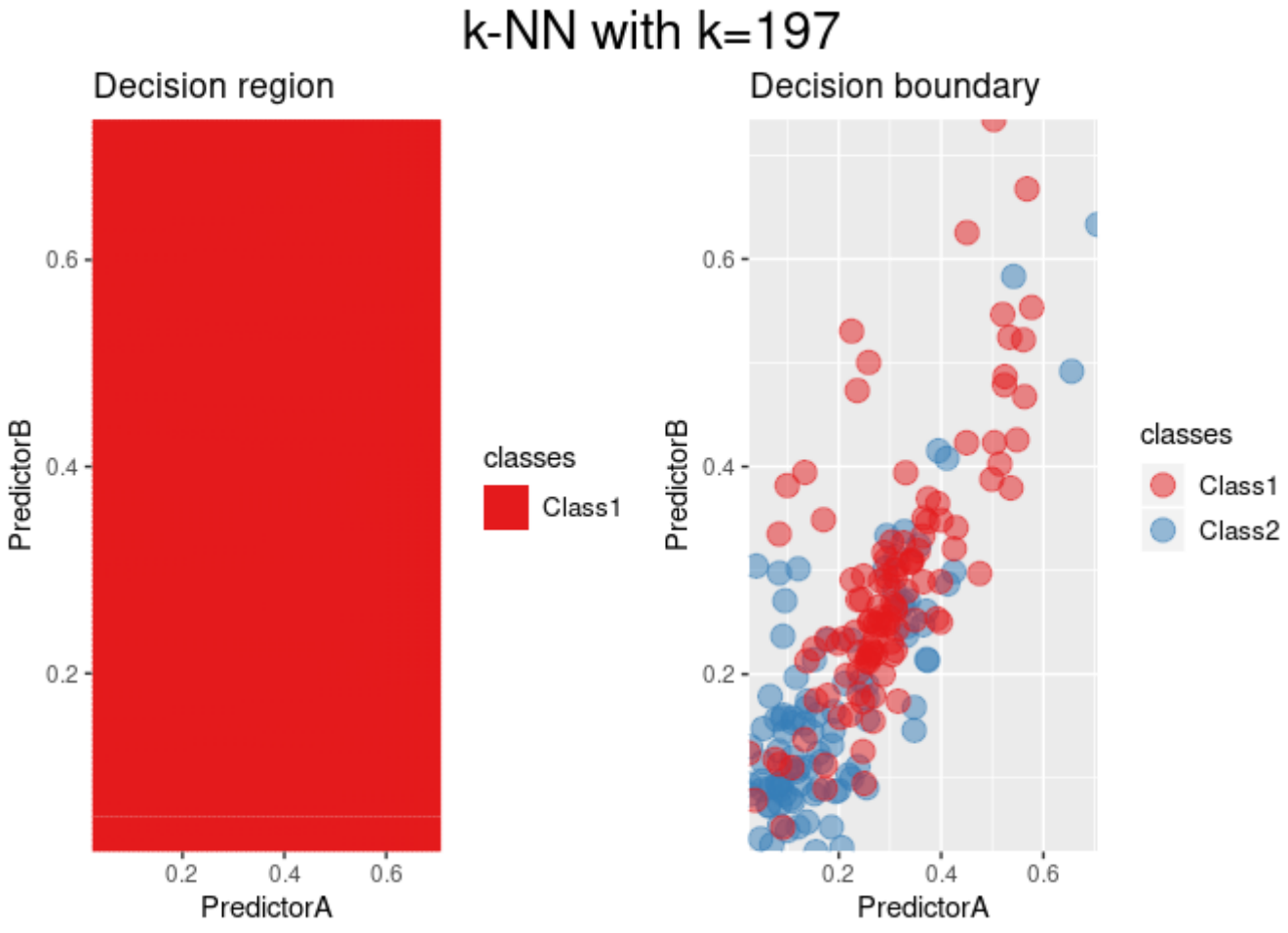
Example: KNN


k-NN with k=181
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: KNN


k-NN with k=189
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: KNN


k-NN with k=197
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Regression and Local Averaging

A naive idea

- $\mathbb{E}[Y|\underline{X}]$ can be approximated by a local average:

$$\widehat{f}(\underline{X}) = \frac{1}{|\{\underline{X}_i \in \mathcal{N}(\underline{X})\}|} \sum_{\underline{X}_i \in \mathcal{N}(\underline{X})} Y_i$$

  where $\mathcal{B}(\underline{X})$ is a neighborhood of $\underline{X}$.

- **Heuristic:**
  - If $\underline{X} \to \mathbb{E}[Y|\underline{X}]$ is regular then
    $\mathbb{E}[Y|\underline{X}] \simeq \mathbb{E}\left[\mathbb{E}\left[Y|\underline{X}'\right]|\underline{X}' \in \mathcal{N}(\underline{X})\right] = \mathbb{E}\left[Y|\underline{X}' \in \mathcal{N}(\underline{X})\right]$
  - Replace an expectation by an empirical average:

$$\mathbb{E}\left[Y|\underline{X}' \in \mathcal{N}(\underline{X})\right] \simeq \frac{1}{|\{\underline{X}_i \in \mathcal{N}(\underline{X})\}|} \sum_{\underline{X}_i \in \mathcal{N}(\underline{X})} Y_i$$

Regression and Local Averaging

Neighborhood and Size

- Most classical choice: $\mathcal{N}(\underline{X}) = \{\underline{X}', \|\underline{X} - \underline{X}'\| \leq h\}$ where $\|.\|$ is a (pseudo) norm and h a size (bandwidth) parameter.
- In principle, the norm and h could vary with $\underline{X}$, and the norm can be replaced by a (pseudo) distance.
- Focus here on a fixed distance with a fixed bandwidth h cased.

Bandwidth Heuristic

- A **large bandwidth** ensures that the average is taken on many samples and thus the **variance is small**...
- **A small bandwidth** is thus that the approximation $\mathbb{E}[Y|\underline{X}] \simeq \mathbb{E}\left[Y|\underline{X}' \in \mathcal{N}(\underline{X})\right]$ is more accurate **(small bias)**.

Weighted Local Averaging

Weighted Local Average

- Replace the neighborhood $\mathcal{N}(\underline{X})$ by a decaying **window function** $w(\underline{X}, \underline{X}')$.
- $\mathbb{E}[Y|\underline{X}]$ can be approximated by a **weighted local average**:

$$\widehat{f}(\underline{X}) = \frac{\sum_i w(\underline{X}, \underline{X}'_i) Y_i}{\sum_i w(\underline{X}, \underline{X}'_i)}.$$

Kernel

- Most classical choice: $w(\underline{X}, \underline{X}') = K\left(\frac{\underline{X}-\underline{X}'}{h}\right)$ where h the bandwidth is a scale parameter.
- Examples:
  - **Box kernel:** $K(t) = \mathbf{1}_{\|t\| \leq 1}$ (Neighborhood)
  - **Triangular kernel:** $K(t) = \max(1 - \|t\|, 0)$.
  - **Gaussian kernel:** $K(t) = e^{-t^2/2}$
- **Rk:** K and λK yields the same estimate.

Nadaraya-Watson Heuristic

- Provided all the **densities** exist

$$\mathbb{E}\left[Y|\underline{X}\right] = \frac{\int Yp(\underline{X}, Y)dY}{\int p(Y, \underline{X})dY} = \frac{\int Yp(\underline{X}, Y)dY}{p(\underline{X})}$$

- Replace the unknown densities by their **estimates**:

$$\widehat{p}(\underline{X}) = \frac{1}{n}\sum_{i=1}^{n} K(\underline{X} - \underline{X}_i)$$

$$\widehat{p}(\underline{X}, Y) = \frac{1}{n}\sum_{i=1}^{n} K(\underline{X} - \underline{X}_i)K'(Y - Y_i)$$

- Now if K' is a kernel such that $\int YK'(Y)dY = 0$ then

$$\int Y\widehat{p}(\underline{X}, Y)dY = \frac{1}{n}\sum_{i=1}^{n} K(\underline{X} - \underline{X}_i)Y_i$$

From Density Estimation to Regression

Nadaraya-Watson

- Resulting estimator of $\mathbb{E}[Y|\underline{X}]$

$$\widehat{f}(\underline{X}) = \frac{\sum_{i=1}^{n} Y_i K_h(\underline{X} - \underline{X}_i)}{\sum_{i=1}^{n} K_h(\underline{X} - \underline{X}_i)}$$

- Same **local weighted average** estimator!

Bandwidth Choice

- Bandwidth h of K allows to **balance between bias and variance**.
- Theoretical analysis of the error is possible.
- The smoother the densities the easier the estimation but the optimal bandwidth depends on the unknown regularity!

Local Linear Estimation

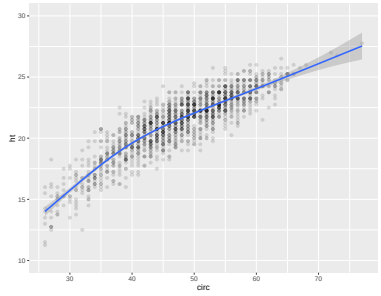
Another Point of View on Kernel

- Nadaraya-Watson estimator:
$$\widehat{f}(\underline{X}) = \frac{\sum_{i=1}^n Y_i K_h(\underline{X} - \underline{X}_i)}{\sum_{i=1}^n K_h(\underline{X} - \underline{X}_i)}$$
- Can be view as a **minimizer** of
$$\sum_{i=1}^n |Y_i - \beta|^2 K_h(\underline{X} - \underline{X}_i)$$
- **Local regression** of order 0!

Local Linear Model

- Estimate $\mathbb{E}[Y|\underline{X}]$ by $\widehat{f}(\underline{X}) = \phi(\underline{X})^\top \widehat{\beta}(\underline{X})$ where ϕ is any function of $\underline{X}$ and $\widehat{\beta}(\underline{X})$ is the minimizer of
$$\sum_{i=1}^n |Y_i - \phi(\underline{X}_i)^\top \beta|^2 K_h(\underline{X} - \underline{X}_i).$$

LOESS: LOcal polynomial regrESSion



1D Nonparametric Regression

- Assume that $\underline{X} \in \mathbb{R}$ and let $\phi(\underline{X}) = (1, \underline{X}, \ldots, \underline{X}^d)$.
- **LOESS estimate:** $\widehat{f}(\underline{X}) = \sum_{j=0}^{d} \widehat{\beta}(\underline{X}^{(j)})\underline{X}^j$ with $\widehat{\beta}(\underline{X})$ minimizing

$$\sum_{i=1}^{n} |Y_i - \sum_{j=0}^{d} \beta^{(j)} \underline{X}_i^j|^2 K_h(\underline{X} - \underline{X}_i).$$

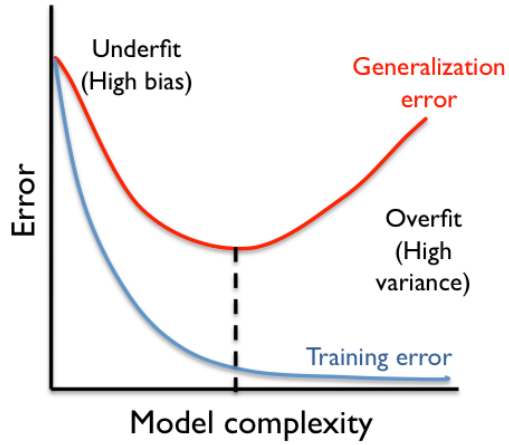
- Most classical kernel used: Tricubic kernel

$$K(t) = \max(1 - |t|^3, 0)^3$$

- Most classical degree: 2...
- Local bandwidth choice such that a proportion of points belongs to the window.

Outline

Training Error Issue



Error behaviour

- Learning/training error (error made on the learning/training set) decays when the complexity of the **method** increases.
- Quite different behavior when the error is computed on new observations (generalization error).

- Overfit for complex methods: parameters learned are too specific to the learning set!
- General situation! (Think of polynomial fit...)
- Need to use a different criterion than the training error!

Source: JMP

Error Estimation vs Method Selection

Predictor Error Estimation

- **Goal:** Given a predictor f assess its quality.
- **Method:** Hold-out error computation (/ Error correction).
- **Usage:** Compute an estimate of the error of a selected f using a **test set** to be used to monitor it in the future.

- Basic block very well understood.

Method Selection

- **Goal:** Given a ML method assess its quality.
- **Method:** Cross Validation (/ Error correction)
- **Usage:** Compute error estimates for several ML methods using **training/validation sets** to choose the most promising one.

- Estimates can be pointwise or better intervals.
- Multiple test issues in method selection.

Cross Validation and Error Correction

Two Approaches

- **Cross validation:** Very efficient (and almost always used in practice!) but slightly biased as it target uses only a fraction of the data.
- **Correction approach:** use empirical loss criterion but *correct* it with a term increasing with the complexity of $\mathcal{S}$

$$R_n(\widehat{f}_{\mathcal{S}}) \to R_n(\widehat{f}_{\mathcal{S}}) + \text{cor}(\mathcal{S})$$

and choose the method with the smallest corrected risk.

Which loss to use?

- The loss used in the risk: most natural!
- The loss used to estimate $\widehat{\theta}$: penalized estimation!

Cross Validation

- **Very simple idea:** use a second learning/verification set to compute a verification error.
- Sufficient to remove the dependency issue!
- Implicit random design setting...

Cross Validation

- Use $(1-\epsilon) \times n$ observations to train and $\epsilon \times n$ to verify!
- Possible issues:
  - Validation for a learning set of size $(1-\epsilon) \times n$ instead of n ?
  - Unstable error estimate if ϵn is too small ?

- Most classical variations:
  - Hold Out,
  - Leave One Out,
  - V-fold cross validation.

Source: JMP

Hold Out

Principle

- Split the dataset $\mathcal{D}$ in 2 sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ of size $n \times (1-\epsilon)$ and $n \times \epsilon$.
- Learn $\widehat{f}^{HO}$ from the subset $\mathcal{D}_{\text{train}}$.
- Compute the empirical error on the subset $\mathcal{D}_{\text{test}}$:

$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} \ell(Y_i, \widehat{f}^{HO}(\underline{X}_i))$$

Predictor Error Estimation

- Use $\hat{f}^{HO}$ as predictor.
- Use $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ as an estimate of the error of this estimator.

Method Selection by Cross Validation

- Compute $\mathcal{R}_n^{HO}(\widehat{f}_{\mathcal{S}}^{HO})$ for all the considered methods,
- Select the method with the smallest CV error,
- Reestimate the $\widehat{f}_{\mathcal{S}}$ with all the data.

Hold Out

Principle

- Split the dataset $\mathcal{D}$ in 2 sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ of size $n \times (1-\epsilon)$ and $n \times \epsilon$.
- Learn $\widehat{f}^{HO}$ from the subset $\mathcal{D}_{\text{train}}$.
- Compute the empirical error on the subset $\mathcal{D}_{\text{test}}$:

$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} \ell(Y_i, \widehat{f}^{HO}(\underline{X}_i))$$

- Only possible setting for error estimation.

Hold Out Limitation for Method Selection

- Biased toward simpler method as the estimation does not use all the data initially.
- Learning variability of $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ not taken into account.

V-fold Cross Validation

Principle

- Split the dataset $\mathcal{D}$ in V sets $\mathcal{D}_v$ of almost equals size.
- For $v \in \{1, .., V\}$:
  - Learn $\widehat{f}^{-v}$ from the dataset $\mathcal{D}$ minus the set $\mathcal{D}_v$.
  - Compute the empirical error:

$$\mathcal{R}_n^{-v}(\widehat{f}^{-v}) = \frac{1}{n_v} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_v} \ell(Y_i, \widehat{f}^{-v}(\underline{X}_i))$$

- Compute the average empirical error:

$$\mathcal{R}_n^{CV}(\widehat{f}) = \frac{1}{V} \sum_{v=1}^{V} \mathcal{R}_n^{-v}(\widehat{f}^{-v})$$

- Estimation of the quality of method not of a given predictor.
- Leave One Out : V = n.

Source: JMP

V-fold Cross Validation

Analysis (when n is a multiple of V)

- The $\mathcal{R}_n^{-v}(\widehat{f}^{-v})$ are identically distributed variable but are not independent!
- Consequence:

$$\mathbb{E}\left[\mathcal{R}_n^{CV}(\widehat{f})\right] = \mathbb{E}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v})\right]$$

$$\mathbb{V}\text{ar}\left[\mathcal{R}_n^{CV}(\widehat{f})\right] = \frac{1}{V}\mathbb{V}\text{ar}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v})\right] + (1-\frac{1}{V})\,\mathbb{C}\text{ov}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v}), \mathcal{R}_n^{-v'}(\widehat{f}^{-v'})\right]$$

- Average risk for a sample of size $(1-\frac{1}{V})n$.
- Variance term much more complex to analyze!
- Fine analysis shows that the larger V the better...

- Accuracy/Speed tradeoff: $V = 5$ or $V = 10$!

Linear Regression and Leave One Out

- Leave One Out = V fold for $V = n$: very expensive in general.

A fast LOO formula for the linear regression

- **Prop:** for the least squares linear regression,

$$\widehat{f}^{-i}(\underline{X}_i) = \frac{\widehat{f}(\underline{X}_i) - h_{ii} Y_i}{1 - h_{ii}}$$

  with h_{ii} the ith diagonal coefficient of the **hat** (projection) matrix.
- Proof based on linear algebra!
- Leads to a fast formula for LOO:

$$\mathcal{R}_n^{LOO}(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i - \widehat{f}(\underline{X}_i)|^2}{(1 - h_{ii})^2}$$

Cross Validation and Confidence Interval

- How to replace pointwise estimation by a confidence interval?
- Can we use the variability of the CV estimates?
- **Negative result:** No unbiased estimate of the variance!

Gaussian Interval (Comparison of the means and $\sim$ indep.)

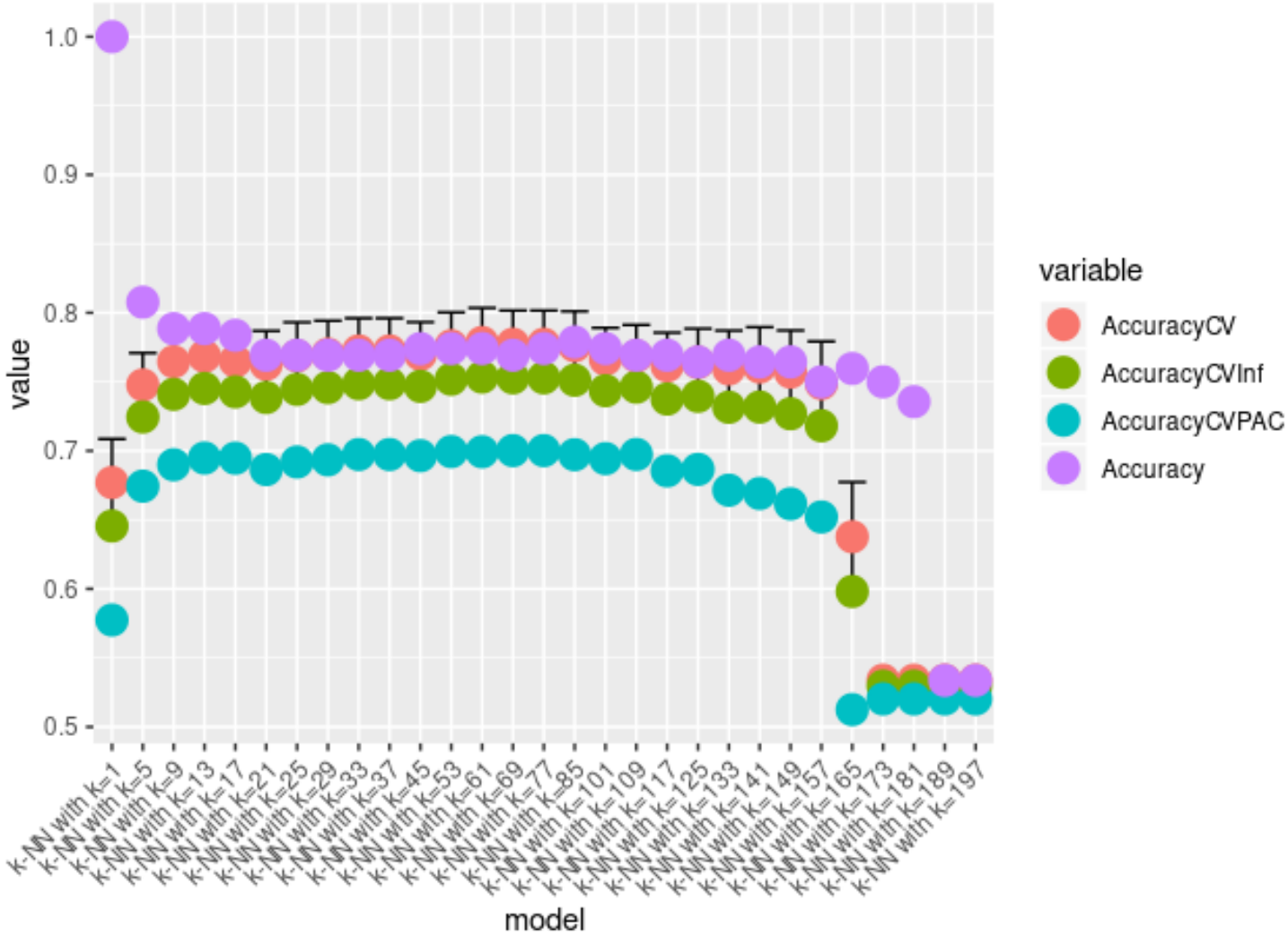
- Compute the empirical variance and divide it by the number of folds to construct an asymptotic Gaussian confidence interval,
- Select the simplest model whose values falls into the confidence interval of the model having the smallest CV error.

