Machine Learning: from Theory to Practice

Statistical Learning and ML Methods

F. d'Alché-Buc and E. Le Pennec

Fall 2016

Doing Data Science

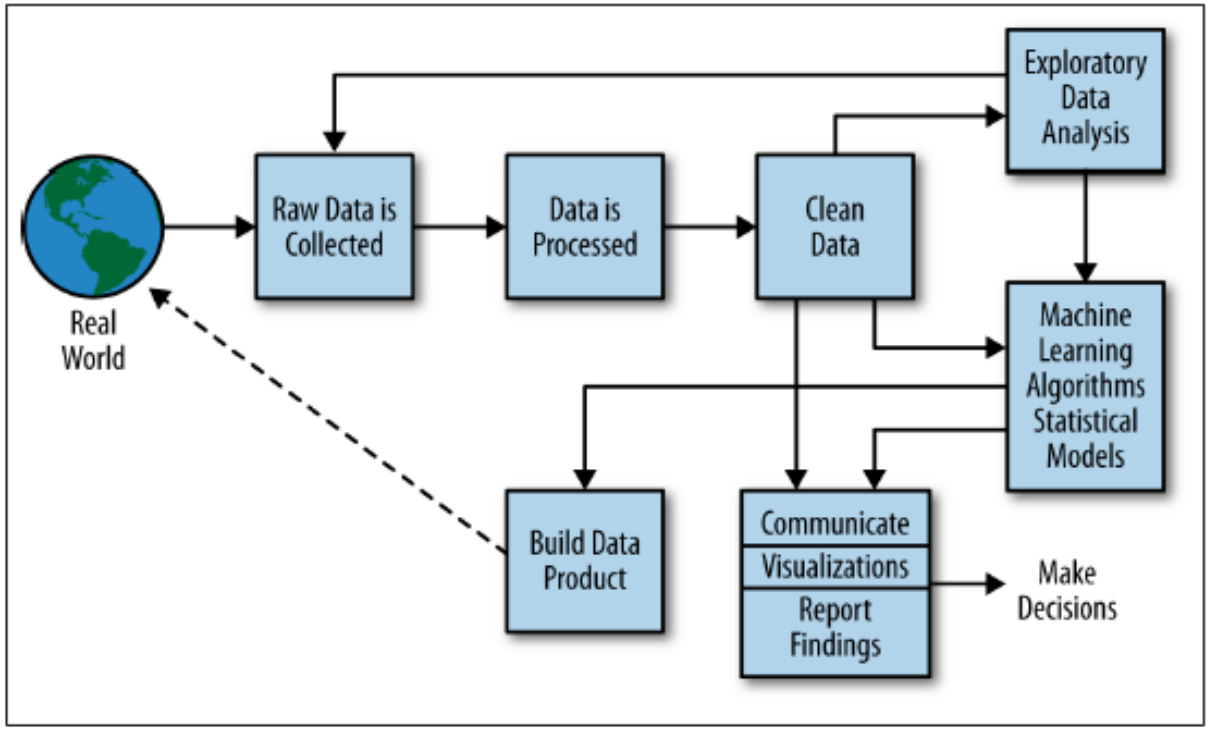
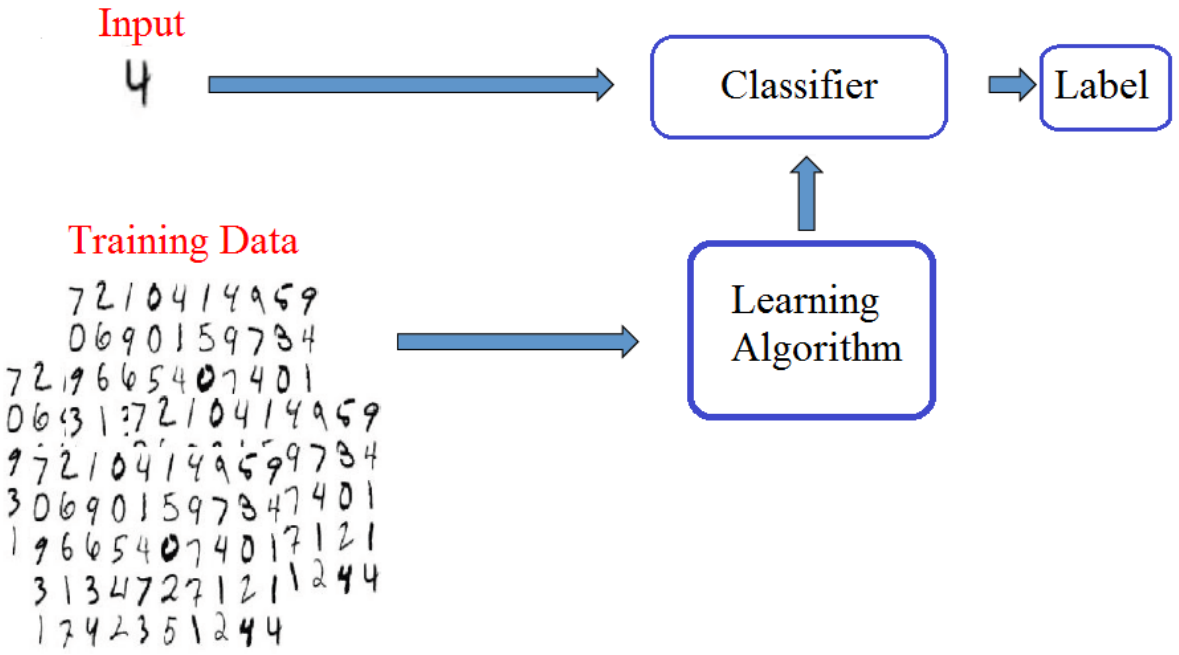


Figure 2-2. The data science process

- Doing Data Science: Straight talk from the frontline.
  - Rachel Schutt, Cathy O'Neil
  - O'Reilly

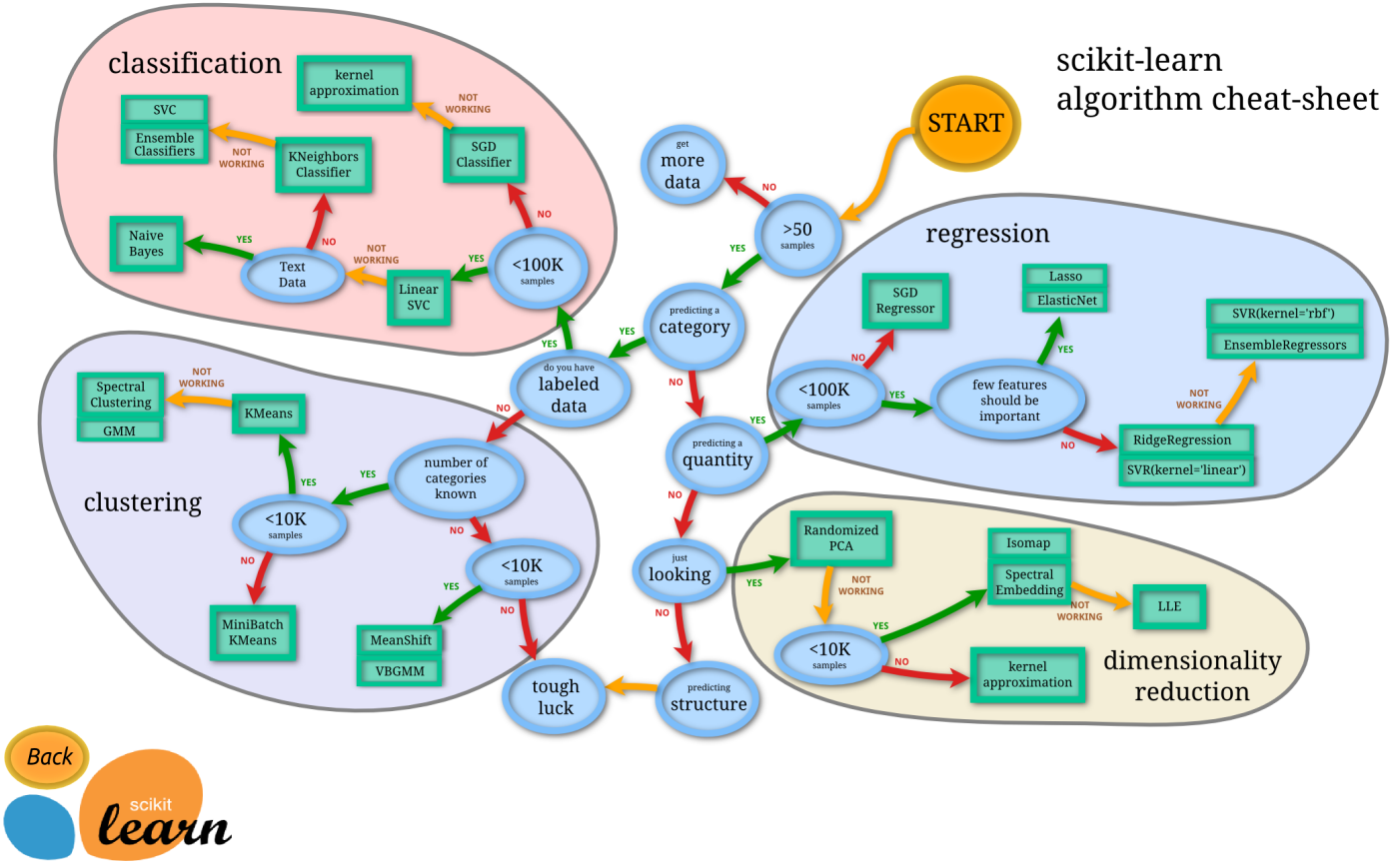
Machine Learning



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.