PAC approach (Quantile, $\sim$ indep. and small risk estim. error)

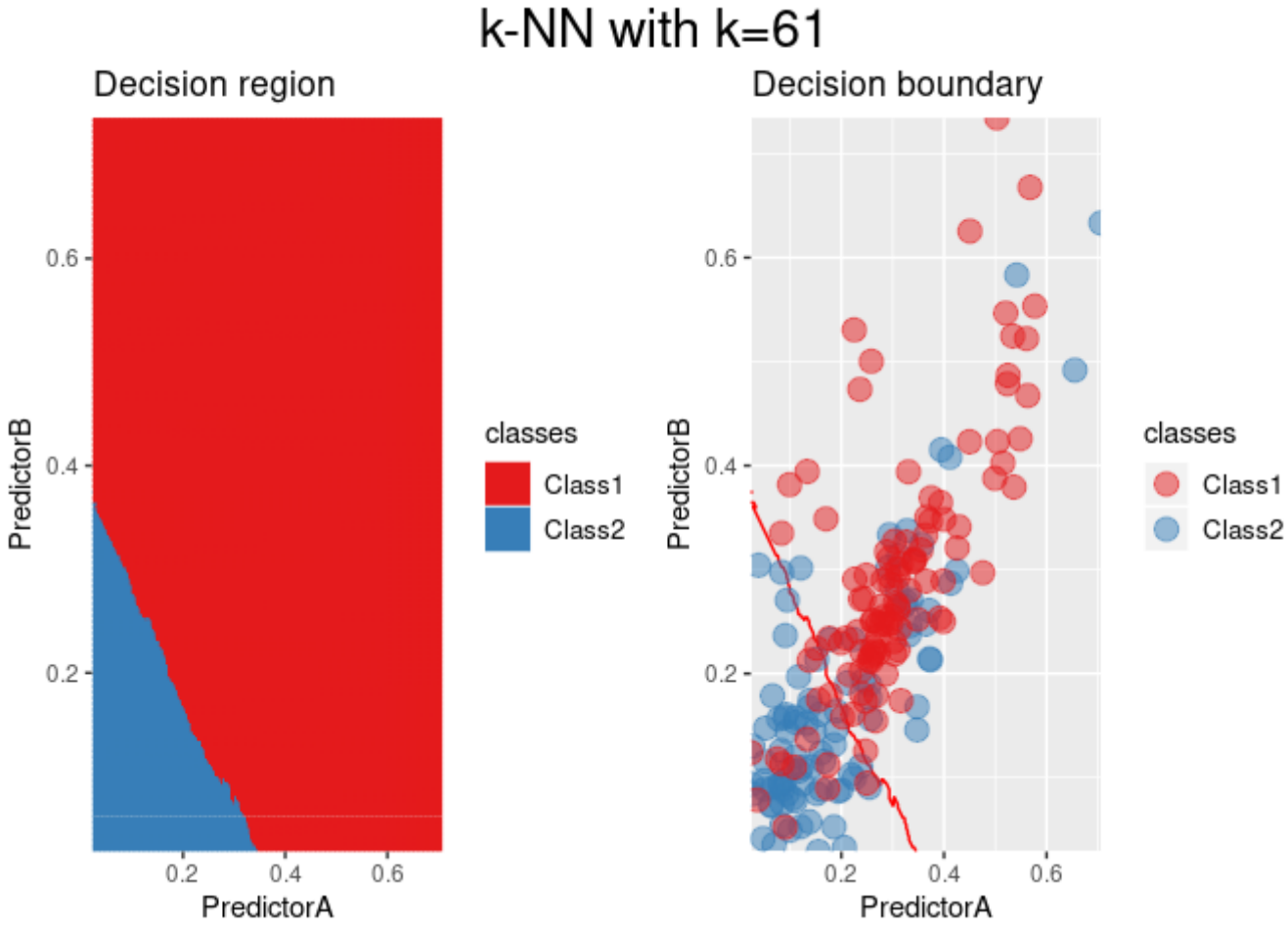
- Compute the raw medians (or a larger raw quantiles)
- Select the model having the smallest quantiles to ensure a small risk with high probability.

- Always reestimate the chosen model with all the data.
- To obtain an unbiased error estimate of the final predictor: hold out error on untouched test data.

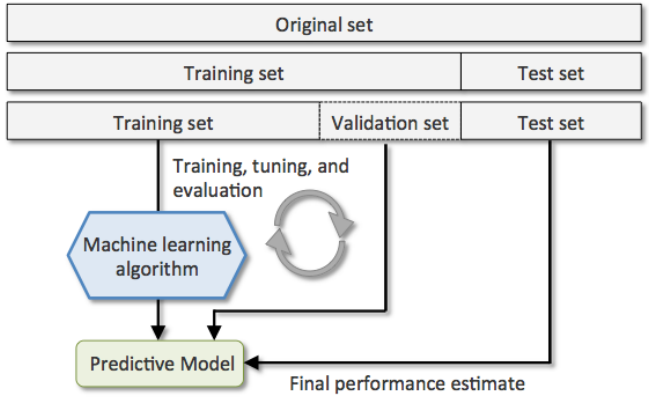
Cross Validation



Example: KNN ($\widehat{k} = 61$ using cross-validation)



Train/Validation/Test



- **Selection Bias Issue:**
  - After method selection, the cross validation is biased.
  - Furthermore, it qualifies the method and not the final predictor.
- Need to (re)estimate the error of the final predictor.

(Train/Validation)/Test strategy

- **Split** the dataset in two a `(Train/Validation)` and `Test`.
- Use **CV** with the `(Train/Validation)` to **select a method**.
- Train this method on `(Train/Validation)` to **obtain a single predictor**.
- Estimate the **performance of this predictor** on `Test`.

Source: Shan-Hung Wu & DataLab

Error Correction

- Empirical loss of an estimator computed on the dataset used to chose is is biased!
- Empirical loss is an optimistic estimate of the true loss.

Risk Correction Heuristic

- Estimate an upper bound of this optimism for a given family.
- Correct the empirical loss by adding this upper bound.

- **Rk:** Finding such an upper bound can be complicated!
- Correction often called a **penalty**.

Penalization

Penalized Loss

- Minimization of

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(X_i)) + \text{pen}(\theta)$$

where $\text{pen}(\theta)$ is an error correction (penalty).

Penalties

- Upper bound of the optimism of the empirical loss
- Depends on the loss and the framework!

Instantiation

- Mallows Cp: Least Squares with $\text{pen}(\theta) = 2\frac{d}{n}\sigma^2$.
- AIC Heuristics: Maximum Likelihood with $\text{pen}(\theta) = \frac{d}{n}$.
- BIC Heuristics: Maximum Likelohood with $\text{pen}(\theta) = \log(n)\frac{d}{n}$.
- Structural Risk Minimization: Pred. loss and clever penalty.

Outline

Probabilistic and Optimization Framework

How to find a good function f with a *small* risk

$$R(f) = \mathbb{E}\left[\ell(Y, f(\underline{X}))\right] \quad ?$$

Canonical approach: $\widehat{f}_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\underline{X}_i))$

Problems

- How to choose $\mathcal{S}$?
- How to compute the minimization?

A Probabilistic Point of View

Solution: For $\underline{X}$, estimate $Y|\underline{X}$ plug this estimate in the Bayes classifier: **(Generalized) Linear Models, Kernel methods, k-nn, Naive Bayes, Tree, Bagging...**

An Optimization Point of View

Solution: If necessary replace the loss ℓ by an upper bound ℓ' and minimize the empirical loss: **SVR, SVM, Neural Network,Tree, Boosting...**

Empirical Risk Minimization

- The best solution $f^\star$ is the one minimizing

$$f^\star = \arg\min R(f) = \arg\min \mathbb{E}\left[\ell(Y, f(\underline{X}))\right]$$

Empirical Risk Minimization

- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the average empirical loss

$$\widehat{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(\underline{X}_i))$$

- Intractable for the $\ell^{0/1}$ loss!

Convexification Strategy

Risk Convexification

- Replace the loss $\ell(Y, f_\theta(\underline{X}))$ by a convex upperbound $\ell'(Y, f_\theta(\underline{X}))$ (surrogate loss).
- Minimize the average of the surrogate empirical loss

$$\tilde{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell'(Y_i, f_\theta(\underline{X}_i))$$

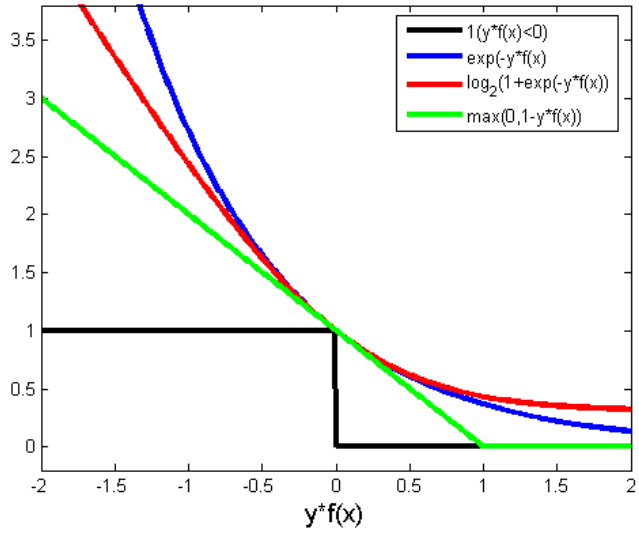
- Use $\widehat{f} = \operatorname{sign}(\tilde{f})$

- Much easier optimization.

Instantiation

- Logistic (Revisited)
- Support Vector Machine
- (Deep) Neural Network
- Boosting

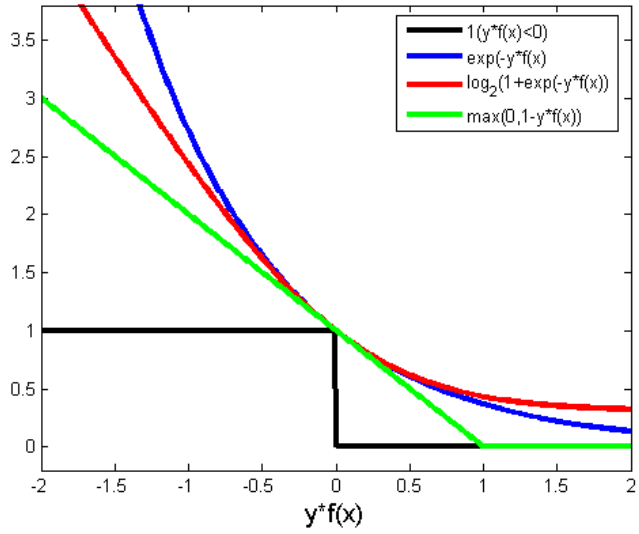
Classification Loss and Convexification



Convexification

- Replace the loss $\ell^{0/1}(Y, f(\underline{X}))$ by
$$\ell'(Y, f(\underline{X})) = l(Y_i f(\underline{X}))$$
with l a convex function.
- **Further mild assumption:** l is decreasing, differentiable at 0 and $l'(0) < 0$.

Classification Loss and Convexification



Classical convexification

- Logistic loss: $\ell'(Y, f(\underline{X})) = \log(1 + e^{-Yf(\underline{X})})$ (Logistic / NN)
- Hinge loss: $\ell'(Y, f(\underline{X})) = (1 - Yf(\underline{X}))_+$ (SVM)
- Exponential loss: $\ell'(Y, f(\underline{X})) = e^{-Yf(\underline{X})}$ (Boosting...)

Properties

The Target is the Bayes Classifier

- The minimizer of
$$\mathbb{E}\left[\ell'(Y, f(\underline{X}))\right] = \mathbb{E}\left[I(Yf(\underline{X}))\right]$$
is the Bayes classifier $f^\star = \text{sign}(2\eta(\underline{X}) - 1)$

Control of the Excess Risk

- It exists a convex function Ψ such that
$$\Psi\left(\mathbb{E}\left[\ell^{0/1}(Y, \text{sign}(f(\underline{X}))\right] - \mathbb{E}\left[\ell^{0/1}(Y, f^\star(\underline{X})\right]\right)$$
$$\leq \mathbb{E}\left[\ell'(Y, f(\underline{X})\right] - \mathbb{E}\left[\ell'(Y, f^\star(\underline{X}))\right]$$

- Theoretical guarantee!

Logistic Revisited

- Ideal solution:

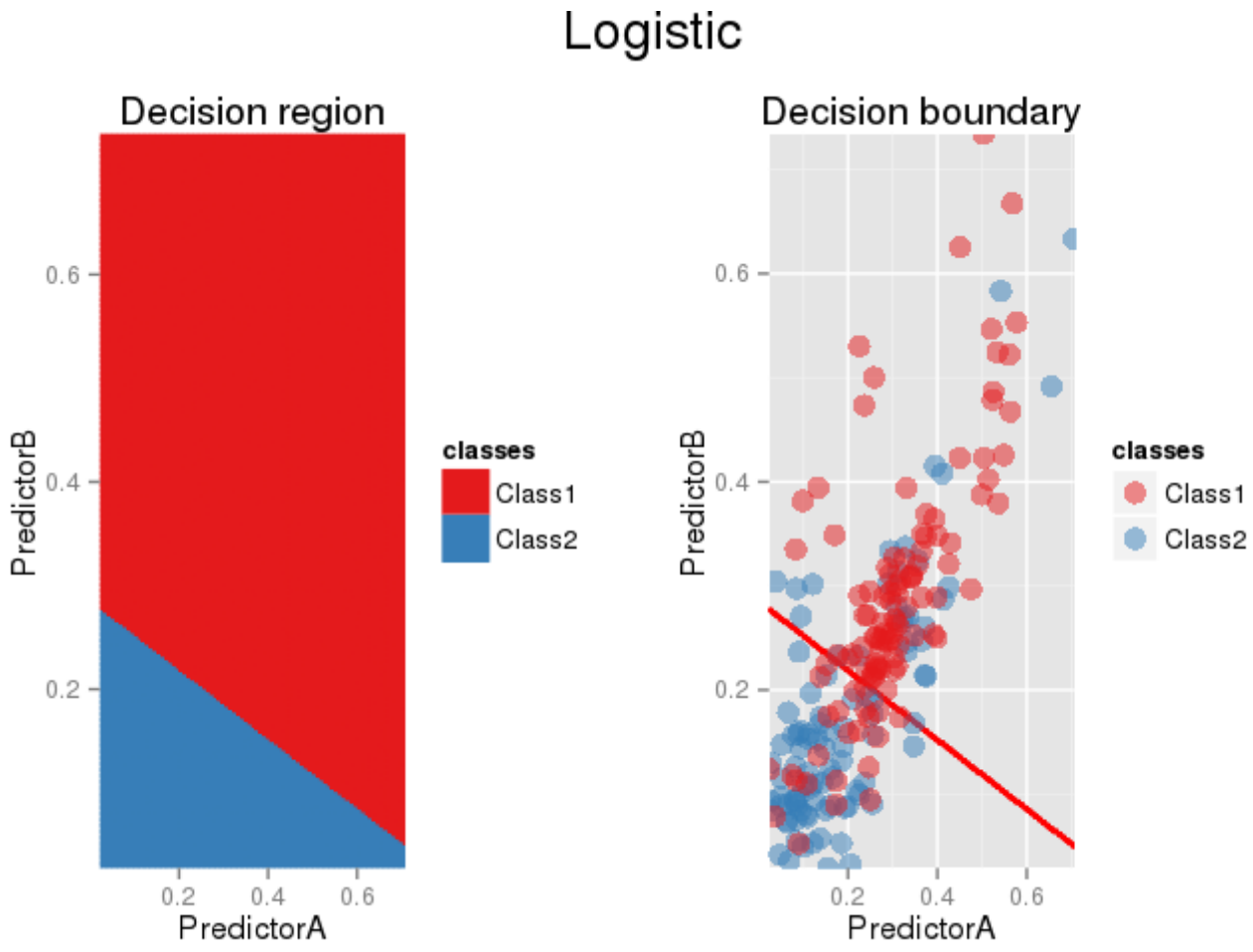
$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

Logistic regression

- Use $f(\underline{X}) = \underline{X}^\top \beta + \beta^{(0)}$.
- Use the logistic loss $\ell(y, f) = \log_2(1 + e^{-yf})$, i.e. the -log-likelihood.

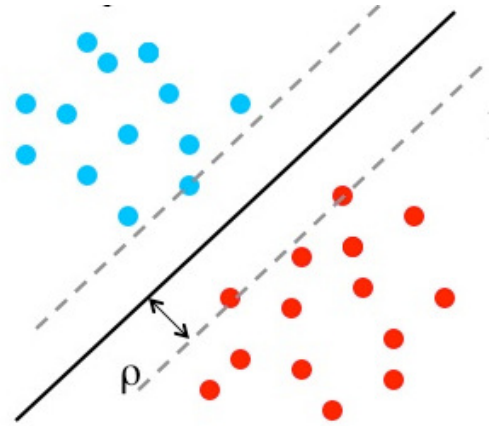
- Different vision than the statistician but same algorithm!

Logistic Revisited



Outline

Ideal Separable Case



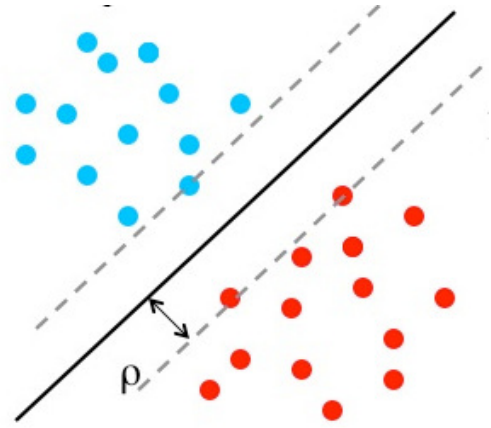
- Linear classifier: $\text{sign}(\underline{X}^\top \beta + \beta^{(0)})$
- Separable case: $\exists(\beta, \beta^{(0)}), \forall i,\ Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) > 0$!

How to choose (β, b) so that the separation is maximal?

- Strict separation: $\exists(\beta, \beta^{(0)}), \forall i,\ Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) \geq 1$
- Distance between $\underline{X}^\top \beta + \beta^{(0)} = 1$ and $\underline{X}^\top \beta + \beta^{(0)} = -1$: $$\frac{2}{\|\beta\|}$$
- Maximizing this distance is equivalent to minimizing $\frac{1}{2}\|\beta\|^2$.

Source: M. Mohri et al.

Ideal Separable Case



Separable SVM

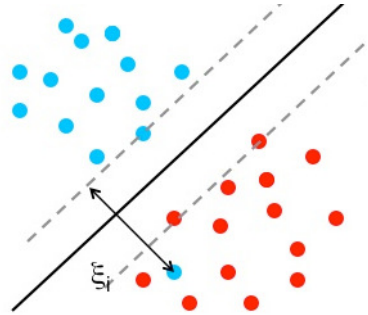
- Constrained optimization formulation:

$$\min \frac{1}{2}\|\beta\|^2 \quad \text{with} \quad \forall i, Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) \geq 1$$

- Quadratic Programming setting.
- Efficient solver available...

Source: M. Mohri et al.

Non Separable Case



- What about the non separable case?

SVM relaxation

- Relax the assumptions

$$\forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1$$

to

$$\forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1 - s_i$$

with the **slack variables** $s_i \geq 0$

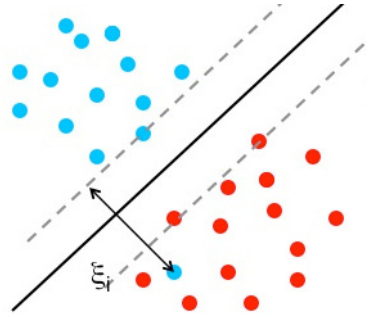
- Keep those slack variables as small as possible by minimizing

$$\frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i$$

where $C > 0$ is the **goodness-of-fit strength**

Source: M. Mohri et al.

Non Separable Case



SVM

- Constrained optimization formulation:

$$\min \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i \quad \text{with} \quad \begin{cases} \forall i, Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) \geq 1 - s_i \\ \forall i, s_i \geq 0 \end{cases}$$

- **Hinge Loss** reformulation:

$$\min \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \underbrace{\max(0, 1 - Y_i(\underline{X_i}^\top \beta + \beta^{(0)}))}_{\text{Hinge Loss}}$$

- Constrained convex optimization algorithms vs gradient descent algorithms.

Source: M. Mohri et al.