Machine Learning: from Theory to Practice



- How to **navigate and criticize** *Scikit-Learn* map of ML algorithms!

Machine Learning: from Theory to Practice

Goal of the course

- to present the **fundamental principles of the classical ML methodologies** in both the **supervised** and the **unsupervised** case: linear regression, logistic regression, naive Bayes, nearest neighbor, SVM, PCA, MDS, k-means, GMM, hierarchical clustering...
- to describe some **advanced methodologies**: RKHS, trees, ensemble methods, neural nets and deep learning, graph...
- to explain how **those tools and their underlying principles** can be used to tackle **less classical setting** such as non tabular data (images/text) or recommendation systems.

Machine Learning: from Theory to Practice

Teaching Team

Florence d'Alché-Buc (Telecom ParisTech)
http://perso.telecom-paristech.fr/~fdalche/
florence.dalche@telecom-paristech.fr

Erwan Le Pennec (Polytechnique)
http://www.cmap.polytechnique.fr/~lepennec/
Erwan.Le-Pennec@polytechnique.edu

Organization

- Course structured in 10 sessions mixing lectures and labs.

Schedule

- **23/09:** Introduction to ML and ML Methods I (ELP)
- **30/09:** ML Methods II (ELP)
- **07/10:** Kernel and RKHS (FdA)
- **14/10:** Unsupervised Learning and Data Lab I (ELP)
- **21/10:** Trees and Ensemble Methods (FdA)
- **04/11:** ML & Graphs (FdA)
- **18/11:** Gaussian Processes and Data Lab II (FdA)
- **25/11:** Data Lab III
- **02/12:** Neural Networks and Deep Learning I (ELP)
- **09/12** Feature Design and Deep Learning II (ELP)

Moodle and Evaluation

Moodle

- Main pedagogical tool with all the slides and labs!
- Do not forget to register if you want to validate.
- Assignment will be done through it.

Modalities

- **Lab reports** during the course period (5 pts)
- **Written exam** in January (10 pts)
- **Small project on a non classical ML task** due by the end of January (5 pts)

- **Retake** session in spring.

Bibliography


R. Schutt and C. O'Neil (2014)
Doing Data Science: Straight talk from the frontline
O'Reilly


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


M. Mohri, A. Rostamizadeh and A. Talwalkar (2012)
Foundations of Machine Learing
The MIT Press.


Ch. Bishop (2006)
Pattern Recognition and Machine Learning
Springer Series in Information Science and Statistics

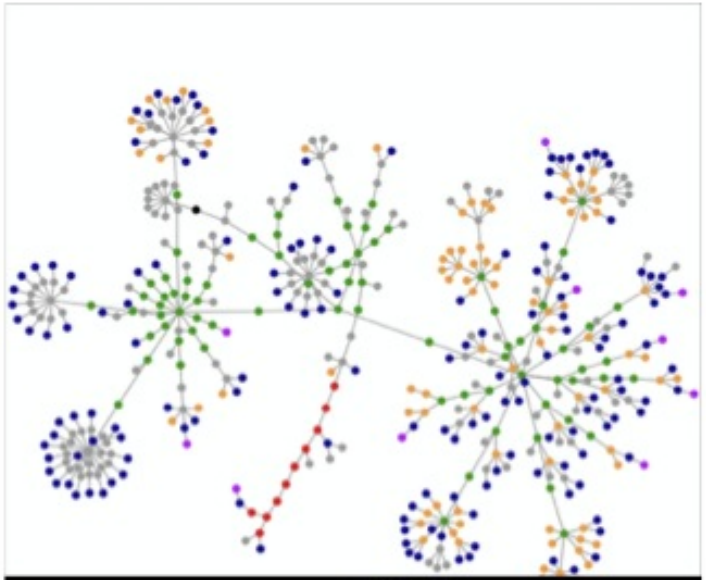

I. Goodfellow, Y. Bengio and A. Courville (2016)
Deep Learning / WIP
The MIT Press

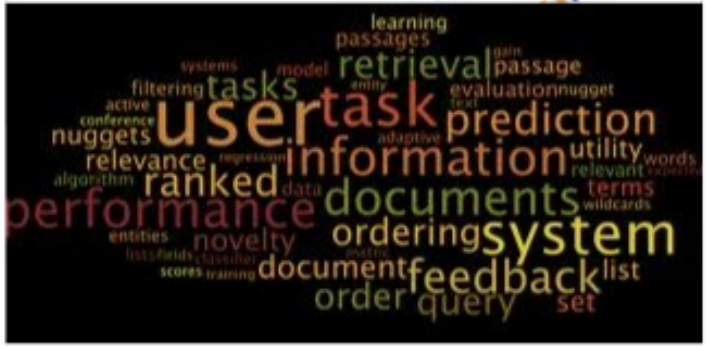
Outline

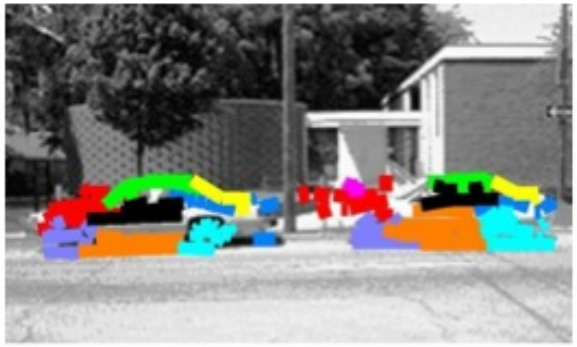
Outline

Machine learning

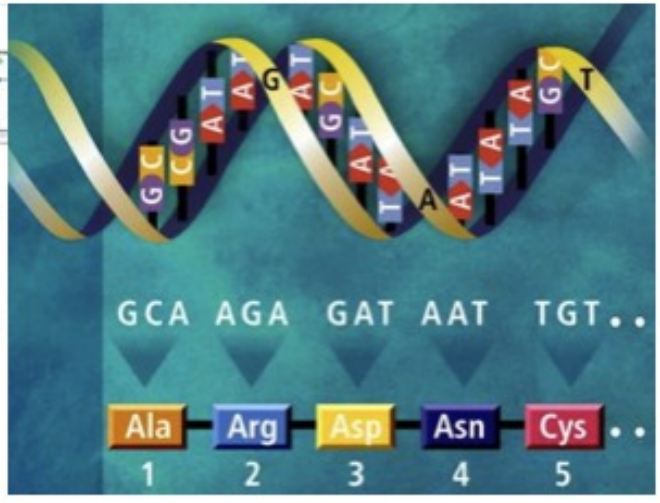
Motivation

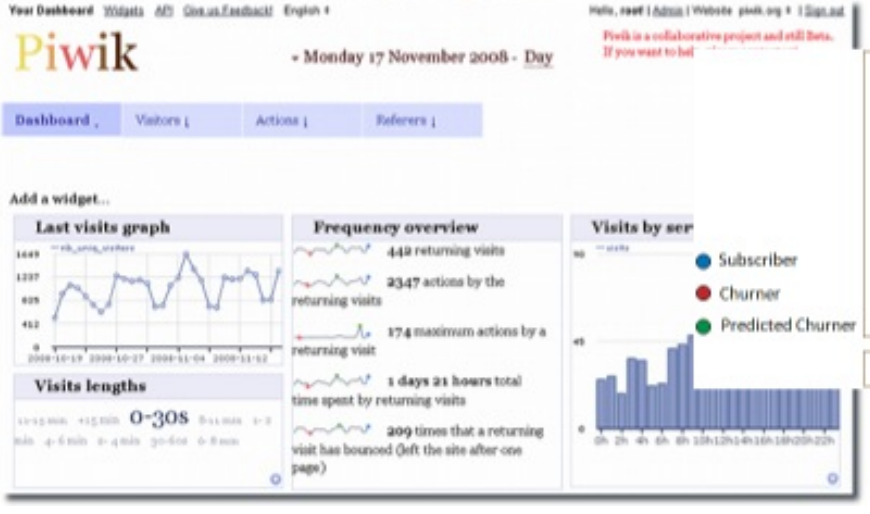


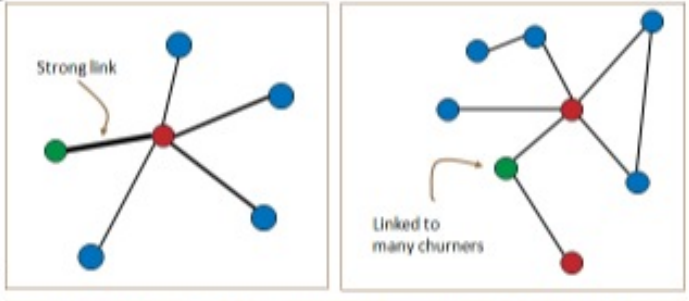




Machine Learning everywhere !







Two features of the call network + seed churners predict future churners

Machine Learning everywhere !

Use data to extract a prediction function

- Search engine, text-mining
- Diagnosis, Fault detection
- Business analytics
- Social network analytics
- Data-mining
- Link prediction
- Robotics

A definition of Machine Learning

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

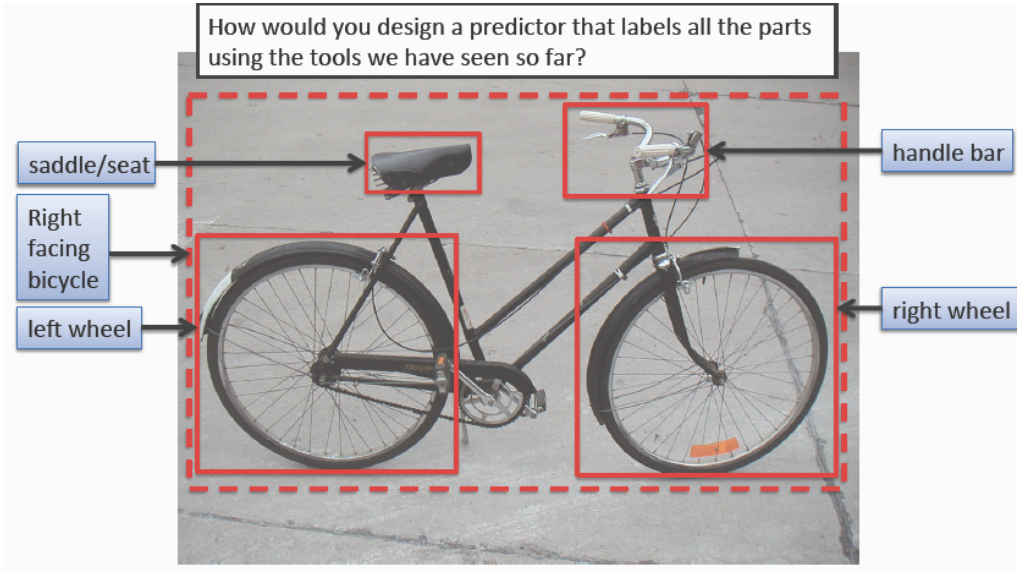
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T , as measured by P, improves with experience E.

A robot that learns

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

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

Object recognition in an image



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

Two kinds of learning 1/2

Online learning: *the learning algorithm keeps on interacting with the environment*

- robotics
- social networks
- cloud servers
- personalized advertising
- autonomous cars
- personalized healthcare
- security systems

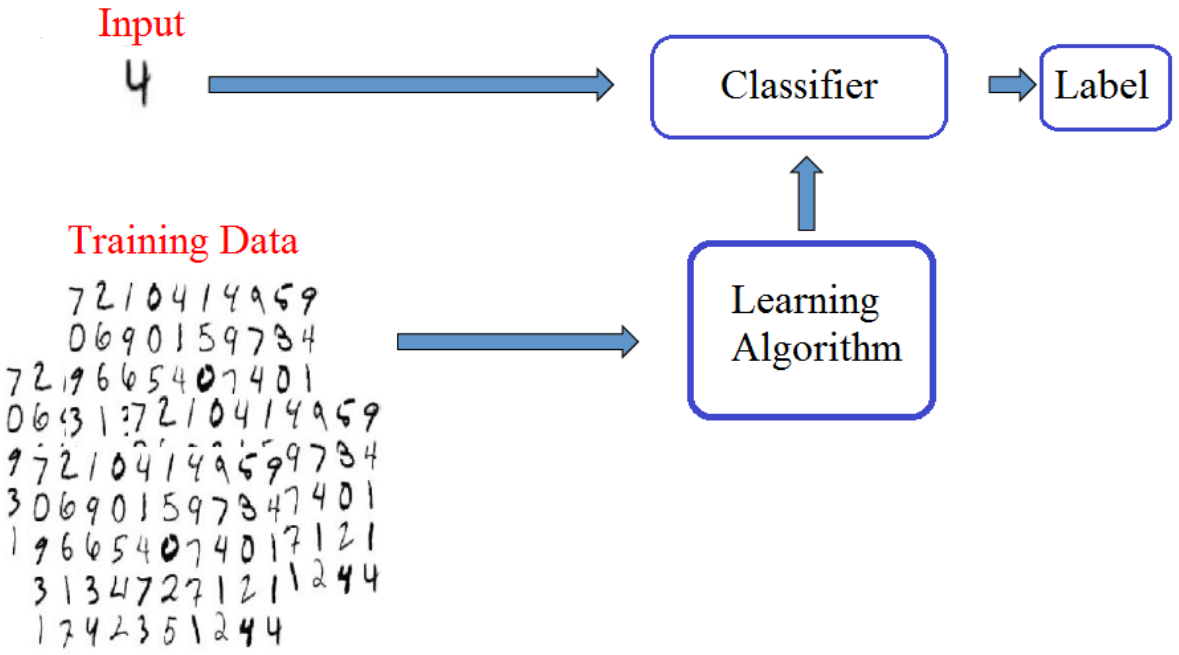
Two kinds of learning 2/2

Offline or batch learning: *the learning algorithm gets a datafile and outputs some function that can be used in turn to new data*

- pattern Recognition
- diagnosis (health, plants, ...)
- link prediction in networks
- data-mining

This course $\rightarrow$ offline learning

Machine Learning



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

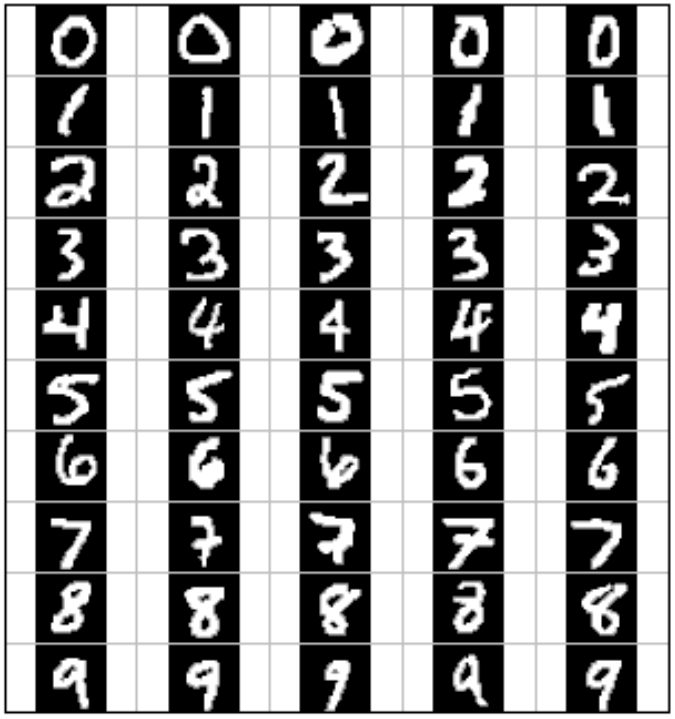
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Supervised and unsupervised

- Supervised Learning:
  - Goal: Learn a function f predicting a variable Y from an individual $\mathbf{X}$.
  - Data: Learning set $(\mathbf{X}_i, Y_i)$
- Unsupervised Learning:
  - Goal: Discover a structure within a set of individuals $(\mathbf{X}_i)$.
  - Data: Learning set $(\mathbf{X}_i)$
- First case is better posed.
- Supervised Learning (8 lectures) vs Unsupervised Learning (2 lectures)

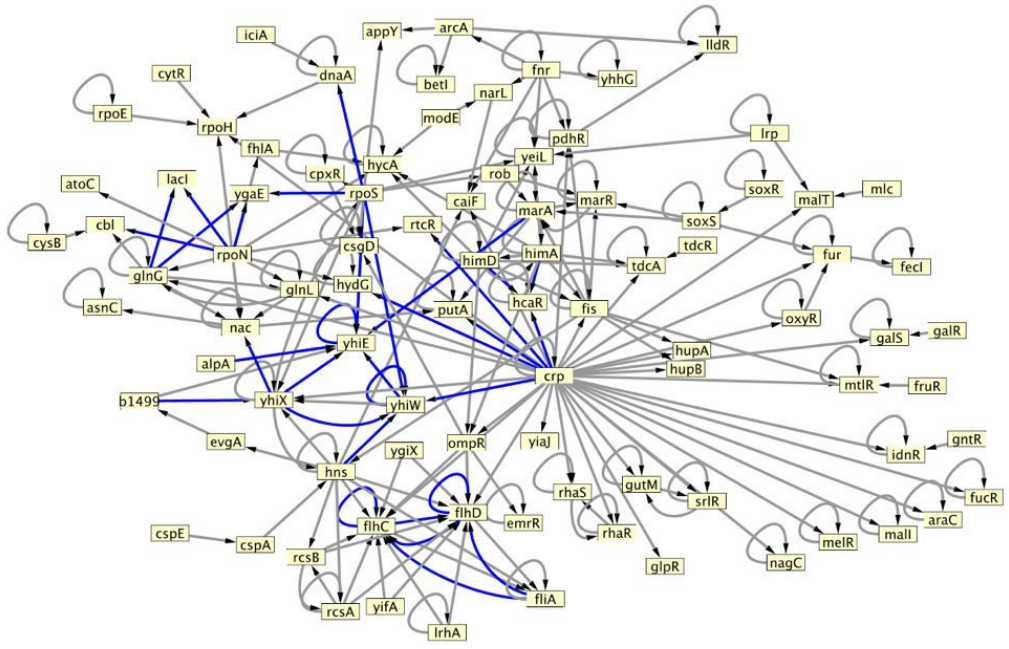
Outline

Number



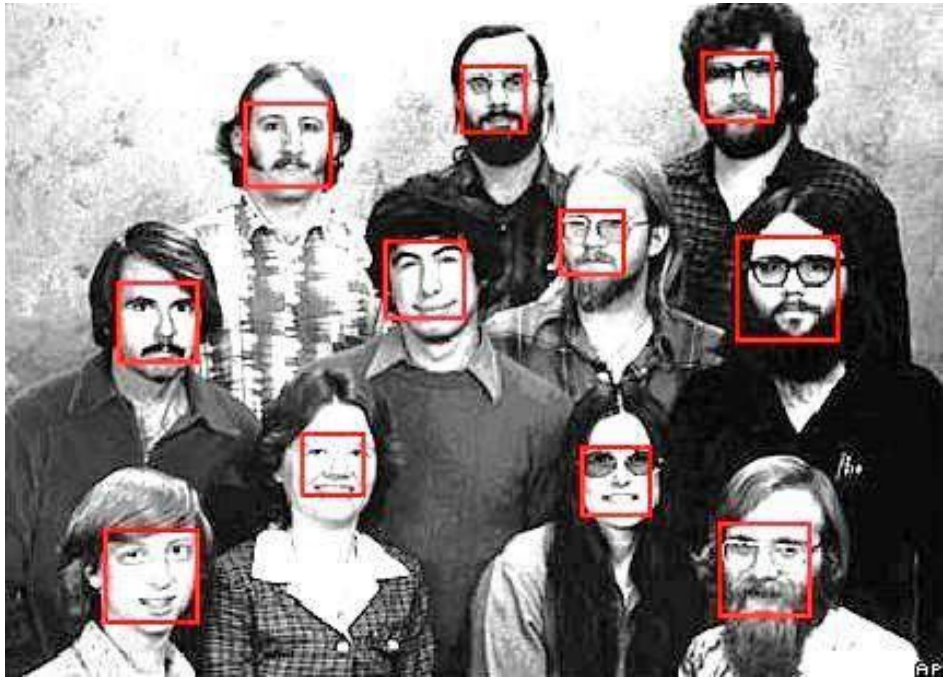
- Read a ZIP code from an envelop.
- Goal: give a number from an image.
- $\mathbf{X}$ = image.
- Y = corresponding number.