SVM as a Penalized Convex Relaxation

- Convex relaxation:

$$\text{argmin } \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(1 - Y_i(\underline{X_i}^\top \beta + \beta^{(0)}), 0)$$

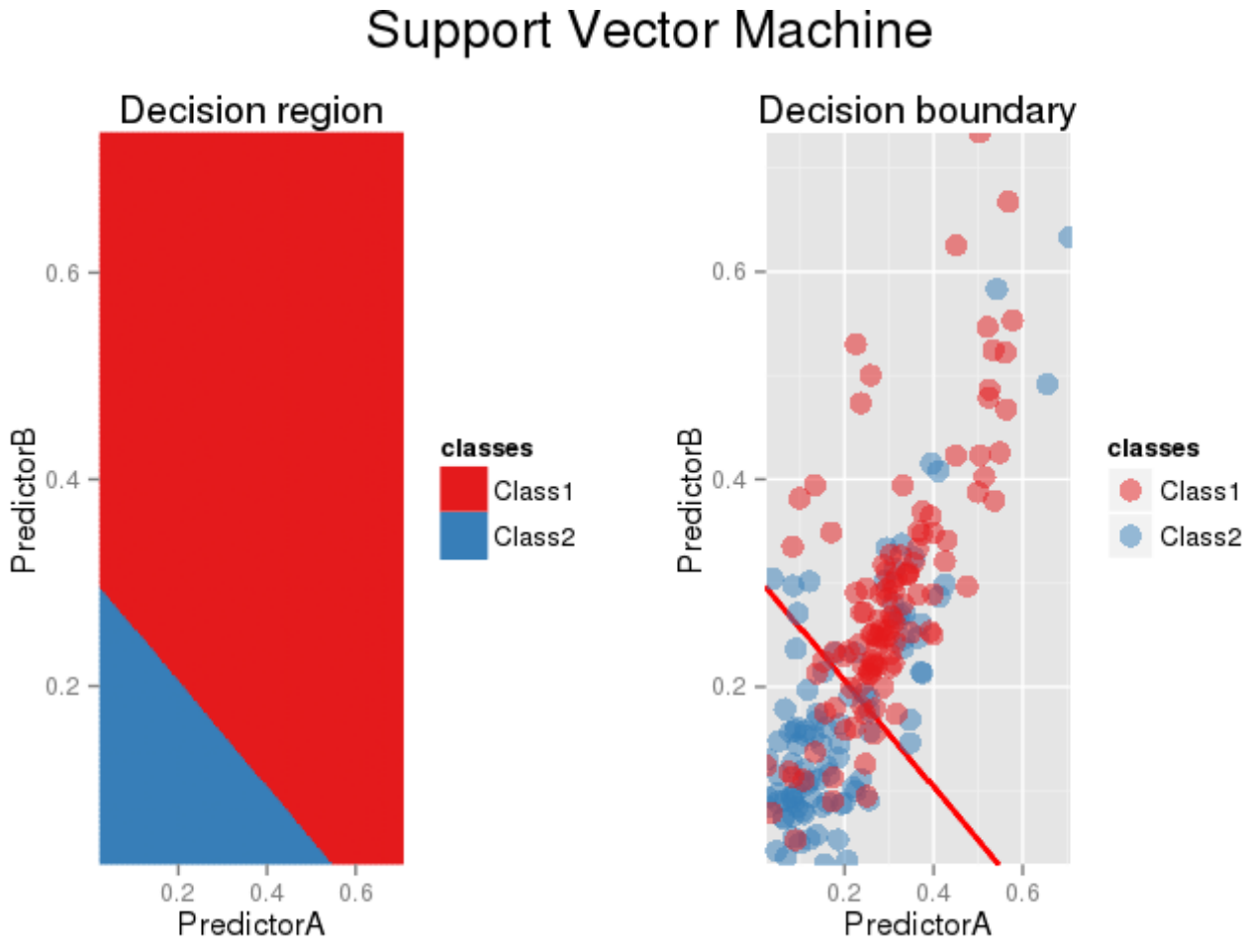
$$= \text{argmin } \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\underline{X_i}^\top \beta + \beta^{(0)}), 0) + \frac{1}{Cn}\frac{1}{2}\|\beta\|^2$$

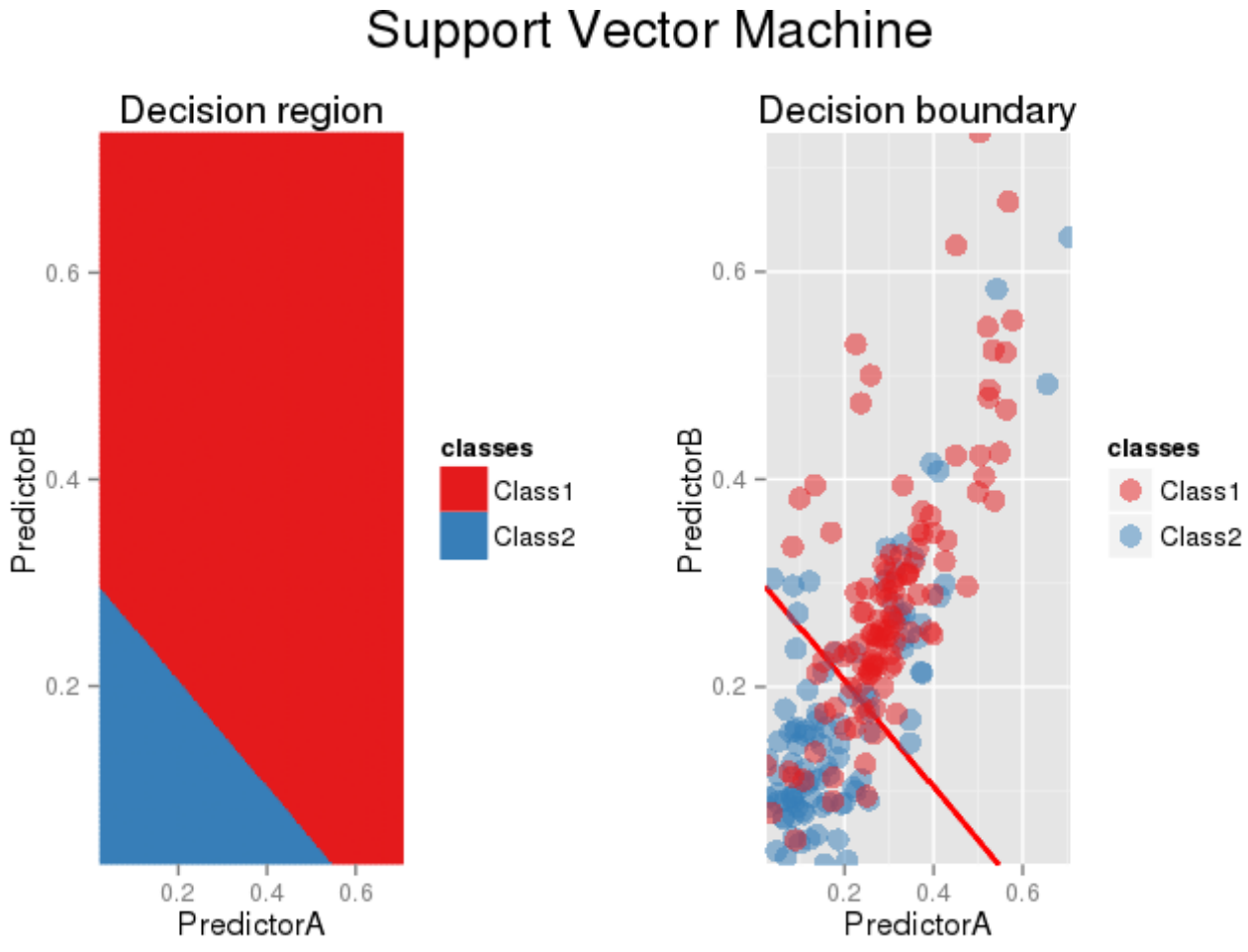
- **Prop:**
$\ell^{0/1}(Y_i, \text{sign}(\underline{X_i}^\top \beta + \beta^{(0)})) \leq \max(1 - Y_i(\underline{X_i}^\top \beta + \beta^{(0)}), 0)$

Penalized convex relaxation (Tikhonov!)

$$\frac{1}{n}\sum_{i=1}^{n} \ell^{0/1}(Y_i, \text{sign}(\underline{X_i}^\top \beta + \beta^{(0)}))$$

$$\leq \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\underline{X_i}^\top \beta + \beta^{(0)}), 0) + \frac{1}{Cn}\frac{1}{2}\|\beta\|^2$$


Support Vector Machine
Decision region
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2
Decision boundary
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2


Support Vector Machine
Decision region
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2
Decision boundary
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2

Constrained Minimization

Constrained Minimization

- Goal:

$$\min_x f(x)$$

$$\text{with } \begin{cases} h_j(x) = 0, & j = 1, \dots p \\ g_i(x) \leq 0, & i = 1, \dots q \end{cases}$$

- or rather with argmin!

Different Setting

- f, h_j, g_i **differentiable**
- f **convex**, h_j **affine** and g_i **concave**.

Feasibility

- x is **feasible** if $h_j(x) = 0$ and $g_i(x) \leq 0$.
- **Rk:** The set of feasible points may be empty

Lagrangian

Constrained Minimization

- Goal:

$$p^* = \min_x f(x) \quad \text{with} \quad \begin{cases} h_j(x) = 0, & j = 1, \dots p \\ g_i(x) \leq 0, & i = 1, \dots q \end{cases}$$

Lagrangian

- **Def:**

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x) + \sum_{i=1}^{q} \mu_i g_i(x)$$

  with $\lambda \in \mathbb{R}^p$ and $\mu \in (\mathbb{R}^+)^q$.

- The λ_j and μ_i are called the dual (or Lagrange) variables.
- **Prop:**

$$\max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \mathcal{L}(x, \lambda, \mu) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +\infty & \text{otherwise} \end{cases}$$

$$\min_x \max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \mathcal{L}(x, \lambda, \mu) = p^*$$

Lagrangial Dual

Lagrangian

- **Def:**

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x) + \sum_{i=1}^{q} \mu_i g_i(x)$$

with $\lambda \in \mathbb{R}^p$ and $\mu \in (\mathbb{R}^+)^q$.

Lagragian Dual

- Lagrandial dual function:

$$Q(\lambda, \mu) = \min_x \mathcal{L}(x, \lambda, \mu)$$

- **Prop:**

$$Q(\lambda, \mu) \leq f(x), \text{ for all feasible } x$$

$$\max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} Q(\lambda, \mu) \leq \min_{x \text{ feasible}} f(x)$$

Duality

Primal

- Primal:

$$p^* = \min_{x \in \mathcal{X}} f(x) \text{ with } \begin{cases} h_j(x) = 0, & j = 1, \dots p \\ g_i(x) \leq 0, & i = 1, \dots q \end{cases}$$

Dual

- Dual:

$$q^* = \max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} Q(\lambda, \mu) = \max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \min_x \mathcal{L}(x, \lambda, \mu)$$

Duality

- Always **weak duality:**

$$q^* \leq p^*$$

$$\max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \min_x \mathcal{L}(x, \lambda, \mu) \leq \min_x \max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \mathcal{L}(x, \lambda, \mu)$$

- Not always strong duality $q^* = p^*$.

Strong Duality

Strong Duality

- **Strong duality:**

$$q^* = p^*$$

$$\max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \min_x \mathcal{L}(x, \lambda, \mu) = \min_x \max_{\lambda \in \mathbb{R}^p,\ \mu \in (\mathbb{R}^+)^q} \mathcal{L}(x, \lambda, \mu)$$

- Allow to compute the solution of one problem from the other.
- Requires some assumptions!

Strong Duality under Convexity and Slater's Condition

- f **convex**, h_j **affine** and g_i **concave**.
- **Slater's condition:** it exists a feasible point such that $h_j(x) = 0$ for all j and $g_i(x) < 0$ for all i.
- Sufficient to prove **strong duality**.
- **Rk:** If the g_i are affine, it suffices to have $h_j(x) = 0$ for all j and $g_i(x) \leq 0$ for all i.

Karush-Kuhn-Tucker Condition

- Stationarity:
$$\nabla_x \mathcal{L}(x^*, \lambda, \mu) = \nabla f(x^*) + \sum_j \lambda_j \nabla h(x^*) + \sum_i \mu_i \nabla g(x^*) = 0$$
- Primal admissibility:
$$h_j(x^*) = 0 \quad \text{and} \quad g_i(x^*) \leq 0$$
- Dual admissibility:
$$\mu_i \geq 0$$
- Complementary slackness:
$$\mu_i g_i(x^*) = 0$$

KKT Theorem

- If f **convex**, h_j **affine** and g_i **concave**, all are differentiable and **strong duality** holds then x^* is a **solution** of the primal problem **if and only if** the **KKT condition holds**

- Same result without differentiability using the sub-gradient...

SVM and Lagrangian

SVM

- Constrained optimization formulation:

$$\min \|\beta\|^2 + C\sum_{i=1}^{n} s_i \quad \text{with} \quad \begin{cases} \forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1 - s_i \\ \forall i, s_i \geq 0 \end{cases}$$

SVM Lagragian

- Lagrangian:

$$\mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i + \sum_i \alpha_i(1 - s_i - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)})) - \sum_i \mu_i s_i$$

SVM and KKT

KKT Optimality Conditions

- Stationarity:

$$\nabla_\beta \mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu) = \beta - \sum_i \alpha_i Y_i \underline{X}_i = 0$$

$$\nabla_{\beta^{(0)}} \mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu) = \sum_i \alpha_i = 0$$

$$\nabla_{s_i} \mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu) = C - \alpha_i - \mu_i = 0$$

- Primal and dual admissibility:

$$(1 - s_i - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)})) \leq 0, \quad s_i \geq 0, \quad \alpha_i \geq 0, \text{ and } \mu_i \geq 0$$

- Complementary slackness:

$$\alpha_i(1 - s_i - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)})) = 0 \quad \text{and} \quad \mu_i s_i = 0$$

Consequence

- $\beta^* = \sum_i \alpha_i Y_i \underline{X}_i$ and $0 \leq \alpha_i \leq C$.
- If $\alpha_i \neq 0$, $\underline{X}_i$ is called a **support vector** and either
  - $s_i = 0$ and $Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) = 1$ (margin hyperplane),
  - or $\alpha_i = C$ (outliers).
- $\beta^{(0)*} = Y_i - \underline{X}_i{}^\top \beta^*$ for any support vector with $0 < \alpha_i < C$.

SVM Dual

SVM Lagrangian Dual

- Lagrangian Dual:

$$Q(\alpha, \mu) = \min_{\beta, \beta^{(0)}, s} \mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu)$$

- Prop:
  - if $\sum_i \alpha_i Y_i \neq 0$ or $\exists i, \alpha_i + \mu_i \neq C$,

$$Q(\alpha, \mu) = -\infty$$

  - if $\sum_i \alpha_i Y_i = 0$ and $\forall i, \alpha_i + \mu_i = C$,

$$Q(\alpha, \mu) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j Y_i Y_j \underline{X}_i^\top \underline{X}_j$$

SVM Dual problem

- Dual problem is a Quadratic Programming problem:

$$\max_{\alpha \geq 0, \mu \geq 0} Q(\alpha, \mu) \Leftrightarrow \max_{0 \leq \alpha \leq C} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j Y_i Y_j \underline{X}_i^\top \underline{X}_j$$

- Involves the $\underline{X}_i$ only through their scalar products.

Mercer Theorem

Mercer Representation Theorem

- For any loss ℓ and any increasing function Φ, the minimizer in β of

$$\sum_{i=1}^{n} \ell(Y_i, \underline{X}_i^\top \beta + \beta^{(0)}) + \Phi(\|\beta\|_2)$$

  is a linear combination of the input points $\beta^* = \sum_{i=1}^{n} \alpha_i' \underline{X}_i$.
- Minimization problem in α':

$$\sum_{i=1}^{n} \ell(Y_i, \sum_j \alpha_j' \underline{X}_i^\top \underline{X}_j + \beta^{(0)}) + \Phi(\|\beta\|_2)$$

  involving only the scalar product of the data.

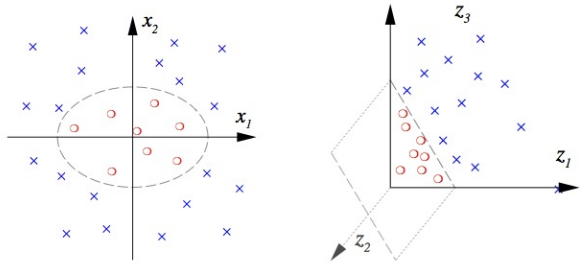
- Optimal predictor requires only to compute scalar products.

$$\hat{f}^*(\underline{X}) = \underline{X}^\top \beta^* + \beta^{(0)} = \sum_i \alpha_i' \underline{X}_i^\top \underline{X}$$

- Transform a problem in dimension $\dim(\mathcal{X})$ in a problem in dimension n.
- Direct minimization in β can be more efficient...

Feature Map

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



Feature Engineering

- Art of creating **new features** from the existing one $\underline{X}$.
- Example: add monomials $(\underline{X}^{(j)})^2$, $\underline{X}^{(j)}\underline{X}^{(j')}$...
- Adding feature increases the dimension.

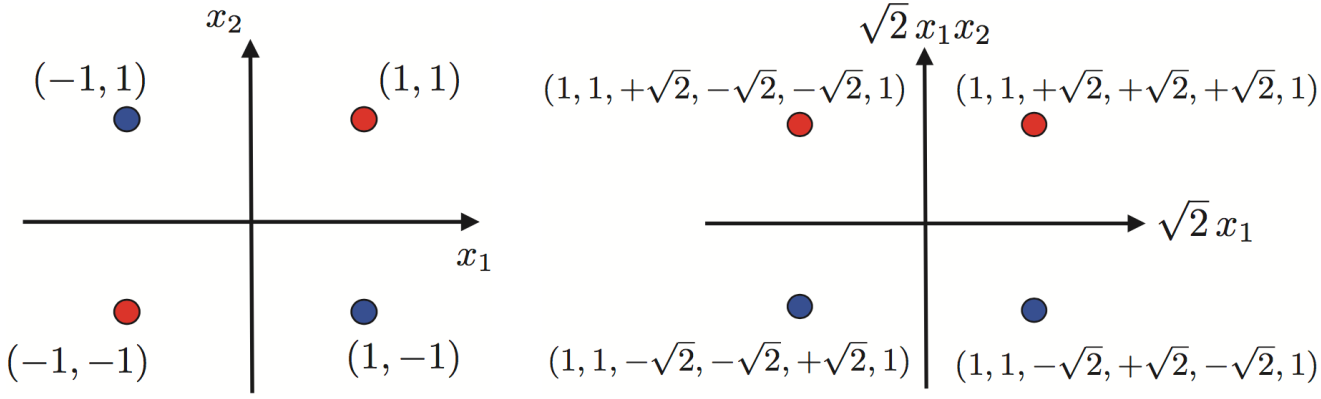
Feature Map

- Application $\phi : \mathcal{X} \to \mathbb{H}$ with $\mathbb{H}$ an Hilbert space.
- Linear decision boundary in $\mathbb{H}$: $\phi(\underline{X})^\top \beta + \beta^{(0)} = 0$ is **not an hyperplane anymore** in $\mathcal{X}$.

- **Heuristic:** Increasing dimension allows to make data almost linearily separable.

Source: Unknown

Polynomial Mapping



Polynomial Mapping of order 2

- $\phi : \mathbb{R}^2 \to \mathbb{R}^6$

$$\phi(\underline{X}) = \left((\underline{X}^{(1)})^2, (\underline{X}^{(2)})^2, \sqrt{2}\underline{X}^{(1)}\underline{X}^{(2)}, \sqrt{2}\underline{X}^{(1)}, \sqrt{2}\underline{X}^{(2)}, 1\right)$$

- Allow to solve the XOR classification problem with the *hyperplane* $\underline{X}^{(1)}\underline{X}^{(2)} = 0$.

Polynomial Mapping and Scalar Product

- **Prop:**

$$\phi(\underline{X})^\top \phi(\underline{X}') = (1 + \underline{X}^\top \underline{X}')^2$$

Source: M. Mohri et al.

SVM Primal and Dual

Primal, Lagrandian and Dual

- Primal:

$$\min \|\beta\|^2 + C\sum_{i=1}^{n} s_i \quad \text{with} \quad \begin{cases} \forall i, Y_i(\phi(\underline{X}_i)^\top \beta + \beta^{(0)}) \geq 1 - s_i \\ \forall i, s_i \geq 0 \end{cases}$$

- Lagrangian:

$$\mathcal{L}(\beta, \beta^{(0)}, s, \alpha, \mu) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i + \sum_i \alpha_i(1 - s_i - Y_i(\phi(\underline{X}_i)^\top \beta + \beta^{(0)})) - \sum_i \mu_i s_i$$

- Dual:

$$\max_{\alpha \geq 0, \mu \geq 0} Q(\alpha, \mu) \Leftrightarrow \max_{0 \leq \alpha \leq C} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j Y_i Y_j \phi(\underline{X}_i)^\top \phi(\underline{X}_j)$$

- Optimal $\underline{X}^\top \beta^* = \sum_i \alpha_i Y_i \phi(\underline{X})^\top \phi(\underline{X}_i)$

- Only need to know to compute $\phi(\underline{X})^\top \phi(\underline{X}')$ to obtain the solution.

From Map to Kernel

- Many algorithms (e.g. SVM) require only to be able to compute the scalar product $\phi(\underline{X})^\top \phi(\underline{X}')$.

Kernel

- Any application
$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
is called a **kernel** over $\mathcal{X}$.

Kernel Trick

- Computing directly the **kernel** $k(x, x') = \phi(\underline{X})^\top \phi(\underline{X}')$ may be easier than computing $\phi(\underline{X})$, $\phi(\underline{X}')$ and then the scalar product.

- Here k is defined from ϕ.
- Under some assumption on k, ϕ can be implicitely *defined* from k!

PDS Kernel

Positive Definite Symmetric Kernels

- A kernel k is PDS if and only if
  - k is symmetric, i.e.
    $$k(\underline{X}, \underline{X}') = k(\underline{X}', \underline{X})$$
  - for any $N \in \mathbb{N}$ and any $(\underline{X}_1, \ldots, \underline{X}_N) \in \mathcal{X}^N$,
    $$\boldsymbol{K} = [k(\underline{X}_i, \underline{X}_j)]_{1 \leq i,j \leq N}$$
    is positive semi-definite, i.e. $\forall u \in \mathbb{R}^N$
    $$u^\top \boldsymbol{K} u = \sum_{1 \leq i,j \leq N} u^{(i)} u^{(j)} k(\underline{X}_i, \underline{X}_j) \geq 0$$
    or equivalently all the eigenvalues of $\boldsymbol{K}$ are non-negative.

- The matrix $\boldsymbol{K}$ is called the **Gram matrix** associated to $(\underline{X}_1, \ldots, \underline{X}_N)$.

Reproducing Kernel Hilbert Space

Mercer Theorem

- For any PDS kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, it exists a Hilbert space $\mathbb{H} \subset \mathbb{R}^{\mathcal{X}}$ with a scalar product $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ such that
  - it exists a mapping $\phi : \mathcal{X} \to \mathbb{H}$ satisfying
$$k(\underline{X}, \underline{X}') = \langle \phi(\underline{X}), \phi(\underline{X}) \rangle_{\mathbb{H}}$$
  - the **reproducing property** holds, i.e. for any $h \in \mathbb{H}$ and any $\underline{X} \in \mathcal{X}$
$$h(\underline{X}) = \langle h, k(\underline{X}, \cdot) \rangle_{\mathbb{H}} .$$

- By def., $\mathbb{H}$ is a **reproducing kernel Hilbert space** (RKHS).
- $\mathbb{H}$ is called the **feature space** associated to k and ϕ the **feature mapping**.
- No unicity in general.
- **Rk:** if $k(\underline{X}, \underline{X}') = \phi'(\underline{X})^\top \phi'(\underline{X}')$ with $\phi' : \mathcal{X} \to \mathbb{R}^p$ then
  - $\mathbb{H}$ can be chosen as $\{\underline{X} \mapsto \phi'(\underline{X})^\top \beta, \beta \in \mathbb{R}^p\}$ and $\|\underline{X} \mapsto \phi'(\underline{X})^\top \beta\|_{\mathbb{H}}^2 = \|\beta\|_2^2$.
  - $\phi(\underline{X})(\underline{X}') = \underline{X}^\top \underline{X}'$.

Kernel Construction Machinery

Separable Kernel

- For any function $\Psi : \mathcal{X} \to \mathbb{R}$, $k(\underline{X}, \underline{X}') = \Psi(\underline{X})\Psi(\underline{X}')$ is PDS.

Kernel Stability

- For any PDS kernels k_1 and k_2, $k_1 + k_2$ and $k_1 k_2$ are PDS kernels.
- For any sequence of PDS kernels k_n converging pointwise to a kernel k, k is a PDS kernel.
- For any PDS kernel k such that $|k| \leq r$ and any power series $\sum_n a_n z^n$ with $a_n \geq 0$ and a convergence radius larger than r, $\sum_n a_n k^n$ is a PDS kernel.
- For any PDS kernel k, the renormalized kernel $k'(\underline{X}, \underline{X}') = \dfrac{k(\underline{X}, \underline{X}')}{\sqrt{k(\underline{X}, \underline{X})k(\underline{X}', \underline{X}')}}$ is a PDS kernel.

- Cauchy-Schwartz for k PDS: $k(\underline{X}, \underline{X}')^2 \leq k(\underline{X}, \underline{X})k(\underline{X}', \underline{X}')$

Classical Kernels

PDS Kernels

- Vanilla kernel:
$$k(\underline{X}, \underline{X}') = \underline{X}^\top \underline{X}'$$
- Polynomial kernel:
$$k(\underline{X}, \underline{X}') = (1 + \underline{X}^\top \underline{X}')^k$$
- Gaussian RBF kernel:
$$k(\underline{X}, \underline{X}') = \exp\left(-\gamma \|\underline{X} - \underline{X}'\|^2\right)$$
- Tanh kernel:
$$k(\underline{X}, \underline{X}') = \tanh(a\underline{X}^\top \underline{X}' + b)$$

- Most classical is the Gaussian RBF kernel...
- Lots of freedom to construct kernel for non classical data.

Representer Theorem

Representer Theorem

- Let k be a PDS kernel and $\mathbb{H}$ its corresponding RKHS, for any increasing function Φ and any function $L : \mathbb{R}^n \to \mathbb{R}$, the optimization problem

$$\operatorname*{argmin}_{h \in \mathbb{H}} L(h(\underline{X}_1), \ldots, h(\underline{X}_n)) + g(\|h\|)$$

admits only solutions of the form

$$\sum_{i=1}^{n} \alpha_i' k(\underline{X}_i, \cdot).$$

- Examples:
  - (kernelized) SVM
  - (kernelized) Penalized Logistic Regression (Ridge)
  - (kernelized) Penalized Regression (Ridge)

Kernelized SVM

Primal

- Constrained Optimization:

$$\min_{f\in\mathbb{H},\beta^{(0)},s} \|f\|_{\mathbb{H}}^2 + C\sum_{i=1}^{n} s_i \quad \text{with} \quad \begin{cases} \forall i,\ Y_i(f(\underline{X}_i)+\beta^{(0)}) \geq 1 - s_i \\ \forall i,\ s_i \geq 0 \end{cases}$$

- Hinge loss:

$$\min_{f\in\mathbb{H},\beta^{(0)}} \|f\|_{\mathbb{H}}^2 + C\sum_{i=1}^{n} \max(0, 1 - Y_i(f(\underline{X}_i)+\beta^{(0)}))$$

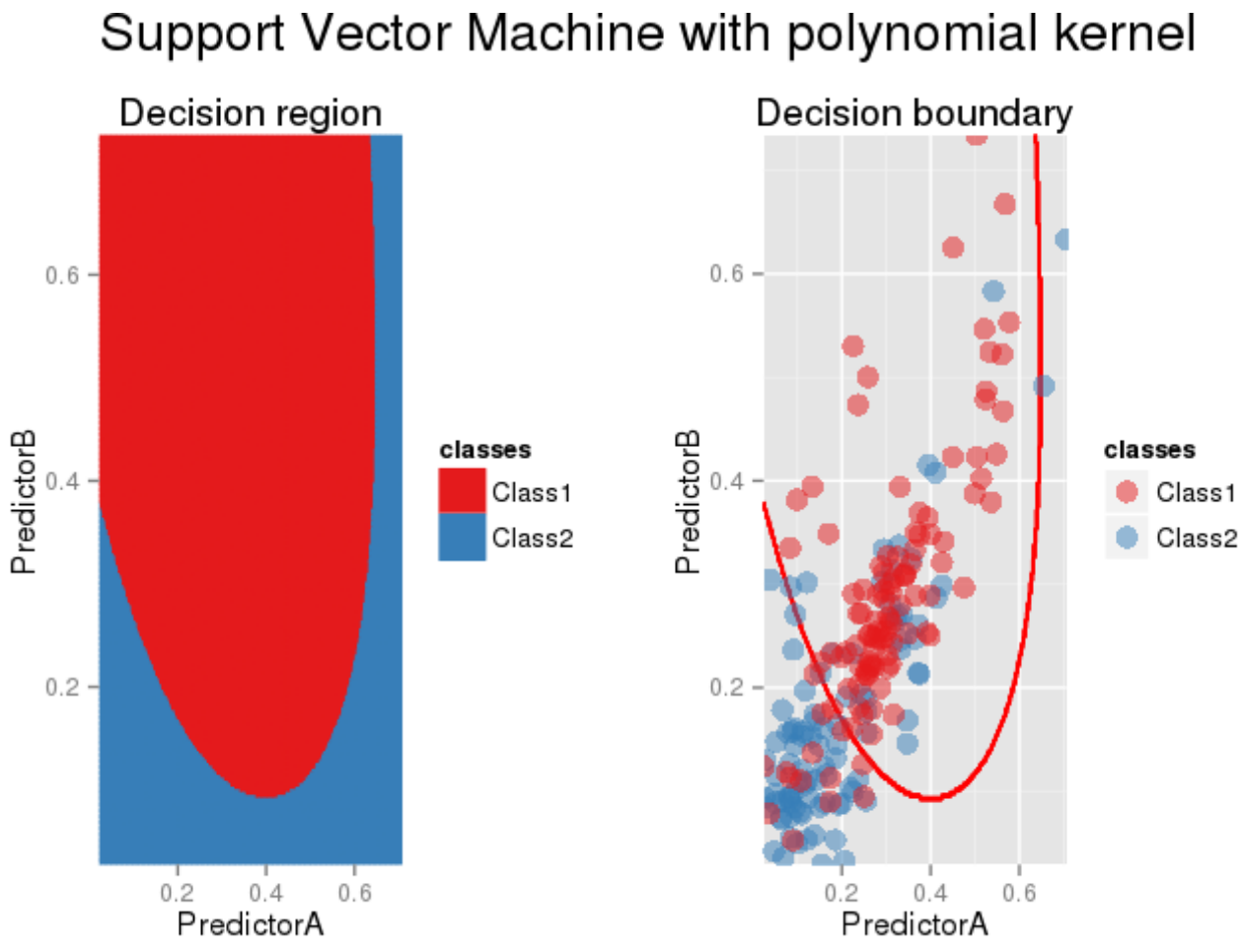
- Representer:

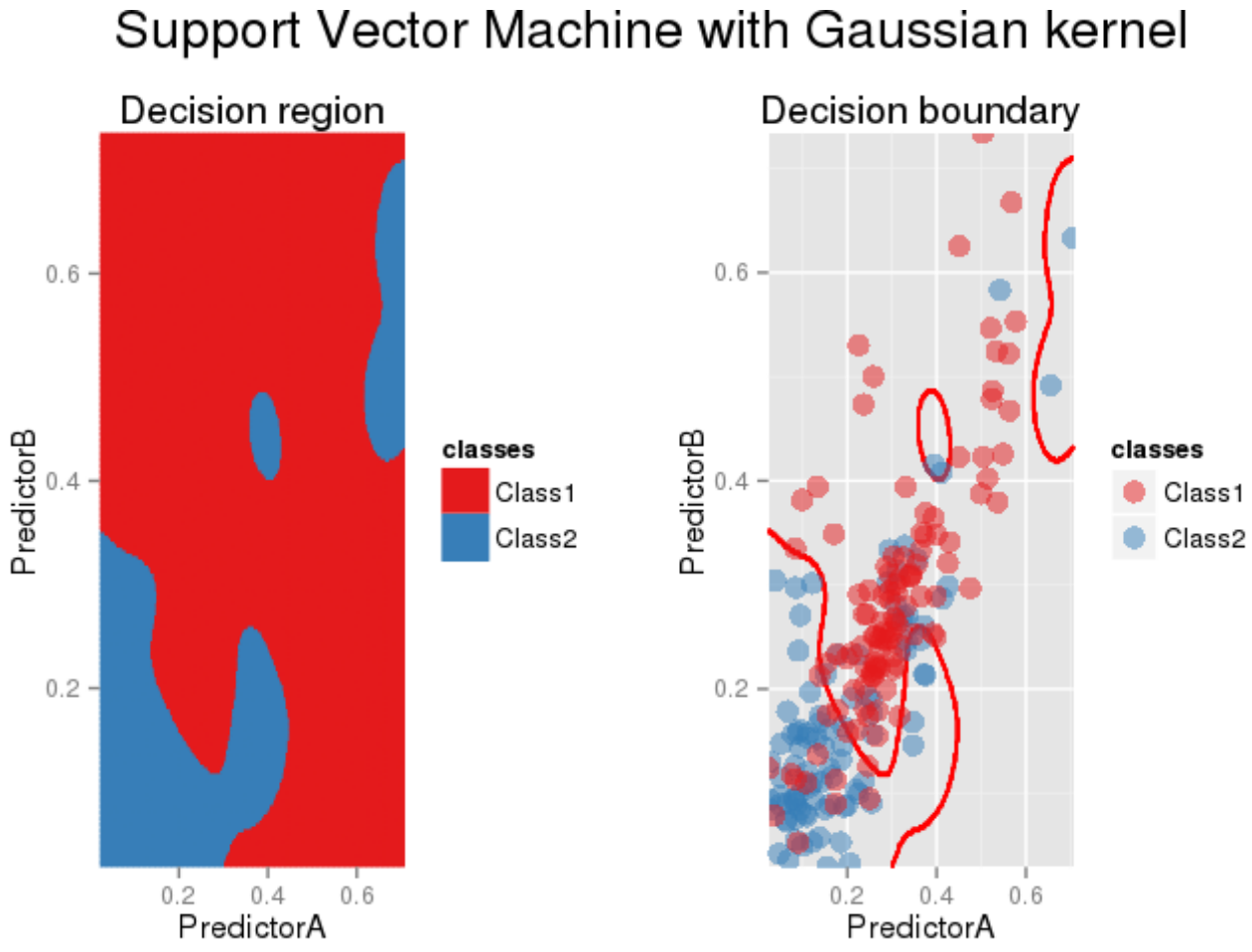
$$\min_{\alpha',\beta^{(0)}} \sum_{i,j} \alpha'_i\alpha'_j k(\underline{X}_i,\underline{X}_j) + C\sum_{i=1}^{n} \max(0, 1 - Y_i(\sum_j \alpha'_j k(\underline{X}_j,\underline{X}_i)+\beta^{(0)}))$$

Dual

- Dual:

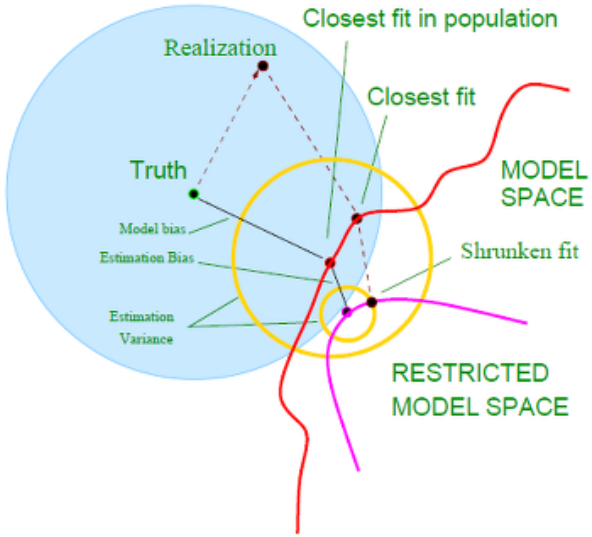
$$\max_{\alpha\geq 0,\mu\geq 0} Q(\alpha,\mu) \Leftrightarrow \max_{0\leq\alpha\leq C} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j Y_i Y_j k(\underline{X}_i,\underline{X}_j)$$


Support Vector Machine with polynomial kernel
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2


Support Vector Machine with Gaussian kernel
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Outline

Simplified Models



Bias-Variance Issue

- Most complex models may not be the best ones due to the variability of the estimate.

- Naive idea: can we *simplify* our model without loosing too much?
  - by using only a subset of the variables?
  - by forcing the coefficients to be small?
- Can we do better than exploring all possibilities?

Source: Tibshirani et al.

Linear Models

- **Setting**: Gen. linear model = prediction of Y by $h(\underline{x}^\top \beta)$.

Model coefficients

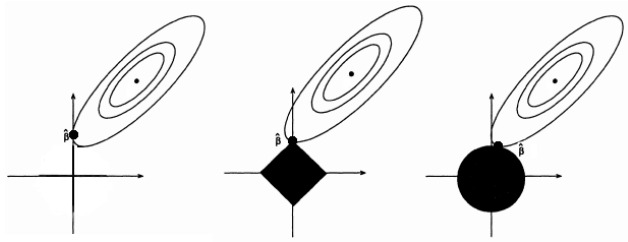
- Model entirely specified by β.
- Coefficientwise:
  - $\beta^{(i)} = 0$ means that the ith covariate is not used.
  - $\beta^{(i)} \sim 0$ means that the ith covariate as a *low* influence...

- If some covariates are useless, better use a simpler model...

Submodels

- *Simplify* the model through a constraint on β!
- Examples:
  - Support: Impose that $\beta^{(i)} = 0$ for $i \notin I$.
  - Support size: Impose that $\|\beta\|_0 = \sum_{i=1}^{d} \mathbf{1}_{\beta^{(i)} \neq 0} < C$
  - Norm: Impose that $\|\beta\|_p < C$ with $1 \leq p$ (Often $p = 2$ or $p = 1$)

Norms and Sparsity



Sparsity

- β is sparse if its number of non-zero coefficients (ℓ_0) is small...
- Easy interpretation in term of dimension/complexity.

Norm Constraint and Sparsity

- Sparsest solution obtained by definition with the ℓ_0 norm.
- No induced sparsity with the ℓ_2 norm...
- Sparsity with the ℓ_1 norm (can even be proved to be the same than with the ℓ_0 norm under some assumptions).
- Geometric explanation.

Source: Tibshirani et al.

Constraint and Penalization

Constrained Optimization

- Choose a constant C.
- Compute β as

$$\underset{\beta\in\mathbb{R}^d, \|\beta\|_p \leq C}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, h(\underline{x}_i{}^\top \beta))$$

Lagrangian Reformulation

- Choose λ and compute β as

$$\underset{\beta\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, h(\underline{x}_i{}^\top \beta)) + \lambda \|\beta\|_p^{p'}$$

  with $p' = p$ except if $p = 0$ where $p' = 1$.
- Easier calibration... but no explicit model $\mathcal{S}$.

- **Rk:** $\|\beta\|_p$ is not scaling invariant if $p \neq 0$...
- Initial rescaling issue.

Penalization

Penalized Linear Model

- Minimization of

$$\underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, h(\underline{x}_i{}^\top \beta)) + \text{pen}(\beta)$$

  where $\text{pen}(\beta)$ is a (sparsity promoting) penalty
- Variable selection if β is sparse.

Classical Penalties

- AIC: $\text{pen}(\beta) = \lambda \|\beta\|_0$ (non convex / sparsity)
- Ridge: $\text{pen}(\beta) = \lambda \|\beta\|_2^2$ (convex / no sparsity)
- Lasso: $\text{pen}(\beta) = \lambda \|\beta\|_1$ (convex / sparsity)
- Elastic net: $\text{pen}(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$ (convex / sparsity)

- Easy optimization if pen (and the loss) is convex...
- **Need to specify λ to define a ML method!**

Penalized Gen. Linear Models

Classical Examples

- Penalized Least Squares
- Penalized Logistic Regression
- Penalized Maximum Likelihood
- SVM
- Tree pruning

- Sometimes used even if the parametrization is not linear...

Penalization

Penalized $\ell^{0/1}$ loss (Structural Risk Minimization)

- Minimization of

$$\underset{f_m, m\in\mathcal{M}, f_m\in\mathcal{S}_m}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell^{0/1}(Y_i, f_m(\underline{X}_i)) + \text{pen}(m)$$

  where $\text{pen}(m)$ is a complexity driven penalty...

- No easy optimization here!

Classical Penalties

- Finite class: $\text{pen}(m) = \lambda\sqrt{\frac{\log|\mathcal{M}|}{n}}$
- Finite VC Dimension: $\text{pen}(m) = \lambda\sqrt{\frac{d_{VC}(\mathcal{S}_m)\log\left(\frac{en}{d_{VC}(\mathcal{S}_m)}\right)}{n}}$

- **Need to specify λ!**

Convexified Loss Penalization

Penalized convexified ℓ loss

- Minimization of

$$\underset{f_m, m\in\mathcal{M}, f_m\in\mathcal{S}_m}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f_m(X_i)) + \text{pen}(m)$$

where $\text{pen}(m)$ is a complexity driven penalty...

- Easy optimization here!
- **Reuse the previous** $\text{pen}(m)$**!**
- **Need to specify** λ**!**
- SVM case:
  - $d_{VC} \sim \|\beta\|^2$ which advocates for a penalty in $\lambda\|\beta\|$...
  - A penalty in $\lambda'\|\beta\|^2$ is more convenient numerically and there is a correspondence between the two problems...

Penalization and Cross-Validation

Practical Selection Methodology

- Choose a penalty shape $\widetilde{\text{pen}}$.
- Compute a CV error for a penalty $\lambda\widetilde{\text{pen}}$ for all $\lambda \in \Lambda$.
- Determine $\widehat{\lambda}$ the λ minimizing the CV error.
- Compute the final model with a penalty $\widehat{\lambda}\widetilde{\text{pen}}$.

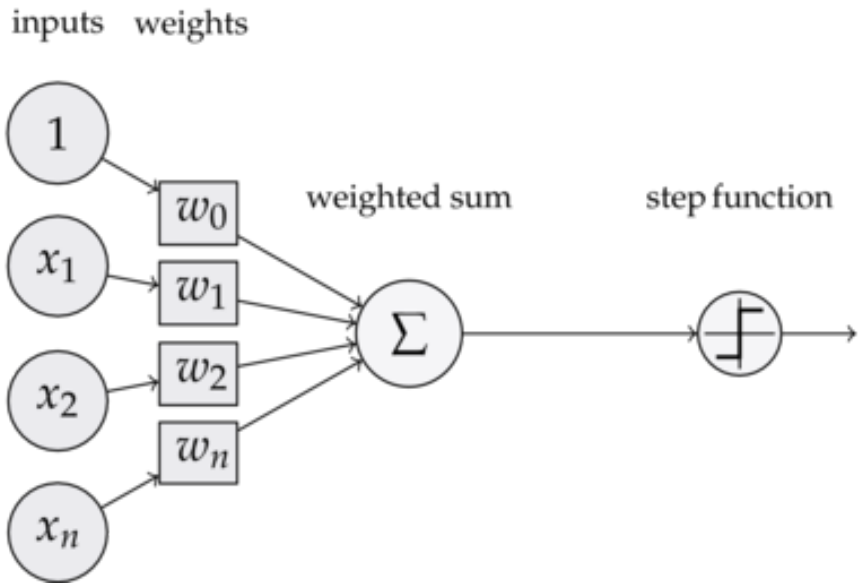
- CV allows to select a ML method, penalized estimation with a penalty $\widehat{\lambda}\widetilde{\text{pen}}$, not a single predictor hence the need of a final reestimation.

Why not using only CV?

- **If the penalized minimization is easy**, much cheaper to compute the CV error for all $\lambda \in \Lambda$ than for all possible estimators (or even models)...
- CV performs best when the set of candidates is not too big (or is structured...)