Bio



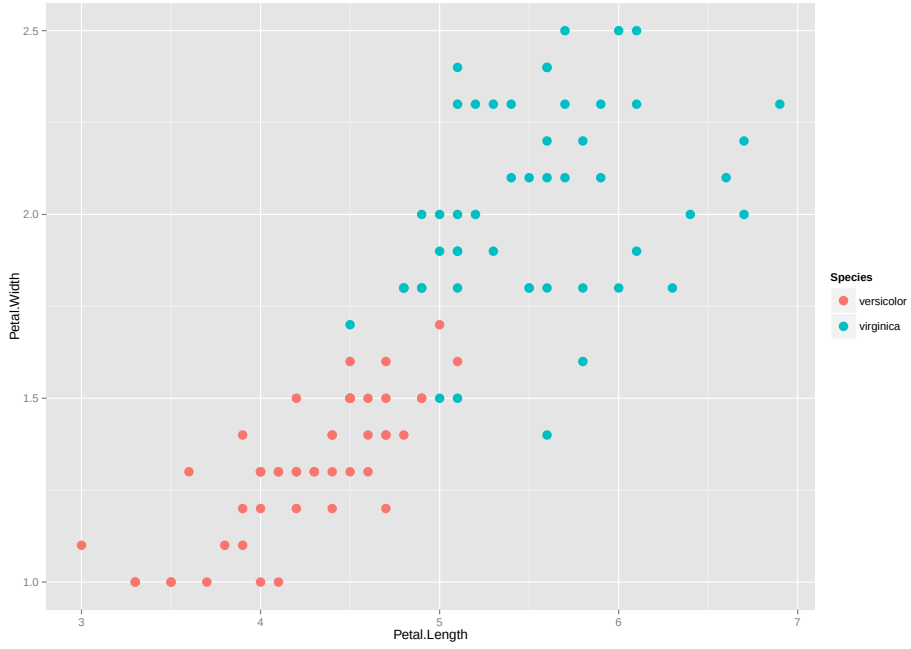
- Protein interaction network prediction.
- Goal: Predict (unknown) interactions between proteins.
- $\mathbf{X}$ = pair of proteins.
- Y = existence or no of interaction.
- Numerous similar questions in bio(informatics): genomic,...

Detection



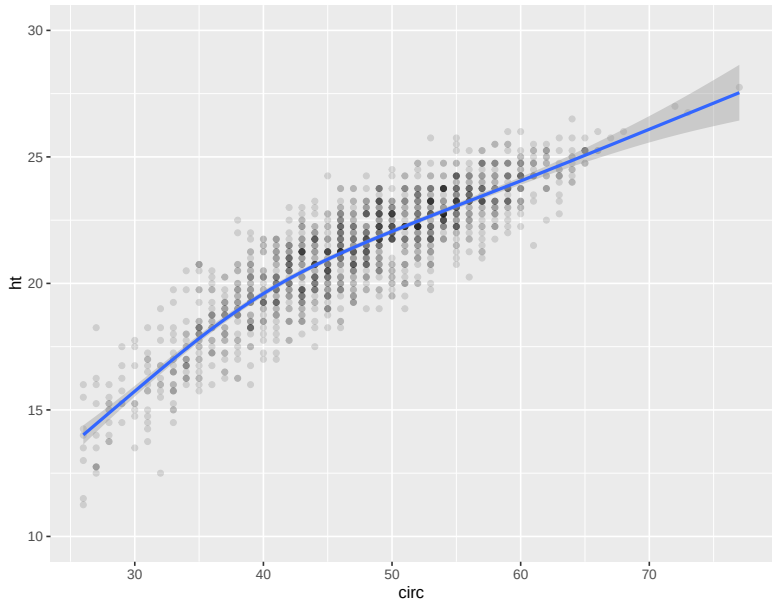
- Goal: Detect the position of faces in an image
- Different setting?
- Reformulation as a supervised learning problem.
- $\mathbf{X}$ = sub window in the image
- Y = presence or no of a face...
- Lots of answer in a single image.
- Post processing required...

Iris



- Very classical dataset.
- Goal: predict the species from the shape of the petal.
- $\mathbf{X}$ = height/width of the petal.
- Y = species.

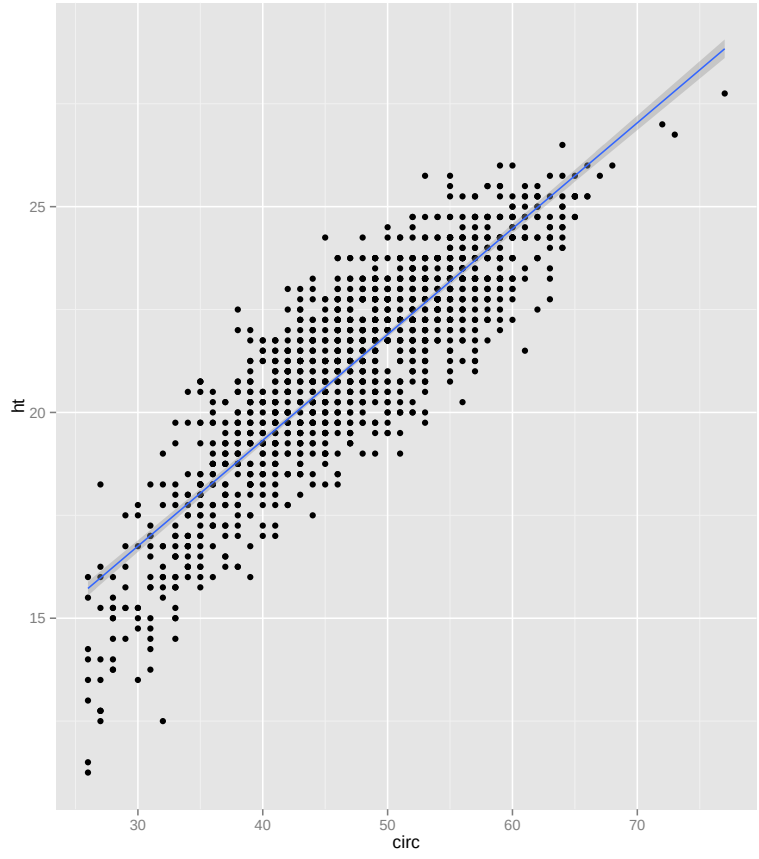
Eucalyptus



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

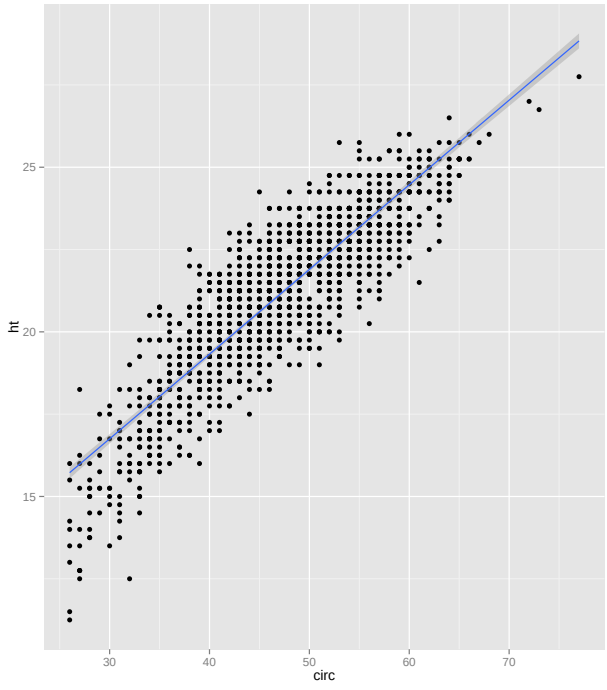
Outline

Eucalyptus



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

Eucalyptus



Linear Model

- Parametric model:

$$f_\beta(\mathbf{circ}) = \beta_1 + \beta_2 \mathbf{circ}$$

- How to choose $\beta = (\beta_1, \beta_2)$?