Outline

Perceptron

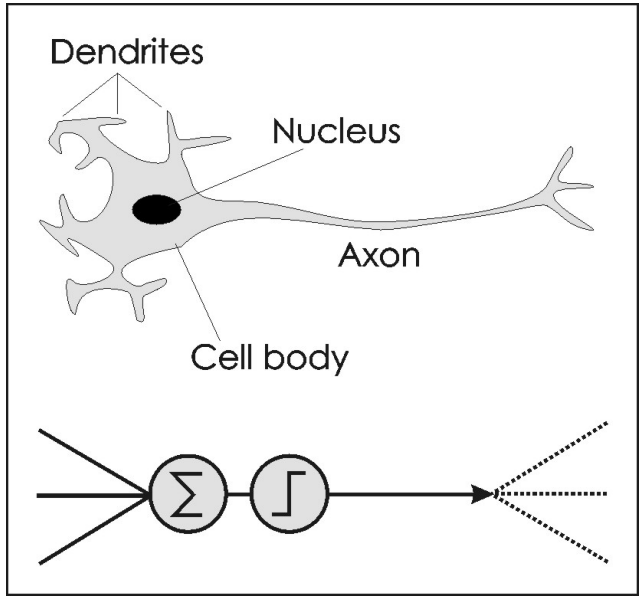


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

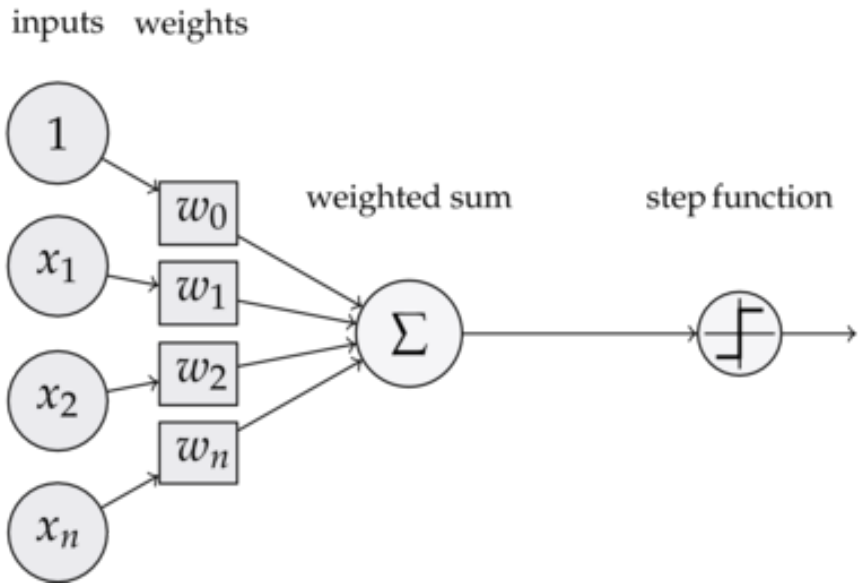


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

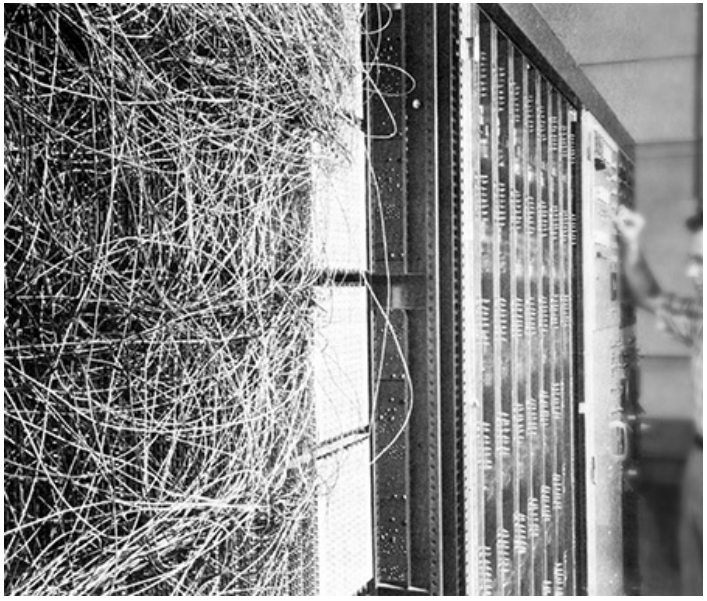


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

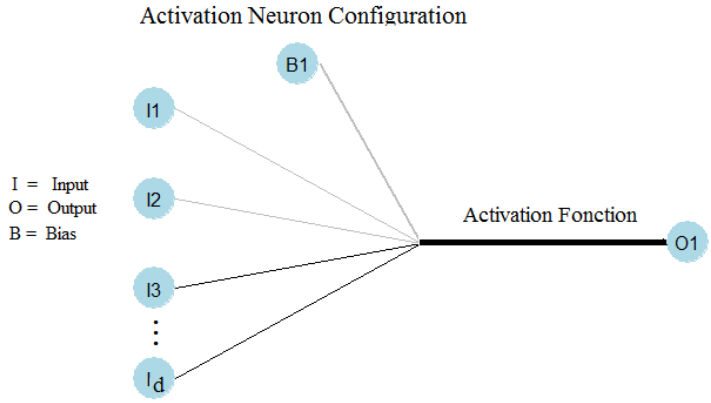


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Avin Calspan Advanced Technology Center

Artificial Neuron and Logistic Regression



Artificial neuron

- Structure:
  - Mix inputs with a **weighted sum**,
  - Apply a (non linear) **activation function** to this sum,
  - Eventually threshold the result to make a decision.
- Weights learned by minimizing a loss function.

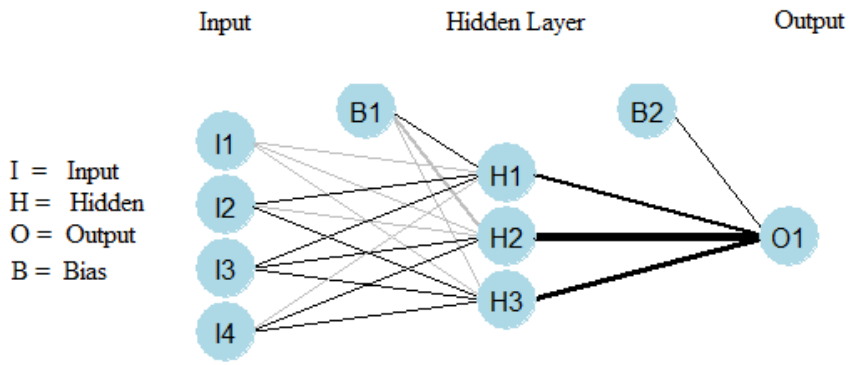
Logistic unit

- Structure:
  - Mix inputs with a **weighted sum**,
  - Apply **logistic function** $\sigma(t) = e^t/(1+e^t)$,
  - Threshold at $1/2$ to make a decision!
- Logistic weights learned by minimizing the -log-likelihood.

- Equivalent to linear regression when using a linear activation function!

Source: Unknown

Multilayer Perceptron



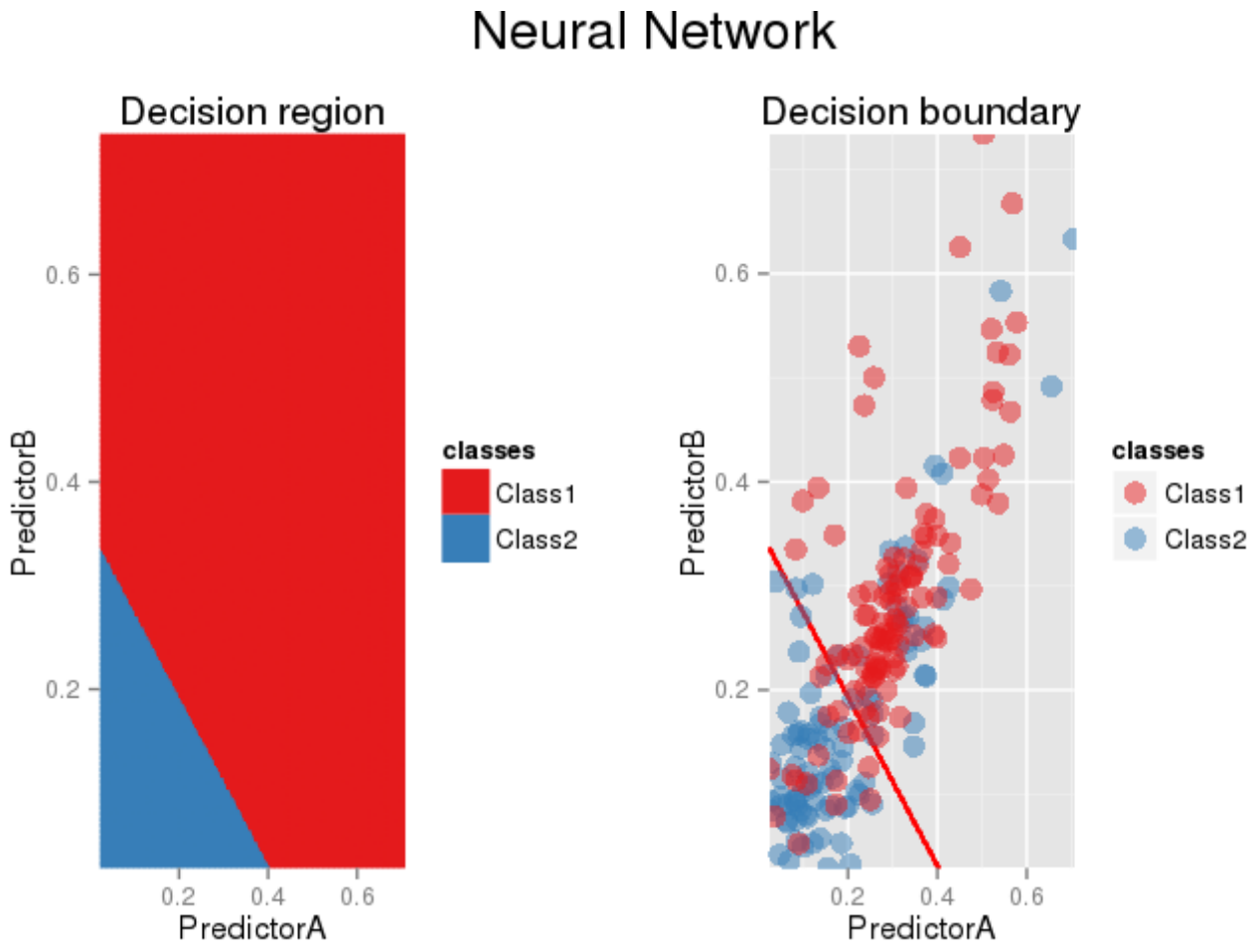
MLP (Rumelhart, McClelland, Hinton - 1986)

- Multilayer Perceptron: cascade of layers of artificial neuron units.
- Optimization through a gradient descent algorithm with a clever implementation (**Backprop**)

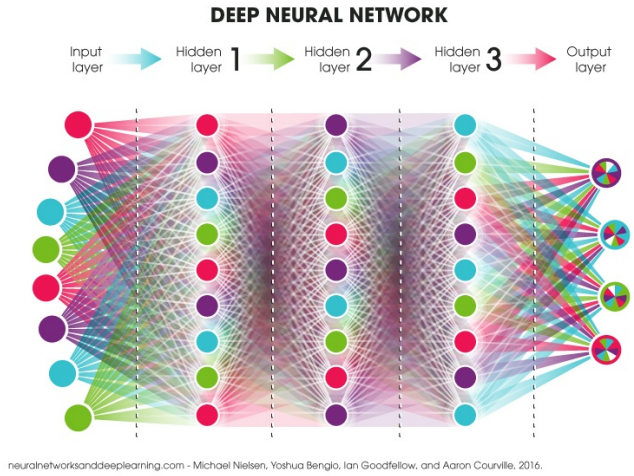
- Construction of a function by composing simple units.
- MLP corresponds to a specific direct acyclic graph structure.
- Non convex optimization problem!

Source: Unknown

Multilayer Perceptron



Deep Neural Network



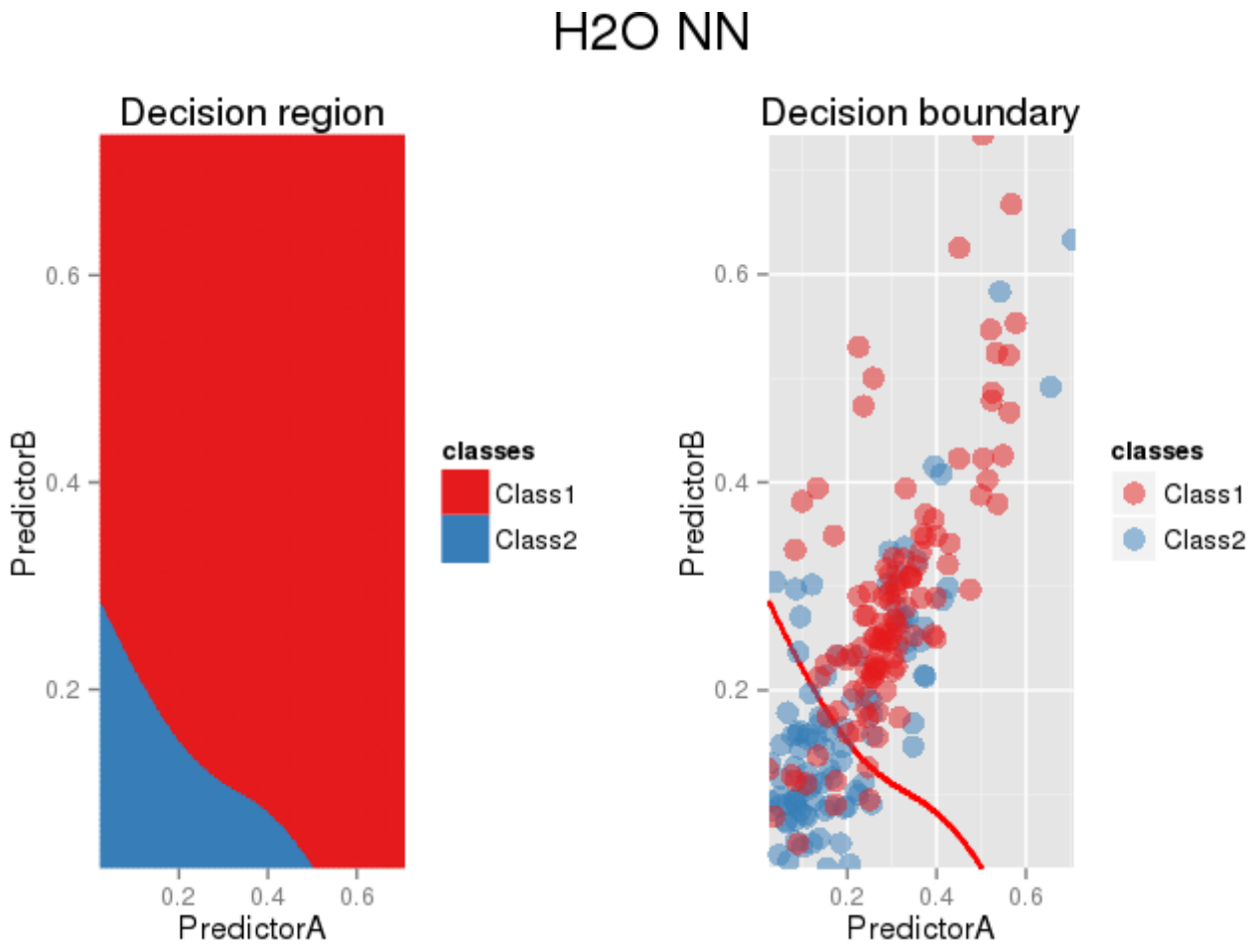
Deep Neural Network structure

- Deep cascade of layers!

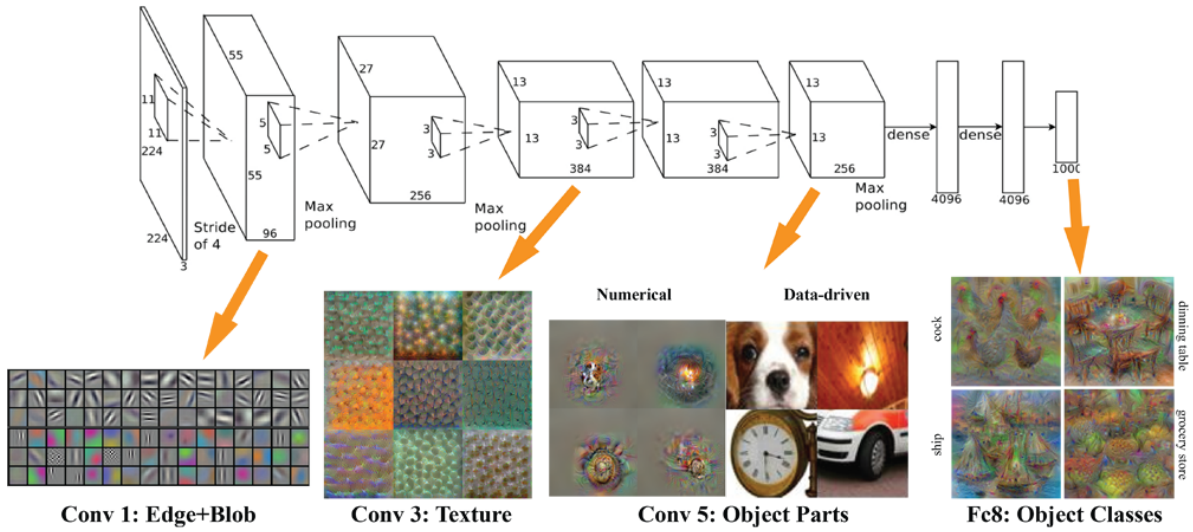
- No conceptual novelty...
- But a **lot of tricks** allowing to obtain a good solution: clever initialization, better activation function, weight regularization, accelerated stochastic gradient descent, early stopping...
- Use of GPU and a lot of data...
- Very impressive results!

Source: Nielsen, Bengio, Goodfellow and Courville

Deep Neural Network



Deep Learning



Family of Machine Learning algorithm combining:

- a (deep) multilayered structure,
- a clever optimization including initialization and regularization.

- Examples: Deep Neural Network, AutoEncoder, Recursive Neural Network...
- Interpretation as a **Representation Learning**
- **Transfer learning:** use as initialization a pretrained net.
- Very efficient and still evolving!

Source: J. Hays

Convolutional Network

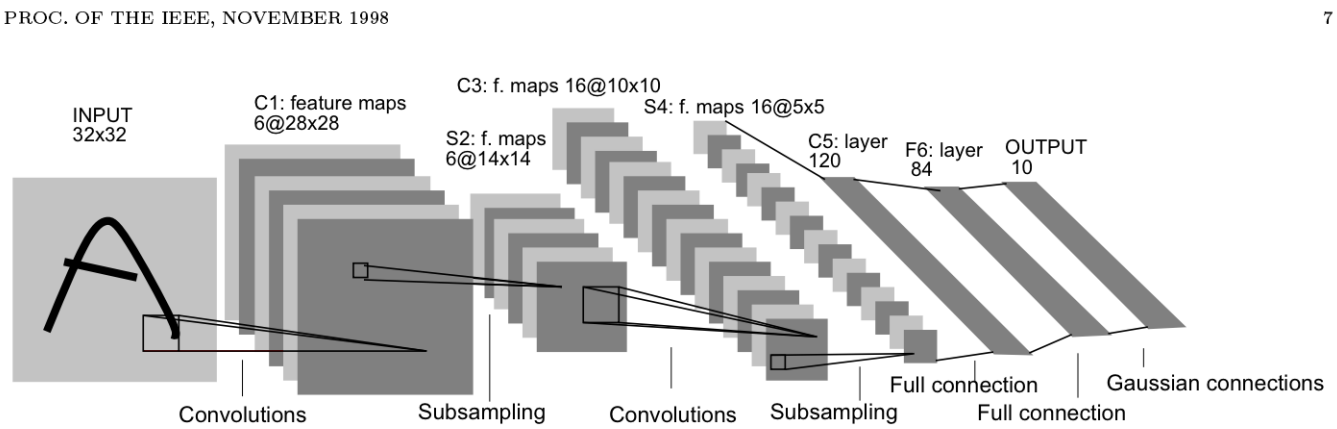


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

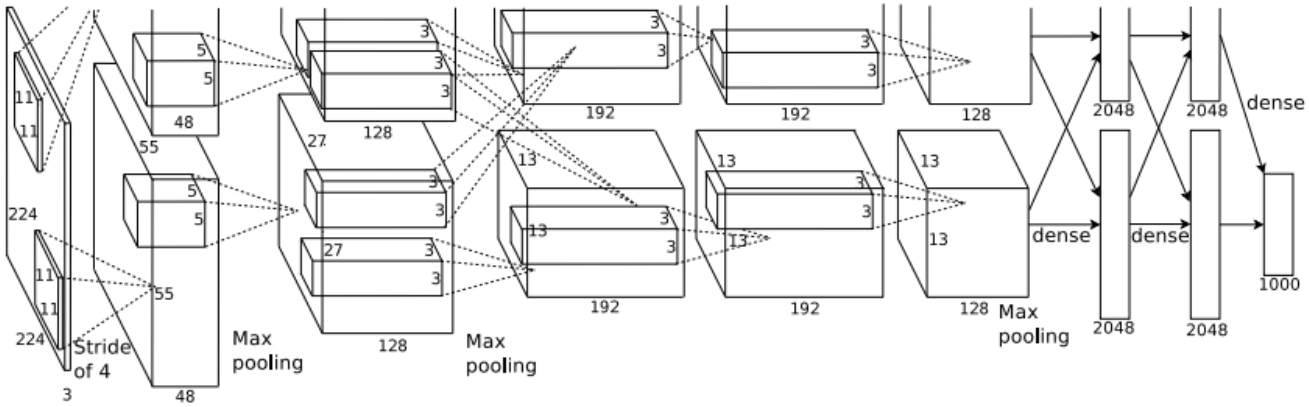
Le Net - Y. LeCun (1989)

- 6 Hidden layer architecture
- Drastic reduction of the number of parameters through a translation invariance principle (convolution)

- Requires 3 days of training for 60 000 examples!
- Tremendous improvement.
- Representation learned through the task.

Source: Y. LeCun

Deep Convolutional Networks



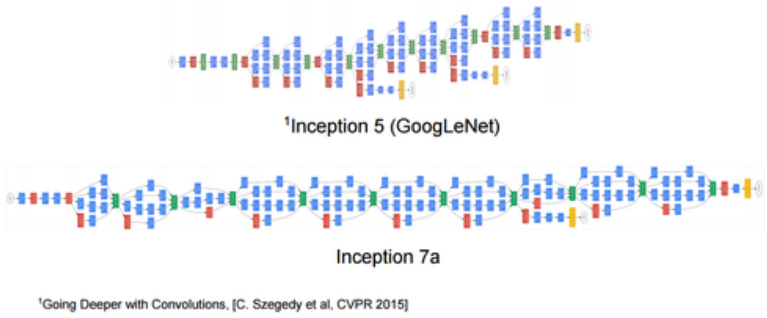
Alenet - A. Krizhevsky, I. Sutskever, G. Hinton (2012)

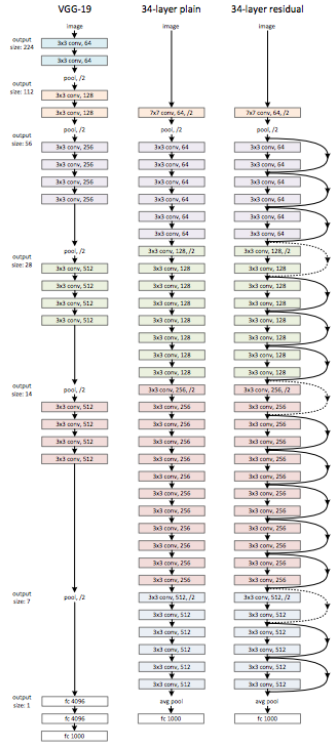
- Bigger and deepr layers and thus much more parameters.
- Clever intialization scheme, RELU, renormalization and use of GPU.