Least Squares

Methodology

- Natural goodness criterion:

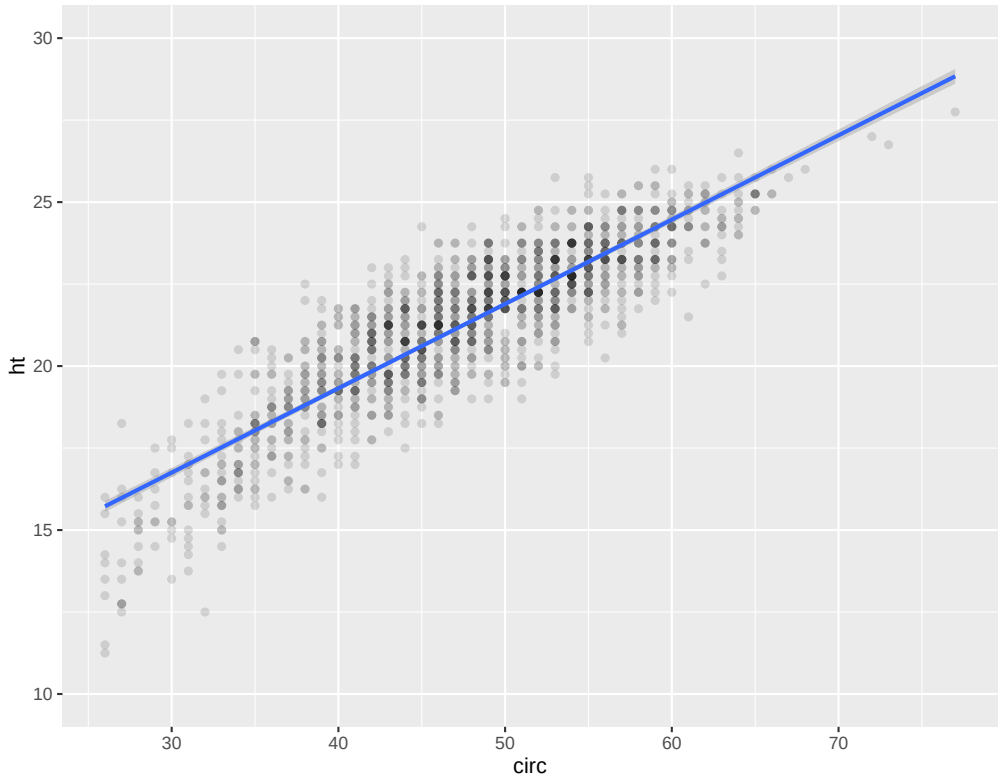
$$\sum_{i=1}^{n} |Y_i - f_\beta(\mathbf{X}_i)|^2 = \sum_{i=1}^{n} |\text{ht}_i - f_\beta(\mathbf{circ}_i)|^2$$
$$= \sum_{i=1}^{n} |\text{ht}_i - (\beta_1 + \beta_2 \mathbf{circ}_i)|^2$$

- Choice of β that minimizes this criterion!

$$\widehat{\beta} = \underset{\beta \in \mathbb{R}^2}{\text{argmin}} \sum_{i=1}^{n} |h_i - (\beta_1 + \beta_2 \mathbf{circ}_i)|^2$$

- Easy minimization with an explicit solution!

Prediction



Prediction

- Linear prediction for the height:

$$\widehat{\text{ht}} = f_{\widehat{\beta}}(\textbf{circ}) = \widehat{\beta}_1 + \widehat{\beta}_2 \textbf{circ}$$

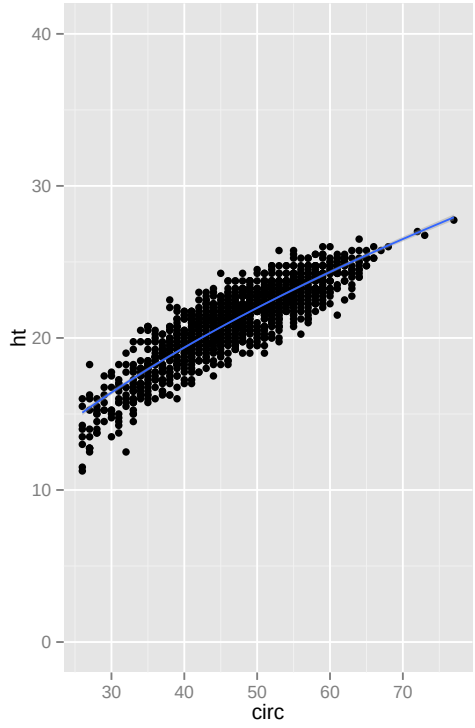
Heuristic

Linear Regression

- **Statistical model:** $(\mathbf{circ}_i, \mathrm{ht}_i)$ i.i.d. with the same law than a generic $(\mathbf{circ}, \mathrm{ht})$.
- **Performance criterion:** Look for f with a small average error
$$\mathbb{E}\left[|\mathrm{ht} - f(\mathbf{circ})|^2\right]$$
- **Empirical criterion:** Replace the unknown law by its empirical counterpart
$$\frac{1}{n}\sum_{i=1}^{n} |\mathrm{ht}_i - f(\mathbf{circ}_i)|^2$$
- **Predictor model:** As the minimum over all function is 0 (if all the $\mathbf{circ}_i$ are different), restrict to the linear functions $f(\mathbf{circ}) = \beta_1 + \beta_2 \mathbf{circ}$ to avoid over-fitting.
- **Model fitting:** Explicit formula here.

- This model can be too simple!

Polynomial Regression

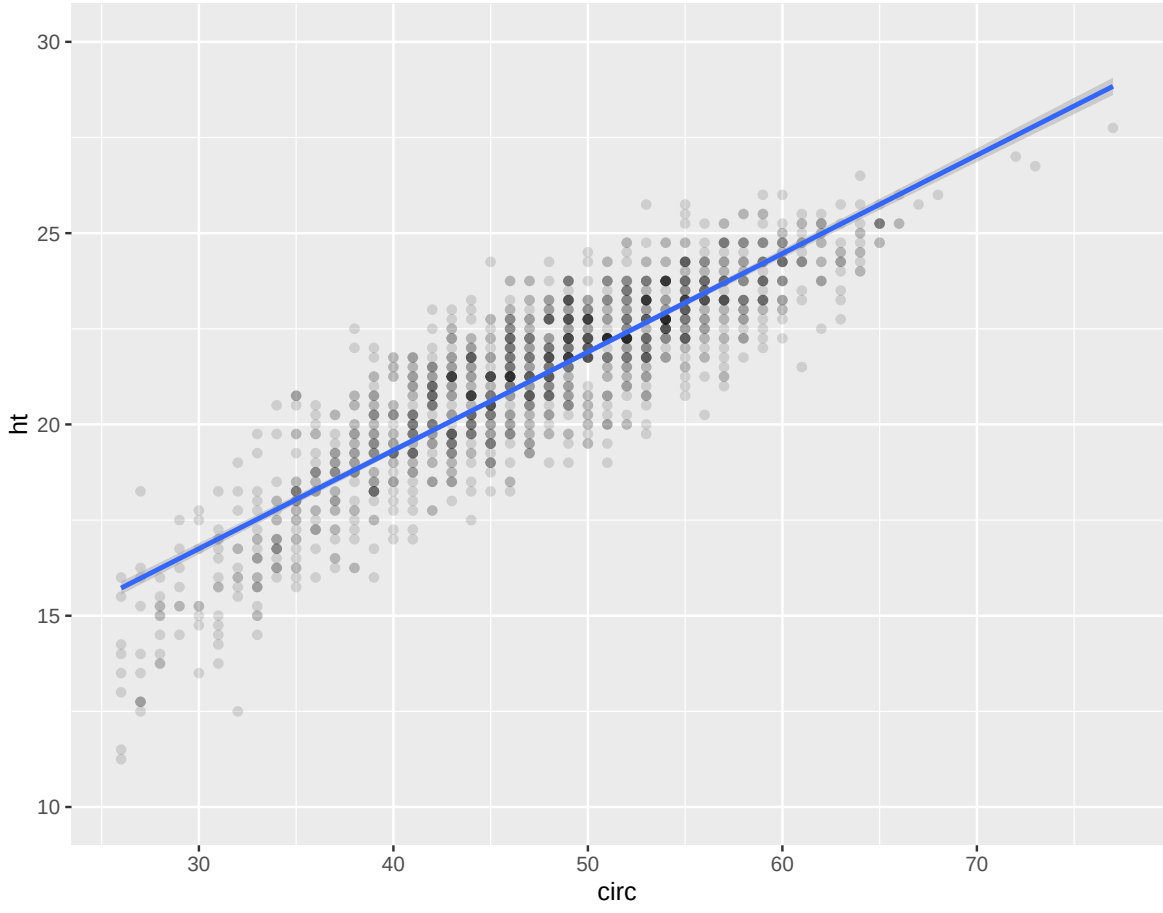


Polynomial Model

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

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

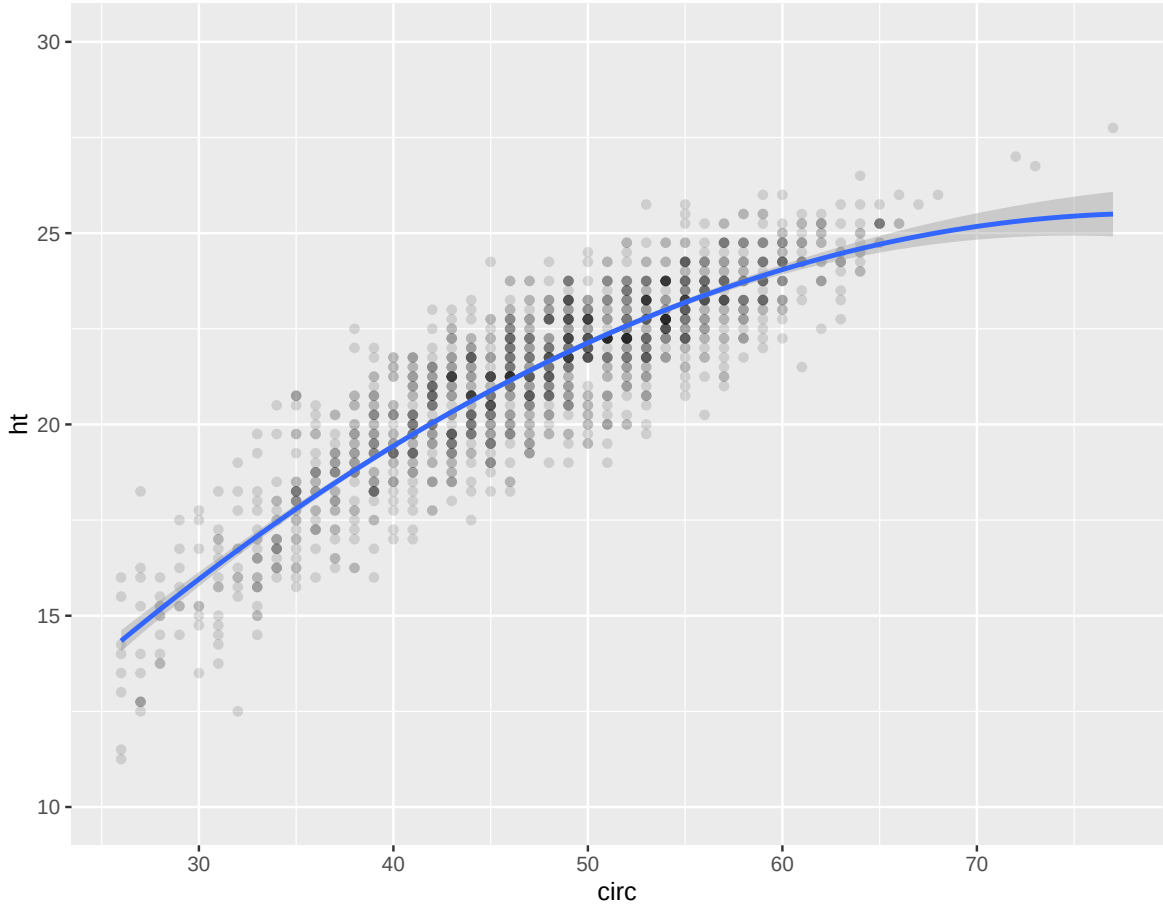
Which Degree?



Models

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

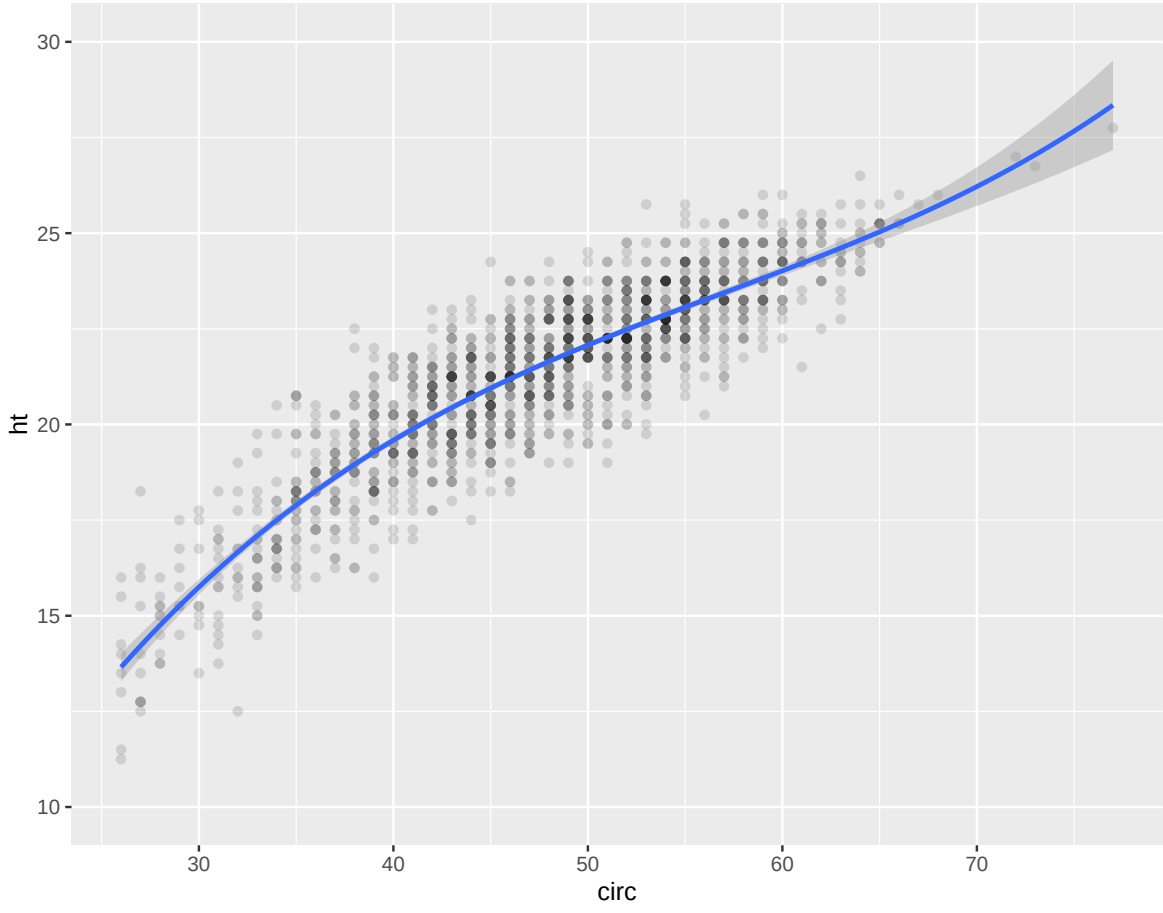
Which Degree?



Models

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

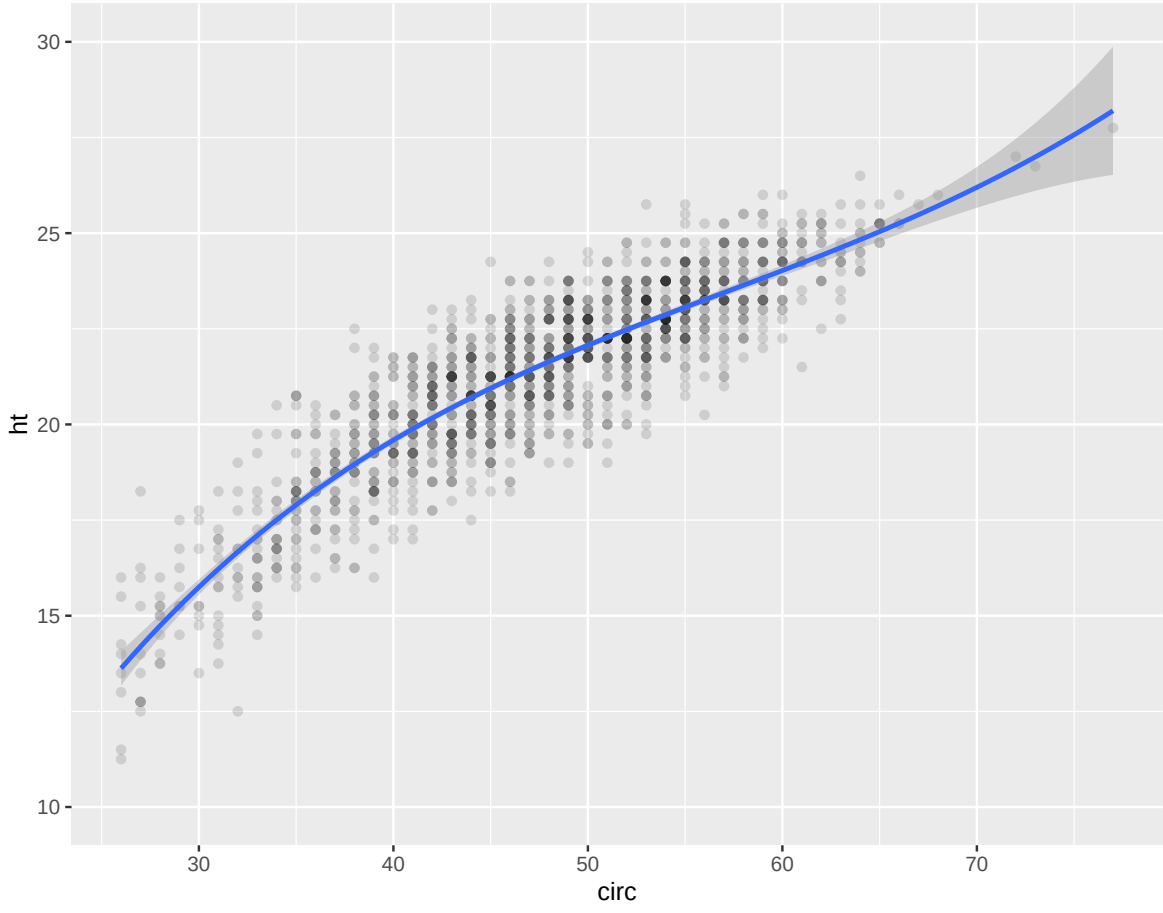
Which Degree?