- 6 days of training for 1.2 millions images.
- Tremendous improvement...

Source: A. Krizhevsky

Deep Convolutional Networks





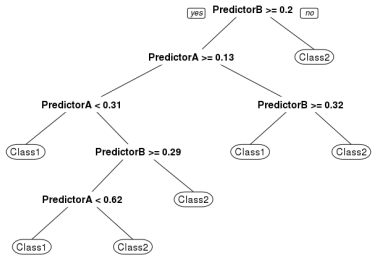
Trends

- Deeper and deeper networks! (GoogLeNet / Residual Neural Network)
- More computational power to learn a better representation.

Sources: Google / Microsoft

Outline

Classification And Regression Trees

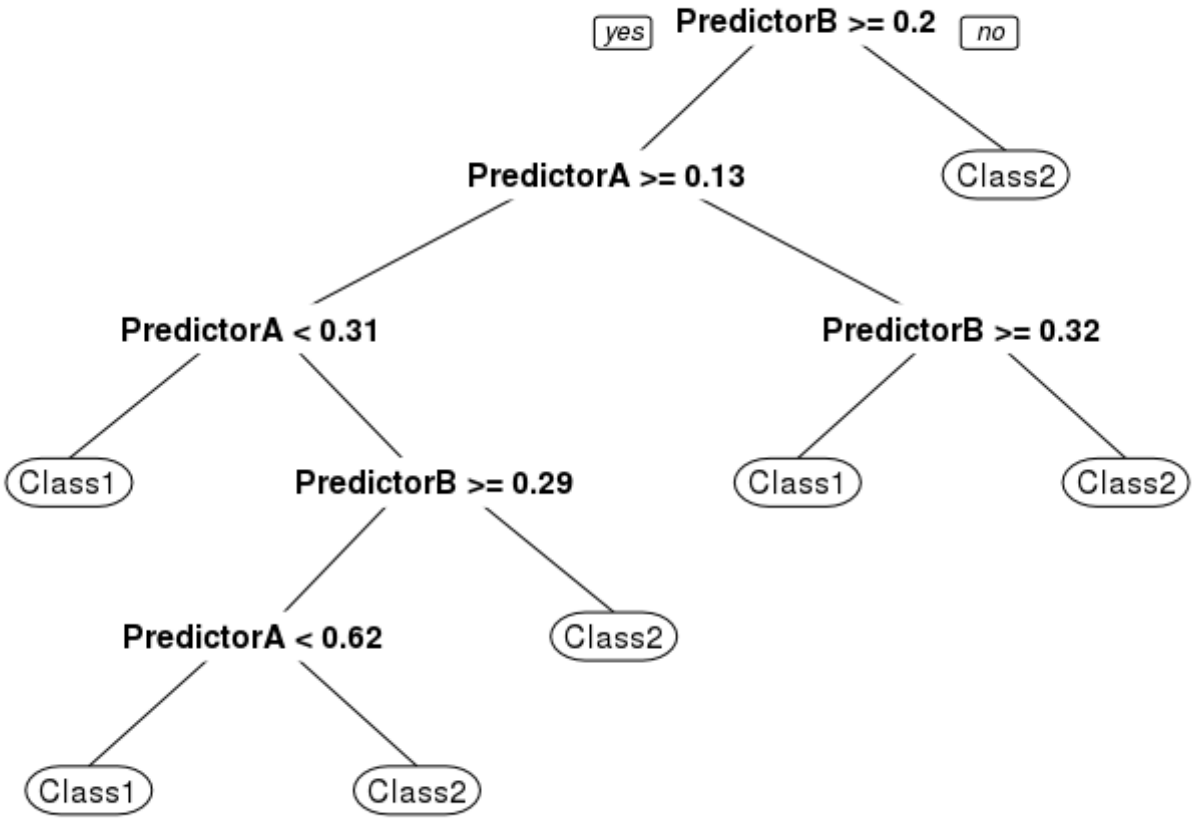


Tree principle (CART by Breiman (85) / ID3 by Quinlan (86))

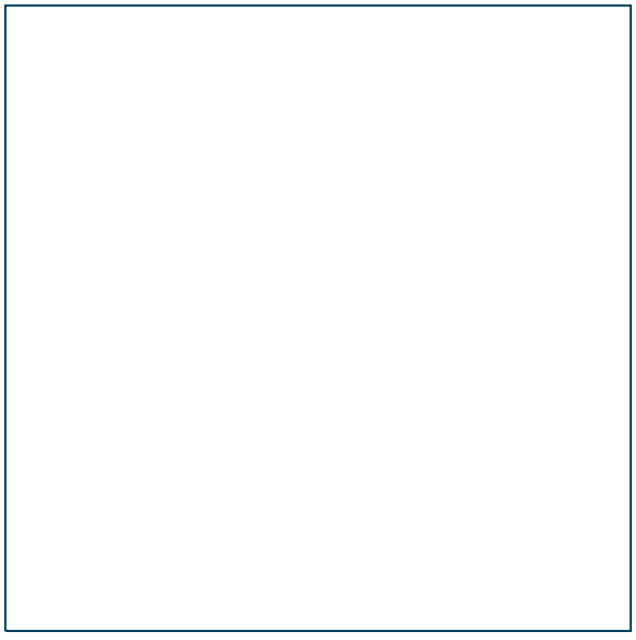
- Construction of a recursive partition through a tree structured set of questions (splits around a given value of a variable)
- For a given partition, probabilistic approach **and** optimization approach yields the same classifier!
- A simple majority vote/averaging in each leaf

- Quality of the prediction depends on the tree (the partition).
- **Intuitively:**
  - small leaves lead to low bias but large variance
  - large leaves lead to large bias but low variance...
- **Issue:** Minim. of the (penalized) empirical error is NP hard!
- Practical tree construction are all based on two steps:
  - a top-down step in which branches are created (branching)
  - a bottom-up in which branches are removed (pruning)

CART


yes
PredictorB >= 0.2
no
PredictorA >= 0.13
Class2
PredictorA < 0.31
PredictorB >= 0.32
Class1
PredictorB >= 0.29
Class1
Class2
PredictorA < 0.62
Class2
Class1
Class2

Branching

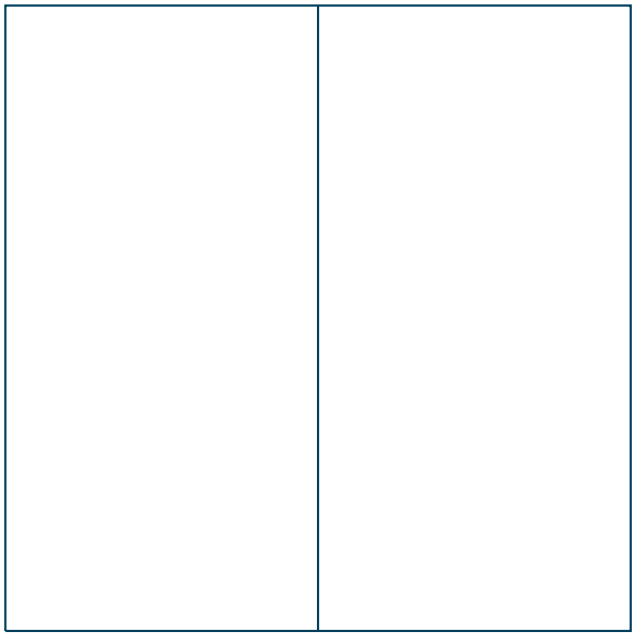


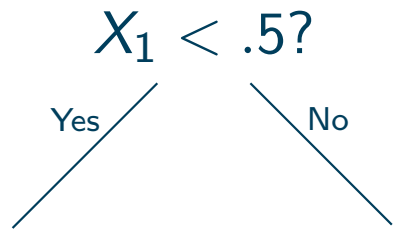
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



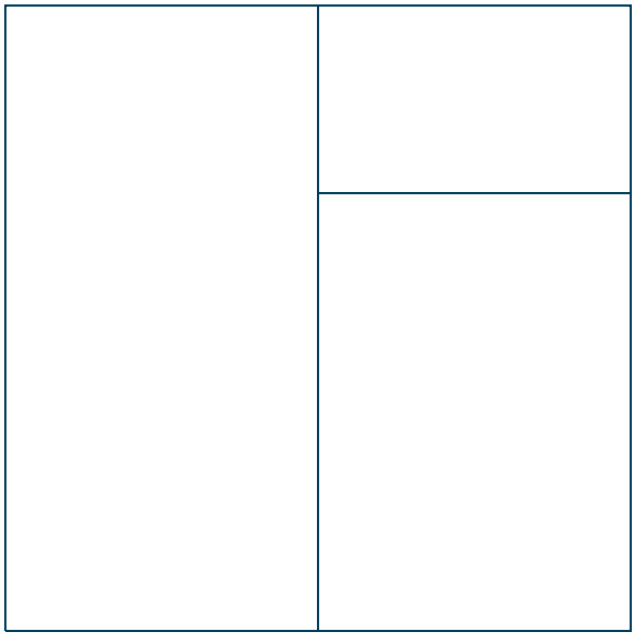


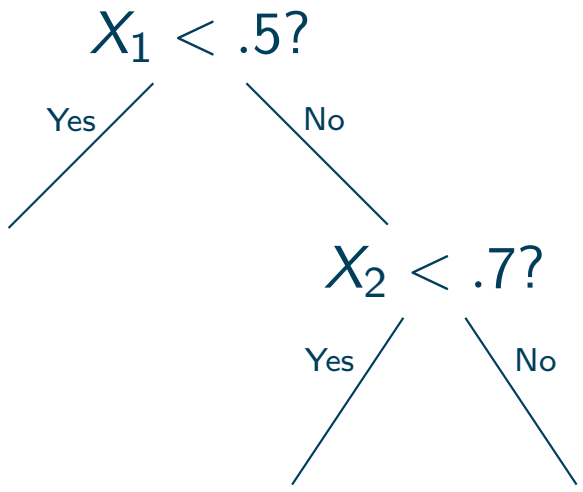
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



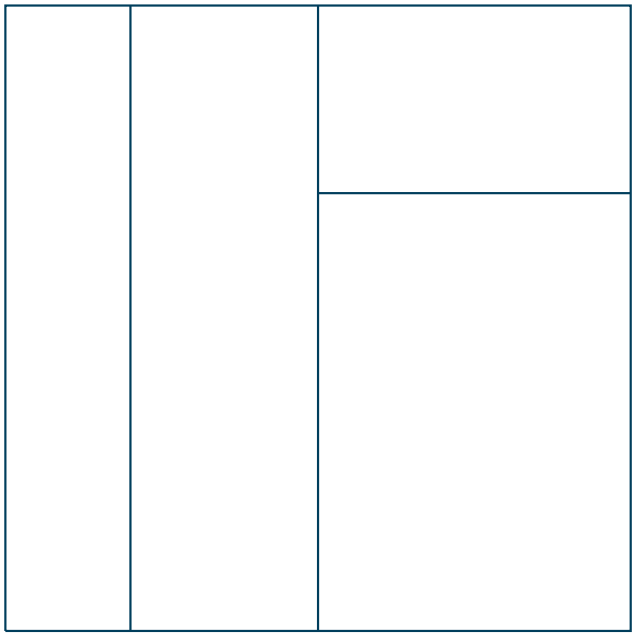


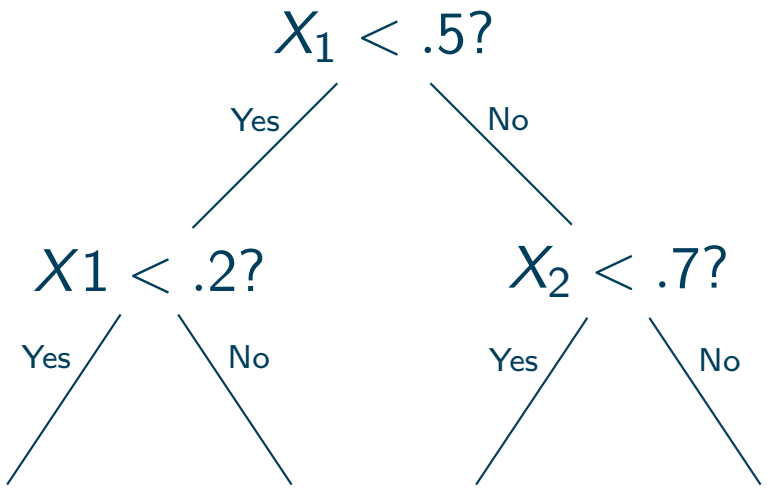
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching





Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching

Various definition of *homogeneous*

- **CART:** empirical loss based criterion (least squares/prediction error)

$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} \ell(y_i, y(R)) + \sum_{\underline{x}_i \in \overline{R}} \ell(y_i, y(\overline{R}))$$

- **CART:** Gini index (Classification)

$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} p(R)(1 - p(R)) + \sum_{\underline{x}_i \in \overline{R}} p(\overline{R})(1 - p(\overline{R}))$$

- **C4.5:** entropy based criterion (Information Theory)

$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} H(R) + \sum_{\underline{x}_i \in \overline{R}} H(\overline{R})$$

- CART with Gini is probably the most used technique...
- Other criterion based on χ^2 homogeneity or based on different local predictors (generalized linear models...)

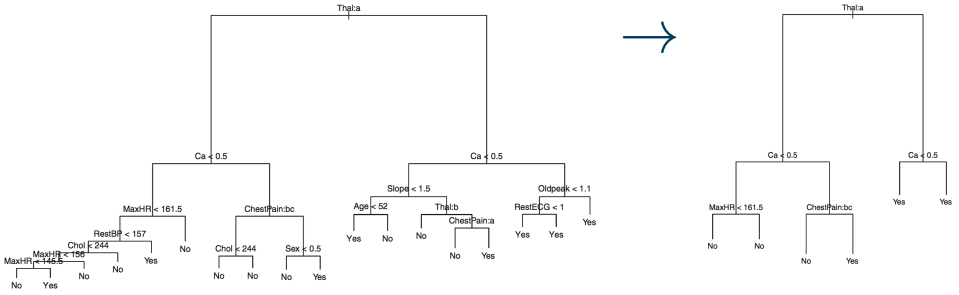
Branching

Choice of the split in a given region

- Compute the criterion for **all features and all possible splitting points** (necessarily among the data values in the region)
- Choose the one **minimizing** the criterion

- **Variations:** split at all categories of a categorical variables using a clever category ordering (ID3), split at a fixed position (median/mean)
- **Stopping rules:**
  - when a leaf/region contains less than a prescribed number of observations
  - when the region is sufficiently homogeneous...
- May lead to a quite complex tree: over-fitting possible!
- Additional pruning often use.

Pruning



- **Model selection** within the (rooted) subtrees of previous tree!
- Number of subtrees can be quite large but the tree structure allows to find the best model efficiently.

Key idea

- The predictor in a leaf depends only on the values in this leaf.
- **Efficient bottom-up (dynamic programming) algorithm** if the criterion used satisfies an additive property

$$C(\mathcal{T}) = \sum_{\mathcal{L} \in \mathcal{T}} c(\mathcal{L})$$

- Example: AIC / CV.

Pruning

Examples of criterion satisfying this assumptions

- AIC type criterion:

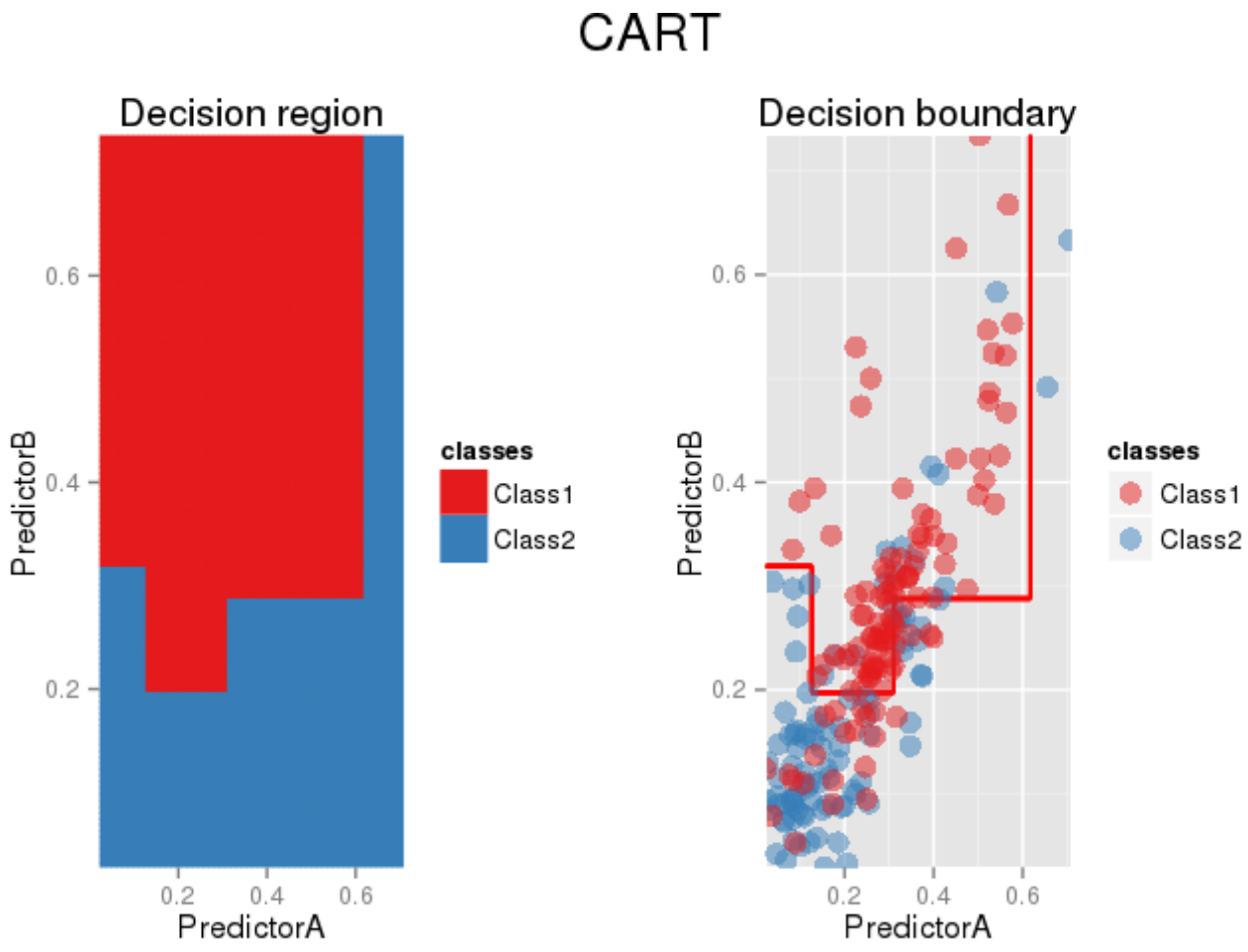
$$\sum_{i=1}^{n} \ell'(y_i, f_{\mathcal{L}(\underline{x}_i)}(\underline{x}_i) + \lambda|\mathcal{T}| = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}_i \in \mathcal{L}} \ell'(y_i, f_{\mathcal{L}}(\underline{x}_i) + \lambda \right)$$

- Simple cross-Validation (with $(\underline{x}'_i, y'_i)$ a different dataset):

$$\sum_{i=1}^{n'} \ell'(y'_i, f_{\mathcal{L}}(\underline{x}'_i) = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}'_i \in \mathcal{L}} \ell'(y'_i, f_{\mathcal{L}}(\underline{x}'_i) \right)$$

- Limits over-fitting for a single tree.
- **Rk:** almost never used when combining several trees...

CART


CART
Decision region
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2
Decision boundary
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2

CART: Pros and Cons

Pros

- Leads to a easily interpretable model
- Fast computation of the prediction
- Easily deals with categorical features

Cons

- Greedy optimization
- Hard decision boundaries
- Lack of stability

Ensemble methods

- Lack of robustness for single trees.
- How to combine trees?