Models

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

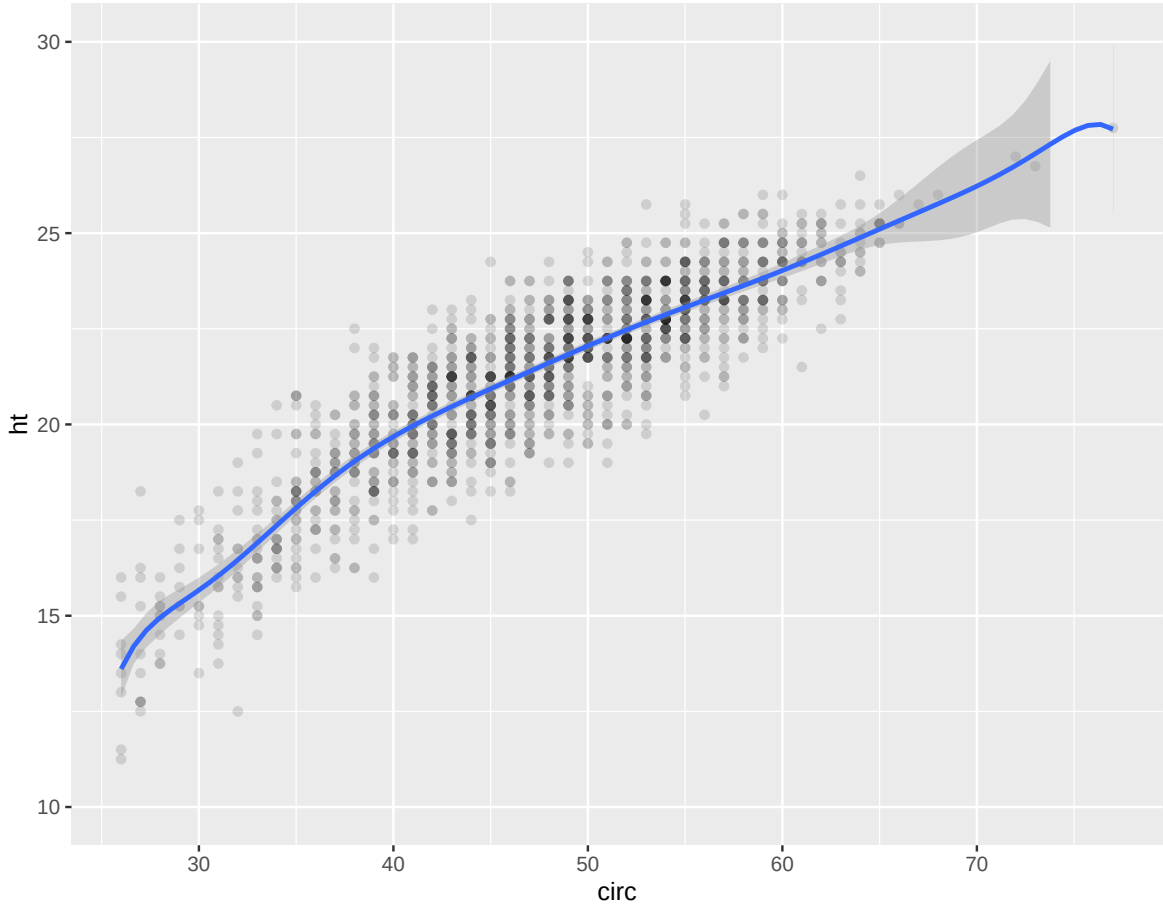
Which Degree?



Models

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

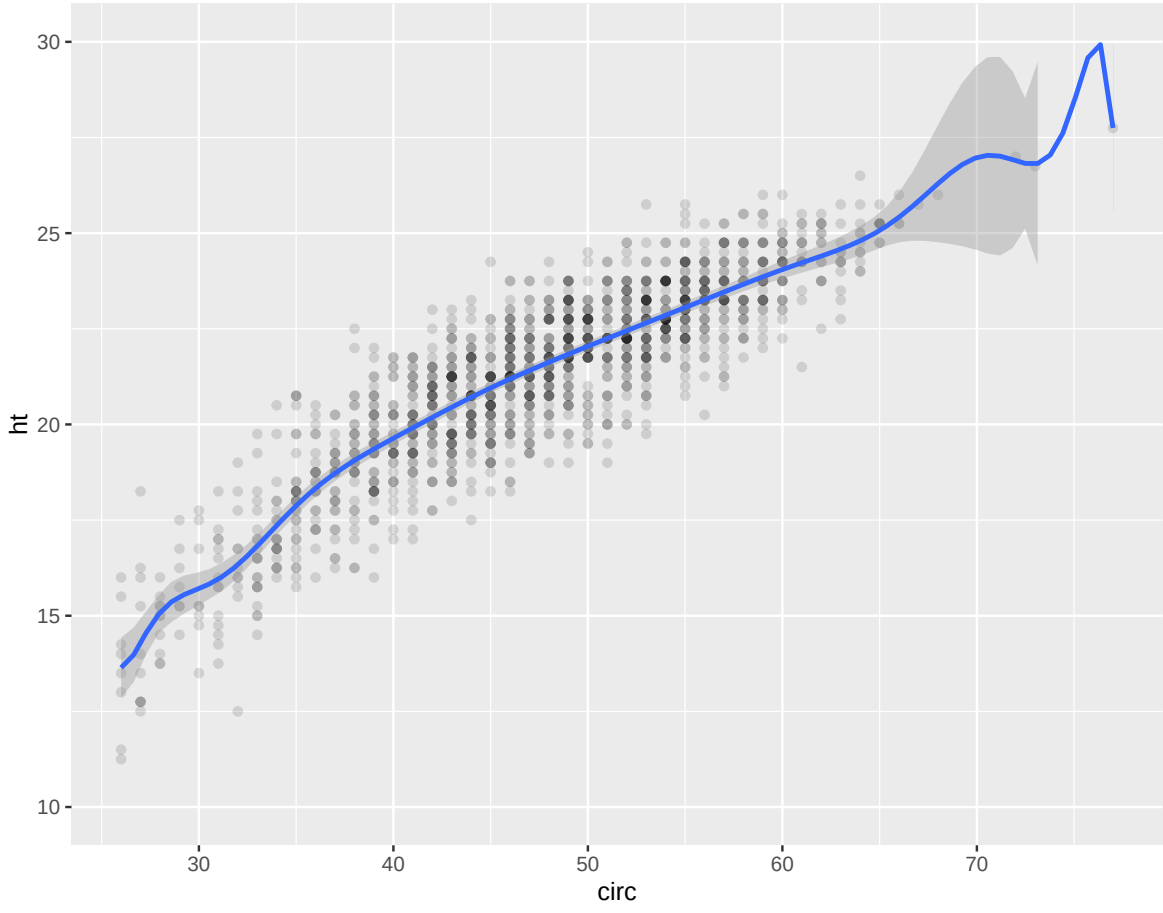
Which Degree?



Models

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

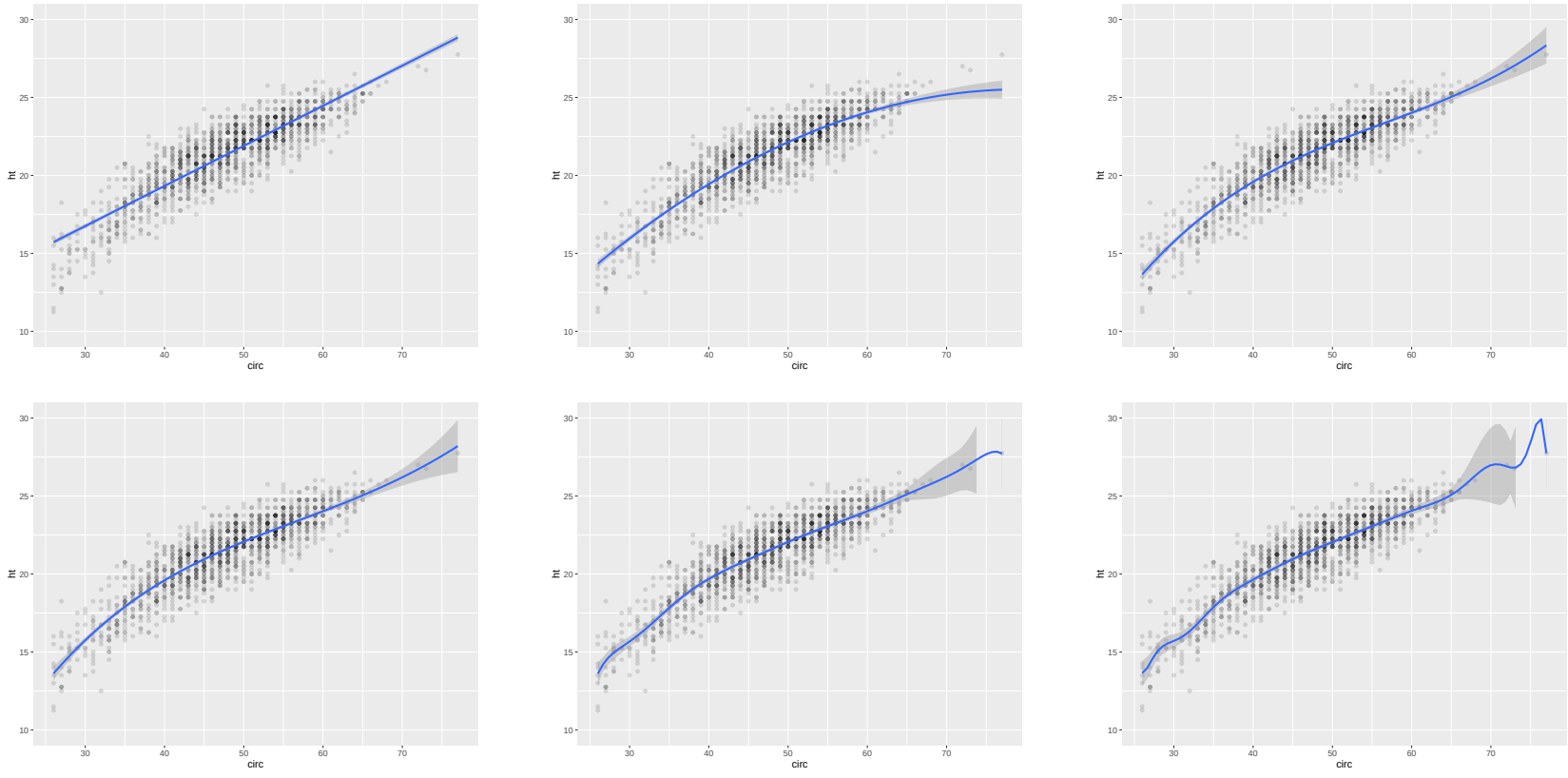
Which Degree?