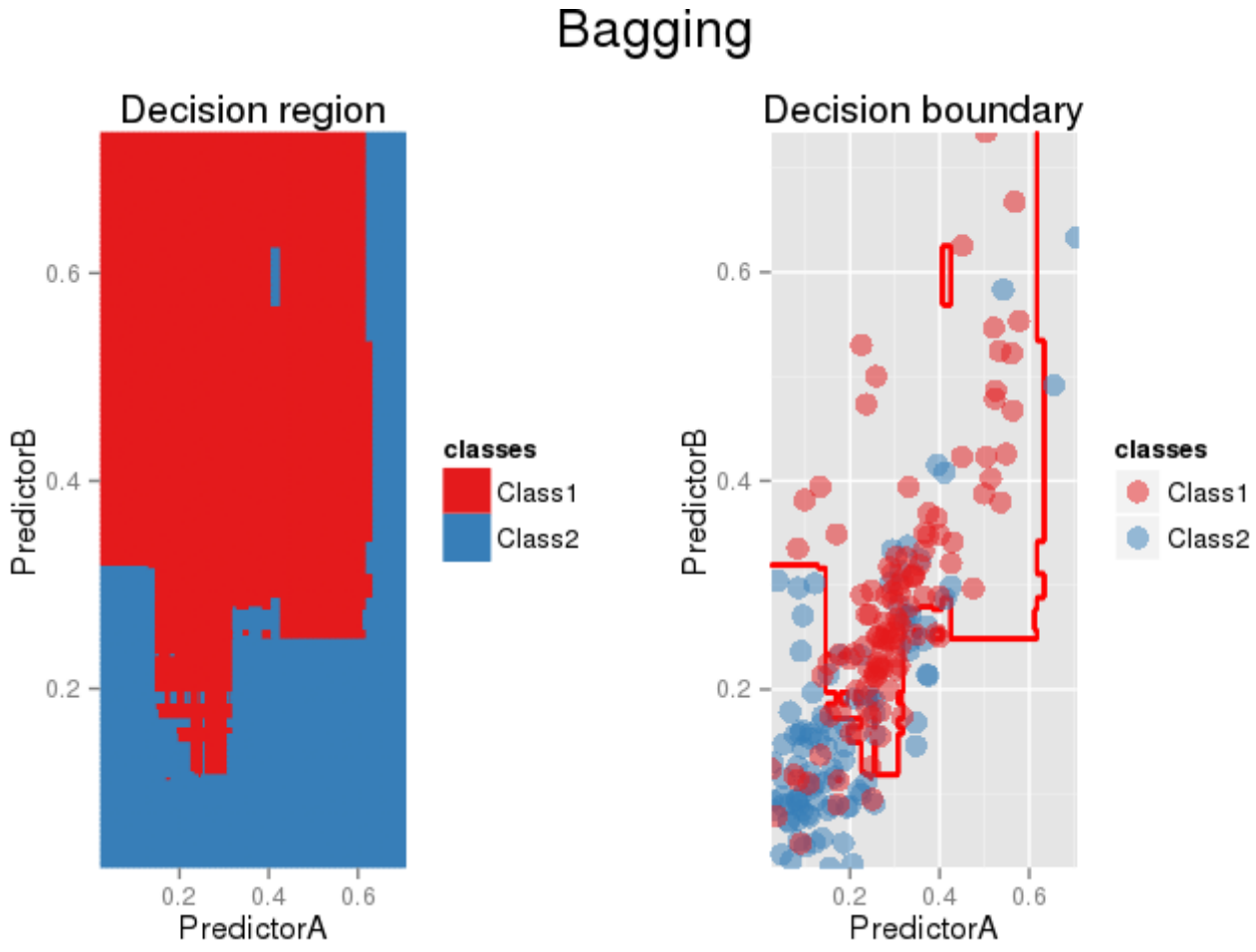
Parallel construction

- Construct several trees from bootstrapped samples and average the responses **(Bagging)**
- Add more randomness in the tree construction **(Random Forests)**

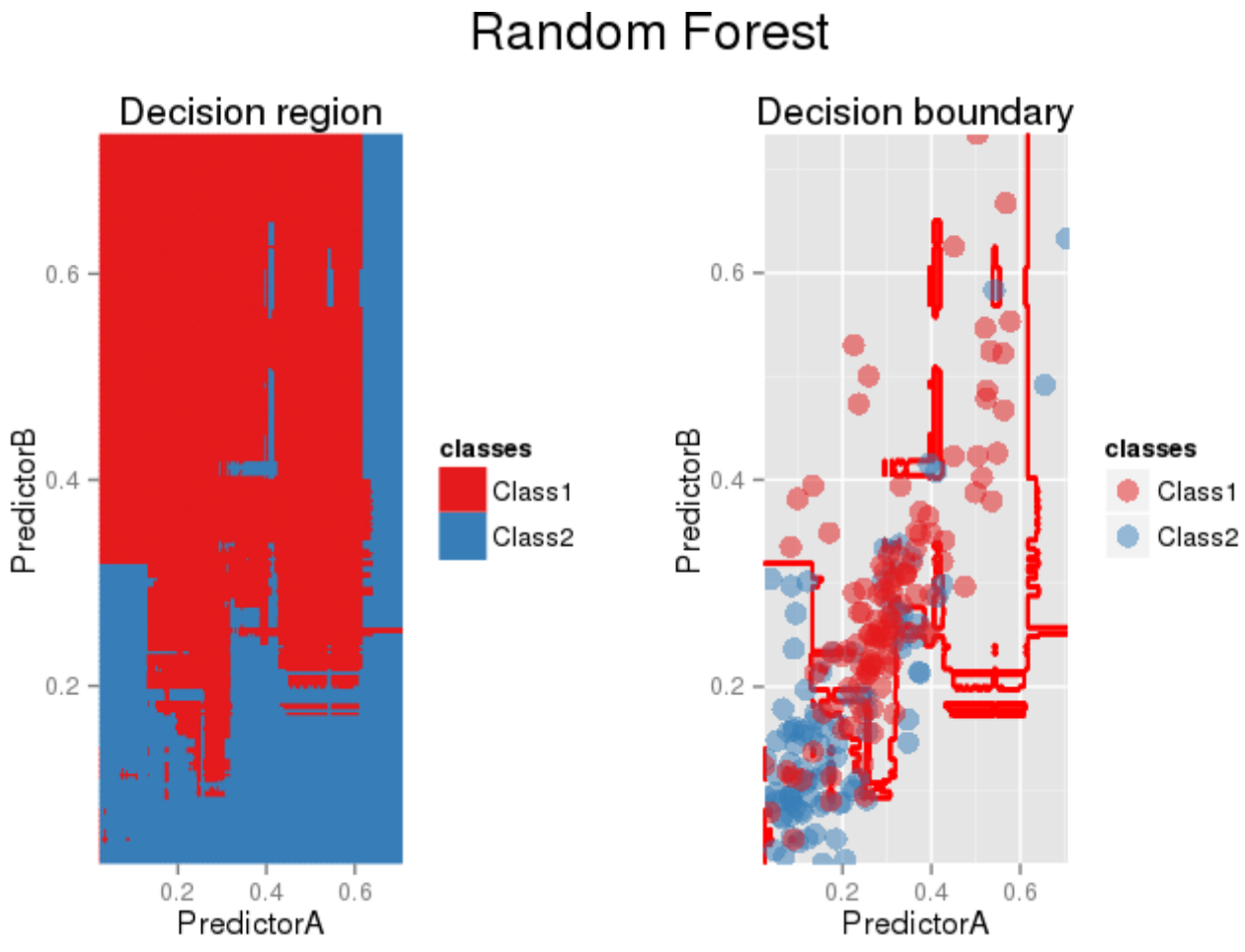
Sequential construction

- Construct a sequence of trees by reweighting sequentially the samples according to their difficulties **(AdaBoost)**
- Reinterpretation as a stagewise additive model **(Boosting)**

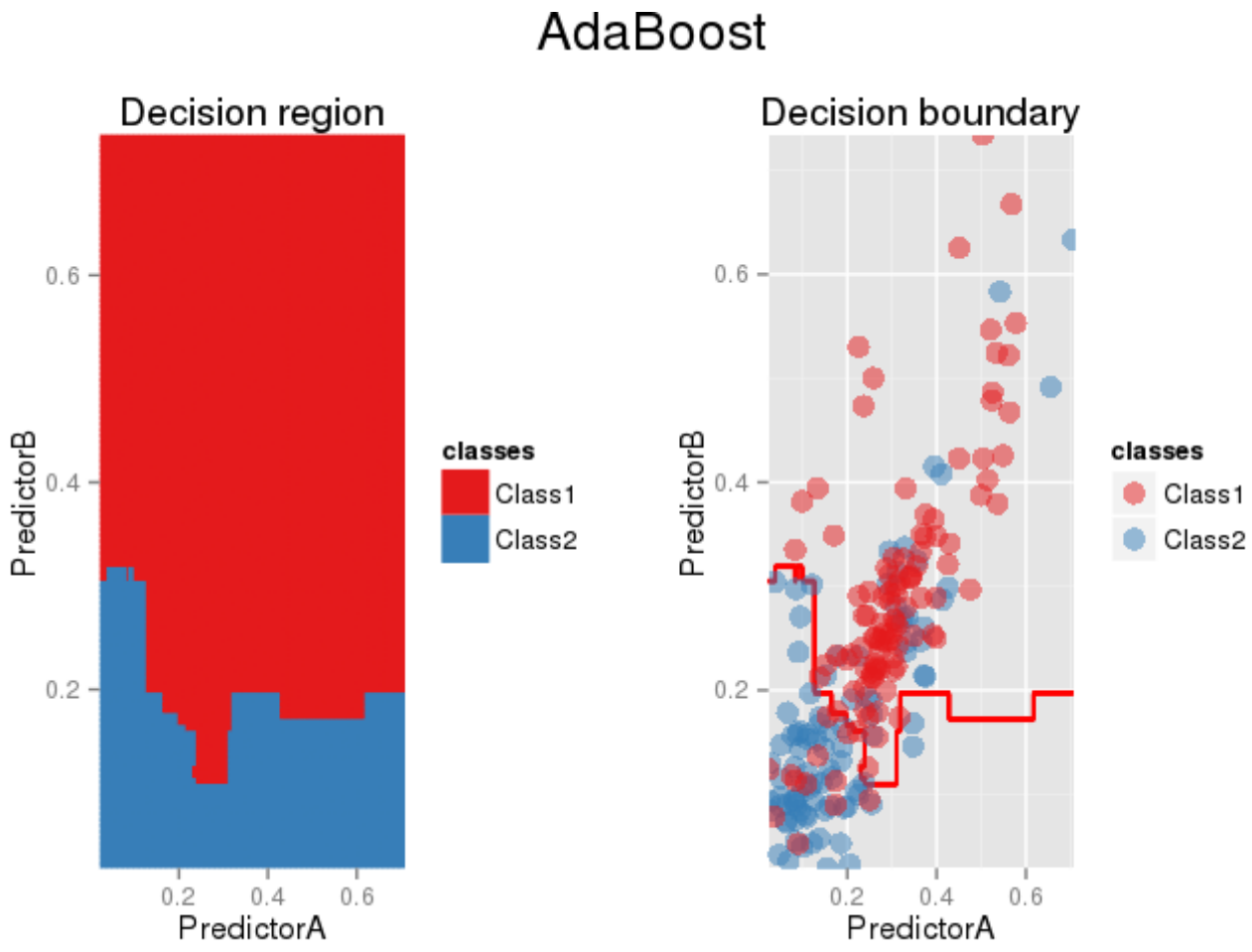
Ensemble methods



Ensemble methods



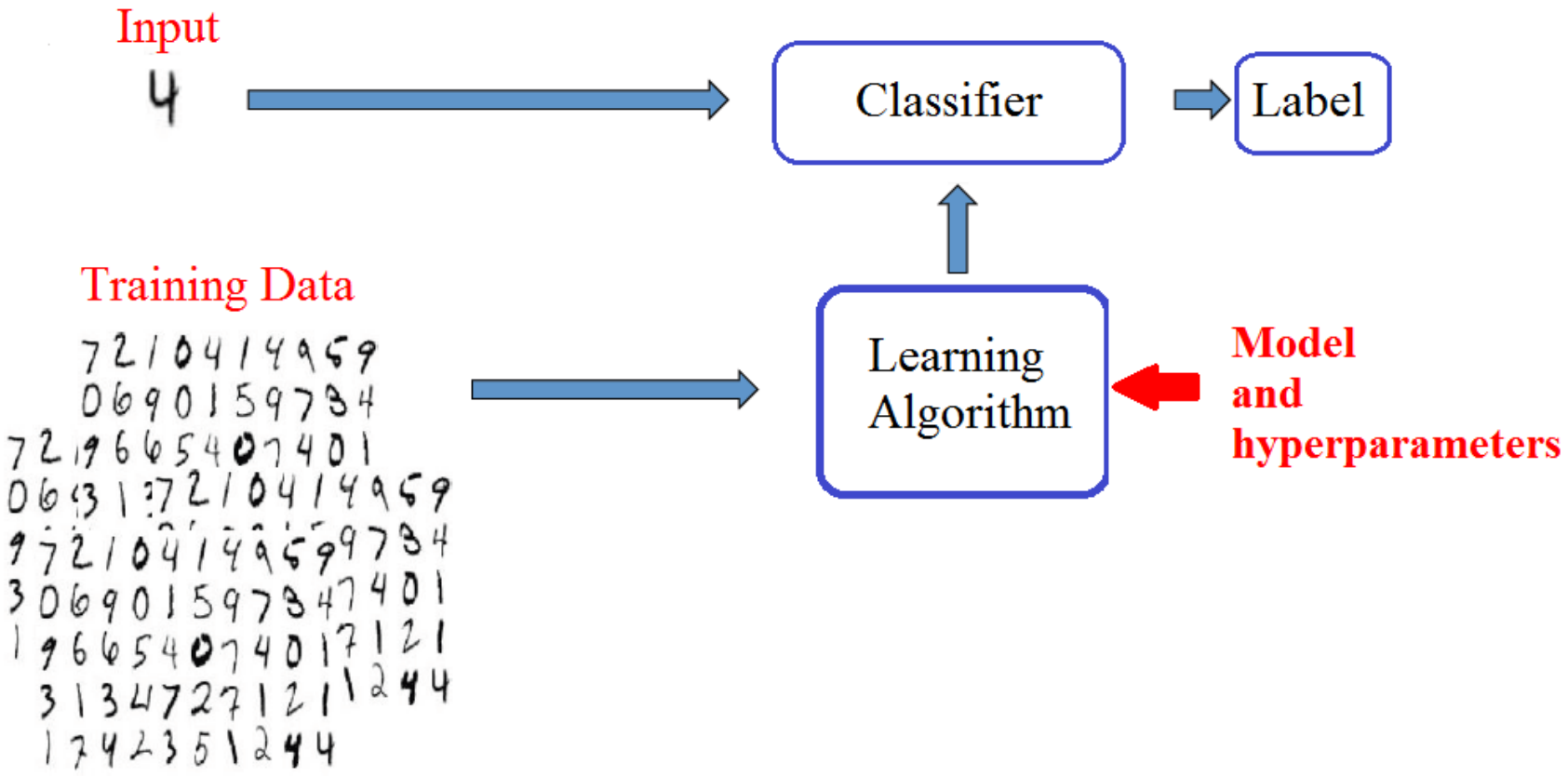
Ensemble methods



Outline

Outline

Model and Hyperparameters



Source: Unknown

Logistic Regression

- Ideal solution:

$$f^*(\underline{x}) = \text{argmax}\, \mathbb{P}\left(Y|\underline{x}\right)$$

Logistic

- Model $Y|\underline{X}$ with a logistic model.
- Estimate its parameters with a Maximum Likelihood approach.
- Plug the estimate in the Bayes classifier.

- Model hyperparameters:
  - Features
  - Parametric model...

Generative Modeling

- Ideal solution:

$$f^*(\underline{x}) = \text{argmax}\, \mathbb{P}\left(Y|\underline{x}\right)$$

Generative Modeling

- Estimate $\underline{X}|Y$ with a density estimator as well as $\mathbb{P}\left(Y\right)$
- Deduce using the Bayes formula an estimate $Y|\underline{X}$.
- Plug the estimate in the Bayes classifier.

- Model hyperparameters:
  - Features
  - Generative model

Kernel Method

- Ideal solution:

$$f^*(\underline{x}) = \text{argmax}\, \mathbb{P}\left(Y|\underline{x}\right)$$

Kernel methods

- Estimate $Y|\underline{X}$ with a kernel conditional density estimator.
- Plug the estimate in the Bayes classifier.

- Model hyperparameters:
  - Features
  - Bandwidth and kernel

Logistic Regression

- Ideal solution:

$$f^* = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \, \mathbb{E}\left[\ell^{0/1}(Y, f(X))\right]$$

Logistic

- Replace $\ell^{0/1}$ by the logistic loss.
- Add a penalty $\lambda \|f\|_p$
- Compute the minimizer.

- Model hyperparameters:
  - Features
  - Penalty and regularization parameter.

SVM

- Ideal solution:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{S}} \mathbb{E}\left[\ell^{0/1}(Y, f(X))\right]$$

SVM

- Replace the expectation by its empirical counterpart.
- Replace $\ell^{0/1}(y, f) = \mathbf{1}_{y=f}$ by $\ell'(y, f) = (1 - yf)_+$.
- Add a penalty $\lambda \|f\|_{\mathcal{S}}^2$.
- Compute the minimizer.

- Model hyperparameters:
  - Features
  - $\mathcal{S}$ RKHS structure: features mapping and metric
  - Regularization parameters λ

(Deep) Neural Networks

- Ideal solution:

$$f^* = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \, \mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right]$$

NN

- Neuron: $\underline{x} \mapsto \sigma(\underline{x}^\top \beta + \beta^{(0)})$
- Neural Network: Convolution system of neurons.
- Replace $\ell^{0/1}(y, f)$ by a smooth/convex loss.
- Minimize the empirical loss using the backprop algorithm (gradient descent)

- Model hyperparameters:
  - Features
  - Net architecture, activation function
  - Initialization strategy
  - Optimization strategy (and regularization strategy)

Tree and Boosting

- Ideal solution:
  $f^*(\underline{x}) = \text{argmax}\, \mathbb{P}(Y|\underline{x})$ and $f^* = \underset{f \in \mathcal{S}}{\text{argmin}}\, \mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right]$

Single tree

- Greedy Partition construction.
- Local conditional density estimation / loss minimization.
- Suboptimal tree optimization through a relaxed criterion

Bagging/Random Forest

- Averaging of several predictors (probabilistic point of view)

Boosting

- Best interpretation as a minimization of the exponential loss $\ell(y, f) = e^{-yf}$ (optimization point of view)

Outline

Feature Design

Transformed Representation

- From $\underline{X}$ to $\Phi(\underline{X})$!
- New description of $\underline{X}$ leads to a different **linear** model:

$$f_\beta(\underline{X}) = \Phi(\underline{X})^\top \beta$$

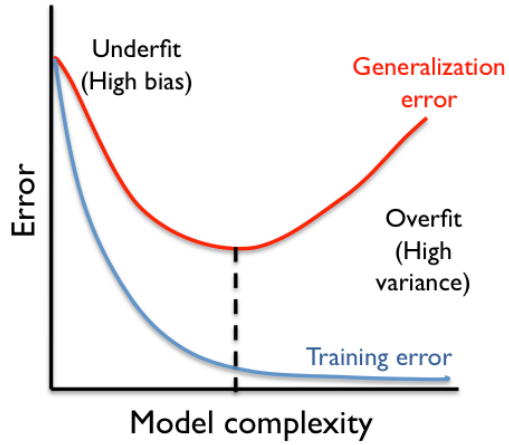
Feature Design

- Art of choosing Φ.
- Examples:
  - Renormalization, (domain specific) transform
  - Basis decomposition
  - Interaction between different variables...

- Need to select a good transformation.

Outline

Over-fitting Issue



Error behaviour

- Learning/training error (error made on the learning/training set) decays when the complexity of the model increases.
- Quite different behavior when the error is computed on new observations (generalization error).

- Overfit for complex models: parameters learned are too specific to the learning set!
- General situation! (Think of polynomial fit...)
- Need to use a different criterion than the training error!

Source: Unknown

Cross Validation and Penalization

Two Approaches

- **Cross validation:** Very efficient (and almost always used in practice!) but slightly biased as it target uses only a fraction of the data.
- **Bias correction approach:** use empirical loss criterion but *correct* with a term increasing with the complexity of $\mathcal{S}$

$$R_n(\widehat{f}_{\mathcal{S}}) \to R_n(\widehat{f}_{\mathcal{S}}) + \text{pen}(\mathcal{S})$$

and choose the model with the smallest corrected risk.

Which loss to use?

- The loss used in the risk: most natural!
- The loss used to estimate $\widehat{\theta}$: penalized estimation!

Ensemble methods

- How to combine several predictors (models)?
- Two strategies: mixture or sequential

Mixture

- Model averaging
- Data dependent model averaging (learn mixture weights)

Stagewise

- Modify learning procedure according to current results.
- Boosting, Cascade...

Outline

Outline

Empirical Risk Minimization

Empirical Risk Minimizer (ERM)

- For any loss ℓ and function class $\mathcal{S}$,

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i) = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \mathcal{R}_n(f)$$

- Key property:

$$\mathcal{R}_n(\widehat{f}) \leq \mathcal{R}_n(f), \forall f \in \mathcal{S}$$

- **Minimization not always tractable in practice!**
- Focus on the $\ell^{0/1}$ case:
  - only algorithm is to try all the functions,
  - not feasible is there are many functions
  - but interesting hindsight!

Outline

ERM and PAC Analysis

- Theoretical control of the random (error estimation) term:
$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^{\star}_{\mathcal{S}})$$

Probably Almost Correct Analysis

- **Theoretical guarantee** that with probability larger than $1-\delta$,
$$\mathbb{P}\left(\mathcal{R}(\widehat{f}) - \mathcal{R}(f^{\star}_{\mathcal{S}}) \leq \epsilon_{\mathcal{S}}(\delta)\right) \geq 1-\delta$$
for a suitable $\epsilon_{\mathcal{S}}(\delta) \geq 0$.
- Implies:
  - $\mathbb{P}\left(\mathcal{R}(\widehat{f}) - \mathcal{R}(f^{\star}) \leq \mathcal{R}(f^{\star}_{\mathcal{S}}) - \mathcal{R}(f^{\star}) + \epsilon_{\mathcal{S}}(\delta)\right) \geq 1-\delta$
  - $\mathbb{E}\left[\mathcal{R}(\widehat{f}) - \mathcal{R}(f^{\star}_{\mathcal{S}})\right] \leq \int_0^{+\infty} \delta_{\mathcal{S}}(\epsilon) d\epsilon$

- The result should hold without any assumption on the law ***P***!

A General Decomposition

- By construction:

$$\begin{aligned}\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) &= \mathcal{R}(\widehat{f}) - \mathcal{R}_n(\widehat{f}) + \mathcal{R}_n(\widehat{f}) - \mathcal{R}_n(f_{\mathcal{S}}^{\star}) + \mathcal{R}_n(f_{\mathcal{S}}^{\star}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \\ &\leq \mathcal{R}(\widehat{f}) - \mathcal{R}_n(\widehat{f}) + \mathcal{R}_n(f_{\mathcal{S}}^{\star}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \\ &\leq \Big(\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star})\Big) - \Big(\mathcal{R}_n(\widehat{f}) - \mathcal{R}_n(f_{\mathcal{S}}^{\star})\Big)\end{aligned}$$

Four possible upperbounds

- $\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sup_{f \in \mathcal{S}} \left((\mathcal{R}(f) - \mathcal{R}(f_{\mathcal{S}}^{\star})) - (\mathcal{R}_n(f) - \mathcal{R}_n(f_{\mathcal{S}}^{\star})) \right)$
- $\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sup_{f \in \mathcal{S}} (\mathcal{R}(f) - \mathcal{R}_n(f)) + (\mathcal{R}_n(f_{\mathcal{S}}^{\star}) - \mathcal{R}(f_{\mathcal{S}}^{\star}))$
- $\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sup_{f \in \mathcal{S}} (\mathcal{R}(f) - \mathcal{R}_n(f)) + \sup_{f \in \mathcal{S}} (\mathcal{R}_n(f) - \mathcal{R}(f))$
- $\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq 2 \sup_{f \in \mathcal{S}} |\mathcal{R}(f) - \mathcal{R}_n(f)|$

- Supremum of centered random variables!
- **Key:** Concentration of each variable...

Error Bounds

- By construction, for any $f' \in \mathcal{S}$,
$$\mathcal{R}(f') = \mathcal{R}_n(f') + (\mathcal{R}(f') - \mathcal{R}_n(f'))$$

A uniform upper bound for the error

- Simultaneously $\forall f' \in \mathcal{S}$,
$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sup_{f \in \mathcal{S}} (\mathcal{R}(f) - \mathcal{R}_n(f))$$

- Supremum of centered random variables!
- **Key:** Concentration of each variable...
- Can be interpreted as a justification of the ERM!

Outline

Concentration of the Empirical Loss

- Empirical loss:

$$\mathcal{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

Properties

- $\ell^{0/1}(Y_i, f(\underline{X}_i))$ are i.i.d. random variables in $[0, 1]$.

Concentration

$$\mathbb{P}\left(\mathcal{R}(f) - \mathcal{R}_n(f) \leq \epsilon\right) \geq 1 - e^{-2n\epsilon^2}$$

$$\mathbb{P}\left(\mathcal{R}_n(f) - \mathcal{R}(f) \leq \epsilon\right) \geq 1 - e^{-2n\epsilon^2}$$

$$\mathbb{P}\left(|\mathcal{R}_n(f) - \mathcal{R}(f)| \leq \epsilon\right) \geq 1 - 2e^{-2n\epsilon^2}$$

- Concentration of sum of bounded independent variables!
- Hoeffding theorem.

Hoeffding

Theorem

- Let Z_i be a sequence of independent centered random variables supported in $[a_i, b_i]$ then

$$\mathbb{P}\left(\sum_{i=1}^{n} Z_i \geq \epsilon\right) \leq e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n}(b_i-a_i)^2}}$$

- Proof ingredients:
  - Chernov bounds:

$$\mathbb{P}\left(\sum_{i=1}^{n} Z_i \geq \epsilon\right) \leq \frac{\mathbb{E}\left[e^{\lambda \sum_{i=1}^{n} Z_i}\right]}{e^{\lambda\epsilon}} \leq \frac{\prod_{i=1}^{n} \mathbb{E}\left[e^{\lambda Z_i}\right]}{e^{\lambda\epsilon}}$$

  - Exponential moment bounds: $\mathbb{E}\left[e^{\lambda Z_i}\right] \leq e^{\frac{\lambda^2(b_i-a_i)^2}{8}}$
  - Optimization in λ
- **Prop:**

$$\mathbb{E}\left[e^{\lambda \sum_{i=1}^{n} Z_i}\right] \leq e^{\frac{\lambda^2 \sum_{i=1}^{n}(b_i-a_i)^2}{8}}.$$

Hoeffding Inequality

Theorem

- Let Z_i be a sequence of independent centered random variables supported in $[a_i, b_i]$ then

$$\mathbb{P}\left(\sum_{i=1}^{n} Z_i \geq \epsilon\right) \leq e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n}(b_i-a_i)^2}}$$

- $Z_i = \frac{1}{n}\left(\mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right] - \ell^{0/1}(Y_i, f(\underline{X}_i))\right)$
- $\mathbb{E}[Z_i] = 0$ and
  $Z_i \in [\frac{1}{n}\left(\mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right] - 1\right), \frac{1}{n}\mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right]]$
- Concentration:

$$\mathbb{P}\left(\mathcal{R}(f) - \mathcal{R}_n(f) \geq \epsilon\right) \leq e^{-2n\epsilon^2}$$

- By symmetry,

$$\mathbb{P}\left(\mathcal{R}_n(f) - \mathcal{R}(f) \geq \epsilon\right) \leq e^{-2n\epsilon^2}$$

- Combining the two yields

$$\mathbb{P}\left(|\mathcal{R}_n(f) - \mathcal{R}(f)| \geq \epsilon\right) \leq 2e^{-2n\epsilon^2}$$

Finite Class Case

Concentration

- If $\mathcal{S}$ is finite of cardinality $|\mathcal{S}|$,

$$\mathbb{P}\left(\sup_f \left(\mathcal{R}(f) - \mathcal{R}_n(f)\right) \leq \sqrt{\frac{\log|\mathcal{S}| + \log(1/\delta)}{2n}}\right) \geq 1 - \delta$$

$$\mathbb{P}\left(\sup_f \left|\mathcal{R}_n(f) - \mathcal{R}(f)\right| \leq \sqrt{\frac{\log|\mathcal{S}| + \log(1/\delta)}{2n}}\right) \geq 1 - 2\delta$$

- Control of the supremum by a quantity depending on the cardinality and the probability parameter δ.
- Simple combination of Hoeffding and a union bound.

Finite Class Case

PAC Bounds

- If $\mathcal{S}$ is finite of cardinality $|\mathcal{S}|$, with proba greater than $1-2\delta$

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f^\star_{\mathcal{S}}) \leq \sqrt{\frac{\log|\mathcal{S}| + \log(1/\delta)}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

$$\leq 2\sqrt{\frac{\log|\mathcal{S}| + \log(1/\delta)}{2n}}$$

- If $\mathcal{S}$ is finite of cardinality $|\mathcal{S}|$, with proba greater than $1-\delta$, simultaneously $\forall f' \in \mathcal{S}$,

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{\log|\mathcal{S}| + \log(1/\delta)}{2n}}$$

$$\leq \mathcal{R}_n(f') + \sqrt{\frac{\log|\mathcal{S}|}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

Finite Class Case

PAC Bounds

- If $\mathcal{S}$ is finite of cardinality $|\mathcal{S}|$, with proba greater than $1-2\delta$

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f^\star_{\mathcal{S}}) \leq \sqrt{\frac{\log|\mathcal{S}|}{2n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

- If $\mathcal{S}$ is finite of cardinality $|\mathcal{S}|$, with proba greater than $1-\delta$, simultaneously $\forall f' \in \mathcal{S}$,

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{\log|\mathcal{S}|}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- Risk increases with the cardinality of $\mathcal{S}$.
- Similar issue in cross-validation!
- No direct extension for an infinite $\mathcal{S}$...

Outline

Concentration of the Supremum of Empirical Losses

- Supremum of Empirical losses:

$$\Delta_n(\mathcal{S})(\underline{X}_1, \ldots, \underline{X}_n) = \sup_{f \in \mathcal{S}} \mathcal{R}(f) - \mathcal{R}_n(f)$$

$$= \sup_{f \in \mathcal{S}} \left(\mathbb{E}\left[\ell^{0/1}(Y, f(\underline{X}))\right] - \frac{1}{n}\sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i)) \right)$$

Properties

- Bounded difference:

$$|\Delta_n(\mathcal{S})(\underline{X}_1, \ldots, \underline{X}_j, \ldots \underline{X}_n) - \Delta_n(\mathcal{S})(\underline{X}_1, \ldots \underline{X}'_j, \ldots, \underline{X}_n)| \leq 1/n$$

Concentration

$$\mathbb{P}\left(\Delta_n(\mathcal{S}) - \mathbb{E}\left[\Delta_n(\mathcal{S})\right] \leq \epsilon\right) \geq 1 - e^{-2n\epsilon^2}$$

- Concentration of bounded difference function.
- Generalization of Hoeffding theorem: McDiarmid Theorem.

McDiarmid Inequality

Bounded difference function

- $g : \mathcal{X}^n \to \mathbb{R}$ is a bounded difference function if it exist c_i such that

$$\forall (\underline{X}_i)_{i=1}^n, (\underline{X}'_i)_{i=1}^n \in \mathbb{R},$$

$$\left| g(\underline{X}_1, \ldots, \underline{X}_j, \ldots, \underline{X}_n) - g(\underline{X}_1, \ldots, \underline{X}'_j, \ldots, \underline{X}_n) \right| \leq c_i$$

Theorem

- If g is a bounded difference function and $\underline{X}_i$ are independent random variables then

$$\mathbb{P}\left(g(\underline{X}_1, \ldots, \underline{X}_n) - \mathbb{E}\left[g(\underline{X}_1, \ldots, \underline{X}_n)\right] \geq \epsilon\right) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}}$$

$$\mathbb{P}\left(\mathbb{E}\left[g(\underline{X}_1, \ldots, \underline{X}_n)\right] - g(\underline{X}_1, \ldots, \underline{X}_n) \geq \epsilon\right) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}}$$

- Proof ingredients:
  - Chernov bounds
  - Martingale decomposition...

McDiarmid Inequality

Theorem

- If g is a bounded difference function and $\underline{X}_i$ are independent random variables then

$$\mathbb{P}\left(g(\underline{X}_1,\ldots,\underline{X}_n) - \mathbb{E}\left[g(\underline{X}_1,\ldots,\underline{X}_n)\right] \geq \epsilon\right) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}}$$

- Using $g = \Delta_n(\mathcal{S})$ for which $c_i = 1/n$ yields immediately

$$\mathbb{P}\left(\Delta_n(\mathcal{S}) - \mathbb{E}\left[\Delta_n(\mathcal{S})\right] \geq \epsilon\right) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}} = e^{-2n\epsilon^2}$$

- We derive then

$$\mathbb{P}\left(\Delta_n(\mathcal{S}) \geq \mathbb{E}\left[\Delta_n(\mathcal{S})\right] + \epsilon\right) \leq e^{\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}} = e^{-2n\epsilon^2}$$

- It remains to upperbound

$$\mathbb{E}\left[\Delta_n\right] = \mathbb{E}\left[\sup_{f\in\mathcal{S}} \mathcal{R}(f) - \mathcal{R}_n(f)\right]$$

Rademacher Complexity

Theorem

- Let σ_i be a sequence of i.i.d. random symmetric Bernoulli variables (Rademacher variables):

$$\mathbb{E}\left[\sup_{f\in\mathcal{S}}\left(\mathcal{R}(f)-\mathcal{R}_n(f)\right)\right] \leq 2\mathbb{E}\left[\sup_{f\in\mathcal{S}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\ell^{0/1}(Y_i,f(\underline{X}_i))\right]$$

Rademacher complexity

- Let $B\subset\mathbf{R}^n$, the Rademacher complexity of B is defined as

$$R_n(B)=\mathbb{E}\left[\sup_{b\in B}\frac{1}{n}\sum_{i=1}^{n}\sigma_i b_i\right]$$

- Theorem gives an upper bound of the expectation in term of the **average Rademacher complexity of the random set** $B_n(\mathcal{S})=\{(\ell^{0/1}(Y_i,f(\underline{X}_i)))_{i=1}^n, f\in\mathcal{S}\}$.
- **Back to finite setting:** This set is at most of cardinality 2^n!

Finite Set Rademacher Complexity Bound

Theorem

- If B is finite and such that $\forall b \in B, \frac{1}{n}\|b\|_2^2 \leq M^2$, then

$$R_n(B) = \mathbb{E}\left[\sup_{b \in B} \frac{1}{n} \sum_{i=1}^{n} \sigma_i b_i\right] \leq \sqrt{\frac{2M^2 \log |B|}{n}}$$

- If $B = B_n(\mathcal{S}) = \{(\ell^{0/1}(Y_i, f(\underline{X}_i)))_{i=1}^n, f \in \mathcal{S}\}$, we have $M = 1$ and thus

$$R_n(B) \leq \sqrt{\frac{2 \log |B_n(\mathcal{S})|}{n}}$$

- We obtain immediately

$$\mathbb{E}\left[\sup_{f \in \mathcal{S}} (\mathcal{R}(f) - \mathcal{R}_n(f))\right] \leq \mathbb{E}\left[\sqrt{\frac{8 \log |B_n(\mathcal{S})|}{n}}\right].$$

Finite Set Rademacher Complexity Bound

Theorem

- With probability greater than $1 - 2\delta$,

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \mathbb{E}\left[\sqrt{\frac{8 \log |B_n(\mathcal{S})|}{n}}\right] + \sqrt{\frac{2 \log(1/\delta)}{n}}$$

- With probability greater than $1 - \delta$, simultaneously $\forall f' \in \mathcal{S}$

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \mathbb{E}\left[\sqrt{\frac{8 \log |B_n(\mathcal{S})|}{n}}\right] + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- This is a direct consequence of the previous bound.

Finite Set Rademacher Complexity Bound

Corollary

- If $\mathcal{S}$ is finite then with probability greater than $1-2\delta$

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sqrt{\frac{8\log|\mathcal{S}|}{n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

- If $\mathcal{S}$ is finite then with probability greater than $1-\delta$, simultaneously $\forall f' \in \mathcal{S}$

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{8\log|\mathcal{S}|}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- It suffices to notice that

$$|B_n(\mathcal{S})| = |\{(\ell^{0/1}(Y_i, f(\underline{X}_i)))_{i=1}^n, f \in \mathcal{S}\}| \leq |\mathcal{S}|$$

Finite Set Rademacher Complexity Bound

- Same result with Hoeffding but with **better** constants!

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sqrt{\frac{\log|\mathcal{S}|}{2n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{\log|\mathcal{S}|}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- Difference due to the *crude* upperbound of

$$\mathbb{E}\left[\sup_{f\in\mathcal{S}}\left(\mathcal{R}(f) - \mathcal{R}_n(f)\right)\right]$$

- **Why bother?:** We do not have to assume that $\mathcal{S}$ is finite!

$$|B_n(\mathcal{S})| \leq 2^n$$

Outline

Back to the Bound

Theorem

$$\mathbb{E}\left[\sup_{f\in\mathcal{S}}\left(\mathcal{R}(f)-\mathcal{R}_n(f)\right)\right]\leq\mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S})|}{n}}\right]$$

- Key quantity: $\mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S})|}{n}}\right]$
- Hard to control due to its structure!

A first data dependent upperbound

$$\mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S})|}{n}}\right]\leq\sqrt{\frac{8\log\mathbb{E}\left[|B_n(\mathcal{S})|\right]}{n}}\quad\text{(Jensen)}$$

- Depends on the unknown $\boldsymbol{P}$!

Shattering Coefficient

Shattering Coefficient (or Growth Function)

- The shattering coefficient of the class $\mathcal{S}$, $s(\mathcal{S}, n)$, is defined as

$$s(\mathcal{S}, n) = \sup_{((\underline{X}_1, Y_1), \ldots, (\underline{X}_n, Y_n)) \in (\mathcal{X} \times \{-1, 1\})^n} |\{(\ell^{0/1}(Y_i, f(\underline{X}_i)))_{i=1}^n, f \in \mathcal{S}\}|$$

- By construction, $|B_n(\mathcal{S})| \leq s(\mathcal{S}, n) \leq \max(2^n, |\mathcal{S}|)$!

A data independent upperbound

$$\mathbb{E}\left[\sqrt{\frac{8 \log |B_n(\mathcal{S})|}{n}}\right] \leq \sqrt{\frac{8 \log s(\mathcal{S}, n)}{n}}$$

Shattering Coefficient

Theorem

- With probability greater than $1 - 2\delta$,

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sqrt{\frac{8 \log s(\mathcal{S}, n)}{n}} + \sqrt{\frac{2 \log(1/\delta)}{n}}$$

- With probability greater than $1 - \delta$, simultaneously $\forall f' \in \mathcal{S}$,

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{8 \log s(\mathcal{S}, n)}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- Depends only on the class $\mathcal{S}$!

Vapnik-Chervonenkis Dimension

VC Dimension

- The VC dimension d_{VC} of $\mathcal{S}$ is defined as the largest integer d such that

$$s(\mathcal{S}, d) = 2^d$$

- The VC dimension can be infinite!

VC Dimension and Dimension

- **Prop:** If span$(\mathcal{S})$ corresponds to the sign of functions in a linear space of dimension d then $d_{VC} \leq d$.

- VC dimension similar to the usual dimension.

VC Dimension and Sauer Lemma

Sauer Lemma

- If the VC dimension d_{VC} of $\mathcal{S}$ is finite

$$s(\mathcal{S}, n) \leq \begin{cases} 2^n & \text{if } n \leq d_{VC} \\ \left(\frac{en}{d_{VC}}\right)^{d_{VC}} & \text{if } n > d_{VC} \end{cases}$$

- **Cor.:** $\log s(\mathcal{S}, n) \leq d_{VC} \log\left(\frac{en}{d_{VC}}\right)$ if $n > d_{VC}$.

VC Dimension and PAC Bounds

PAC Bounds

- If $\mathcal{S}$ is of VC dimension d_{VC} then if $n > d_{VC}$
- With probability greater than $1 - 2\delta$,

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f_{\mathcal{S}}^{\star}) \leq \sqrt{\frac{8 d_{VC} \log\left(\frac{en}{d_{VC}}\right)}{n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

- With probability greater than $1 - \delta$, simultaneously $\forall f' \in \mathcal{S}$,

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{8 d_{VC} \log\left(\frac{en}{d_{VC}}\right)}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- **Rk:** If $d_{VC} = +\infty$ no uniform PAC bounds exists!

Outline

Countable Collection and Non Uniform PAC Bounds

PAC Bounds

- Let $\pi_f > 0$ such that $\sum_{f \in \mathcal{S}} \pi_f = 1$
- With proba greater than $1 - 2\delta$,

$$\mathcal{R}(\widehat{f}) - \mathcal{R}(f^\star_{\mathcal{S}}) \leq \sqrt{\frac{\log(1/\pi_f)}{2n}} + \sqrt{\frac{2\log(1/\delta)}{n}}$$

- With proba greater than $1 - \delta$, simultaneously $\forall f' \in \mathcal{S}$,

$$\mathcal{R}(f') \leq \mathcal{R}_n(f') + \sqrt{\frac{\log(1/\pi_f)}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- Very similar proof than the uniform one!
- Much more interesting idea when combined with several models...

Models, Non Uniform Risk Bounds and SRM

- Assume we have a countable collection of set $(\mathcal{S}_m)_{m\in\mathcal{M}}$ and let π_m be such that $\sum_{m\in\mathcal{M}} \pi_m = 1$.

Non Uniform Risk Bound

- With probability $1-\delta$, simultaneously for all $m \in \mathcal{M}$ and all $f \in \mathcal{S}_m$,

$$\mathcal{R}(f) \leq \mathcal{R}_n(f) + \mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S}_m)|}{n}}\right] + \sqrt{\frac{\log(1/\pi_m)}{2n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

Structural Risk Minimization

- Choose $\hat{f}$ as the minimizer over $m \in \mathcal{M}$ and $f \in \mathcal{S}_m$ of

$$\mathcal{R}_n(f) + \mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S}_m)|}{n}}\right] + \sqrt{\frac{\log(1/\pi_m)}{2n}}$$

- Mimics the minimization of the integrated risk!

SRM and PAC Bound

PAC Bound

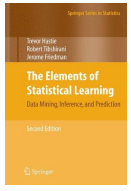
- If $\hat{f}$ is the SRM minimizer then with probability $1-2\delta$,

$$\mathcal{R}(\widehat{f}) \leq \inf_{m\in\mathcal{M}} \inf_{f\in\mathcal{S}_m} \left(\mathcal{R}(f) + \mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S}_m)|}{n}}\right] + \sqrt{\frac{\log(1/\pi_m)}{2n}} \right) + \sqrt{\frac{2\log(1/\delta)}{n}}$$

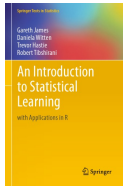
- The SRM minimizer balances the risk $\mathcal{R}(f)$ and the upper bound on the estimation error $\mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S}_m)|}{n}}\right] + \sqrt{\frac{\log(1/\pi_m)}{2n}}$.
- $\mathbb{E}\left[\sqrt{\frac{8\log|B_n(\mathcal{S}_m)|}{n}}\right]$ can be replaced by an upper bound (for instance a VC based one)...

Outline

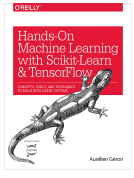
References



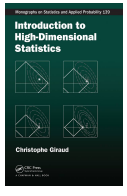
T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009



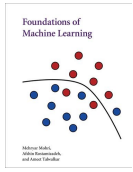
G. James, D. Witten, T. Hastie, and R. Tibshirani.
An Introduction to Statistical Learning with Applications in R.
Springer, 2014



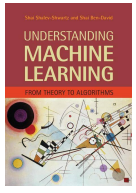
A. Géron.
Hands-On Machine Learning with Scikit-Learn and TensorFlow.
O'Reilly, 2017



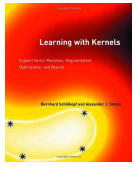
Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014



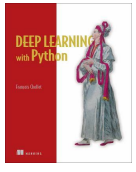
M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012



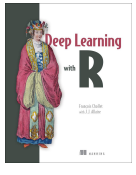
S. Shalev-Shwartz and S. Ben-David.
Understanding Machine Learning.
Cambridge University Press, 2014



B. Schölkopf and A. Smola.
Learning with kernels.
The MIT Press, 2002

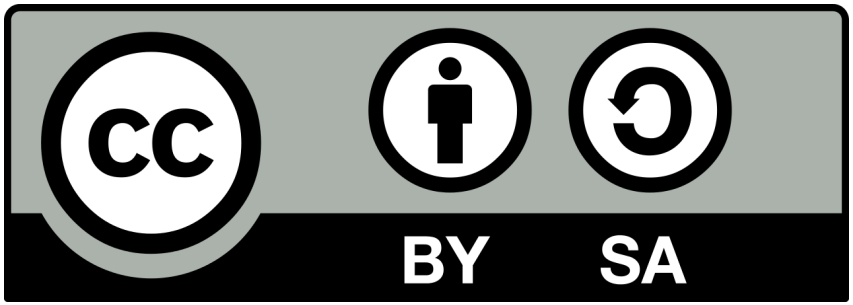


F. Chollet.
Deep Learning with Python.
Manning, 2017



F. Chollet and J.J. Allaire.
Deep Learning with R.
Manning, 2017

Licence and Contributors



Contributors

- S. Boucheron, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Le Pennec, E. Matzner, E. Scornet and X Exed team.