Models

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

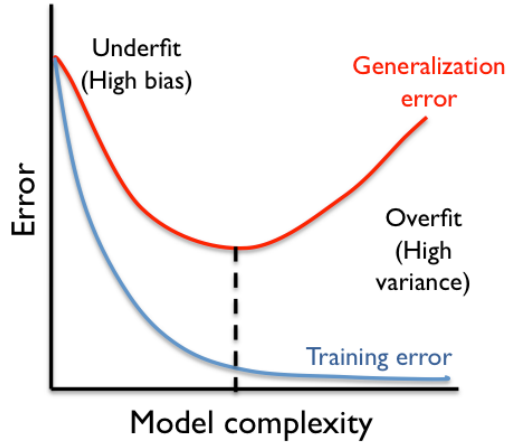
Which Degree?



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

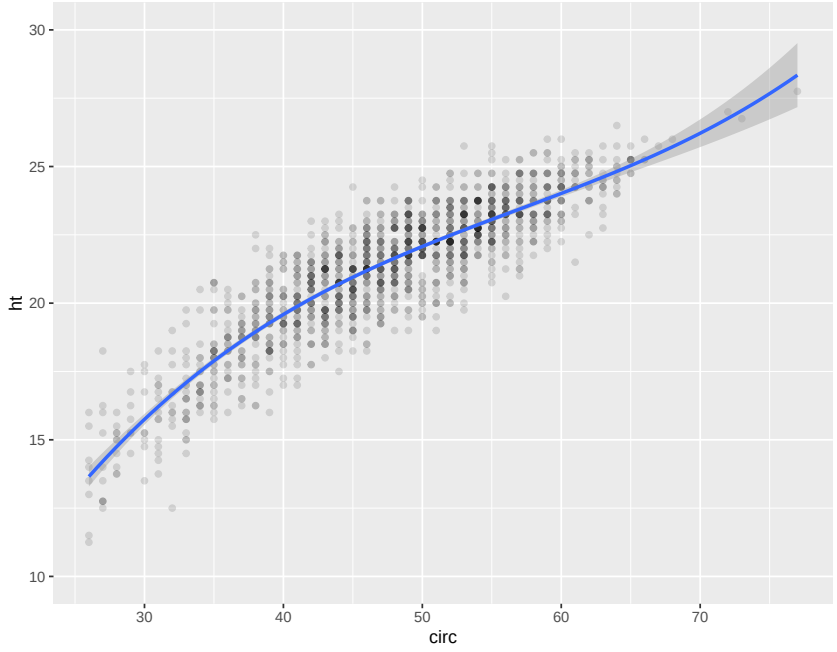
Two directions

- 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 $\mathbf{X} = (X^{(1)}, X^{(2)}, \ldots, X^{(d)}) \in \mathcal{X}$
- Output measurement $Y \in \mathcal{Y}$.
- $(\mathbf{X}, Y) \sim \mathbf{P}$ with $\mathbf{P}$ unknown.
- Training data : $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbf{P}$)
- Often
  - $\mathbf{X} \in \mathbb{R}^d$ and $Y \in \{-1, 1\}$ (classification)
  - or $\mathbf{X} \in \mathbb{R}^d$ and $Y \in \mathbb{R}$ (regression).

- A classifier is a function in $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y} \text{ measurable}\}$

Goal

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

- Need to specify the meaning of good.
- Formally, classification and regression are the same problem!

Loss and Probabilistic Framework

Loss function

- Loss function : $\ell(Y, f(\mathbf{X}))$ measure how well $f(\mathbf{X})$ *predicts* Y.
- Examples:
  - Prediction loss: $\ell(Y, f(\mathbf{X})) = \mathbf{1}_{Y \neq f(\mathbf{X})}$
  - Quadratic loss: $\ell(Y, \mathbf{X}) = |Y - f(\mathbf{X})|^2$

Risk of a generic classifier

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

$$\mathcal{R}(f) = \mathbb{E}\left[\ell(Y, f(\mathbf{X}))\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|\mathbf{X}}\left[\ell(Y, f(\mathbf{X}))\right]\right]$$

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

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

Supervised Learning

Experience, Task and Performance measure

- Training data : $\mathcal{D} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbf{P}$)
- Predictor: $f : \mathcal{X} \to \mathcal{Y}$ measurable
- Cost/Loss function : $\ell(Y, f(\mathbf{X}))$ measure how well $f(\mathbf{X})$ *predicts* Y
- Risk:

$$\mathcal{R}(f) = \mathbb{E}\left[\ell(Y, f(\mathbf{X}))\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|\mathbf{X}}\left[\ell(Y, f(\mathbf{X}))\right]\right]$$

- Often $\ell(Y, f(\mathbf{X})) = |f(\mathbf{X}) - Y|^2$ or $\ell(Y, f(\mathbf{X})) = \mathbf{1}_{Y \neq f(\mathbf{X})}$

Goal

- Learn a rule to construct a classifier $\widehat{f} \in \mathcal{F}$ 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 $\mathcal{D}_n$.

Best Solution

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

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

Bayes Classifier (explicit solution)

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

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

- In regression with the quadratic loss

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

Issue: Explicit solution requires to know $\mathbb{E}[Y|\mathbf{X}]$ for all values of $\mathbf{X}$!

Goal

Machine Learning

- Learn a rule to construct a classifier $\widehat{f} \in \mathcal{F}$ 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 $\mathcal{D}_n$.

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

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

- Examples:
  - Linear regression
  - Linear discrimination with

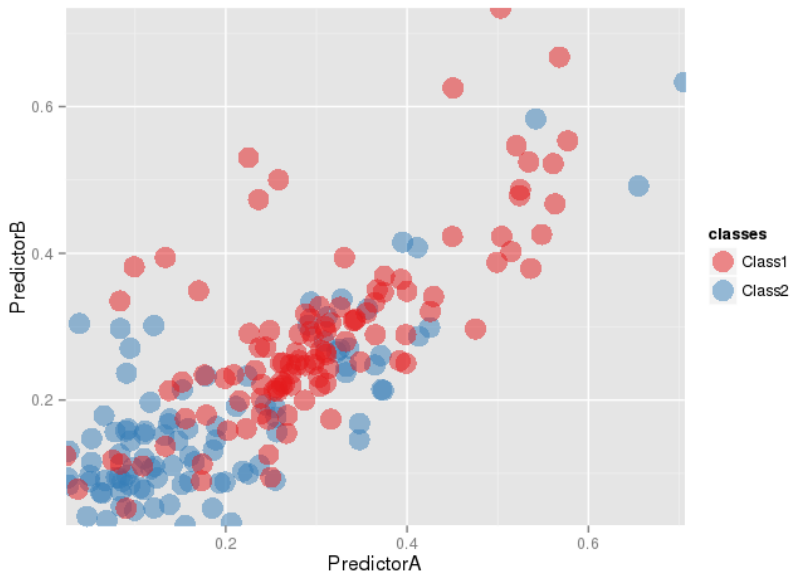
$$\mathcal{S} = \{\mathbf{x} \mapsto \mathtt{sign}\{\beta^T \mathbf{x} + \beta_0\} / \beta \in \mathbb{R}^d, \beta_0 \in \mathbb{R}\}$$

Example: TwoClass Dataset

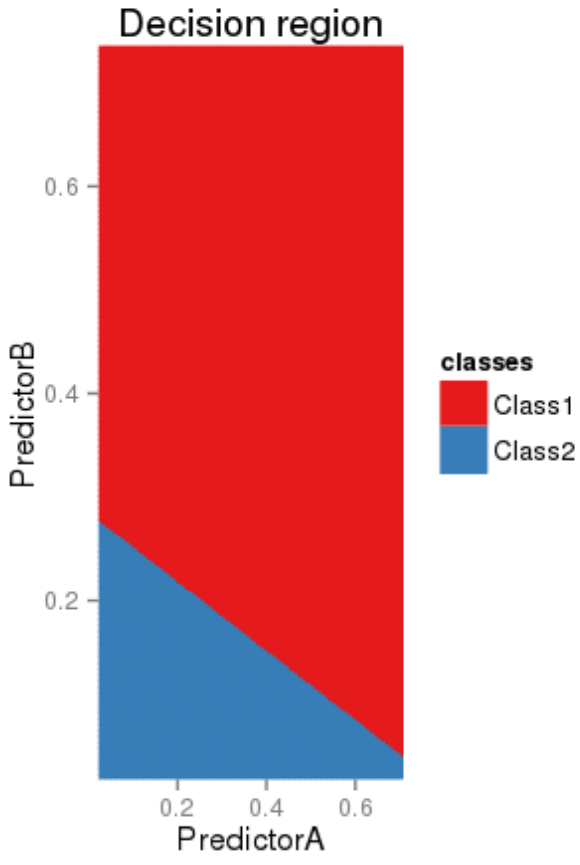
Synthetic Dataset

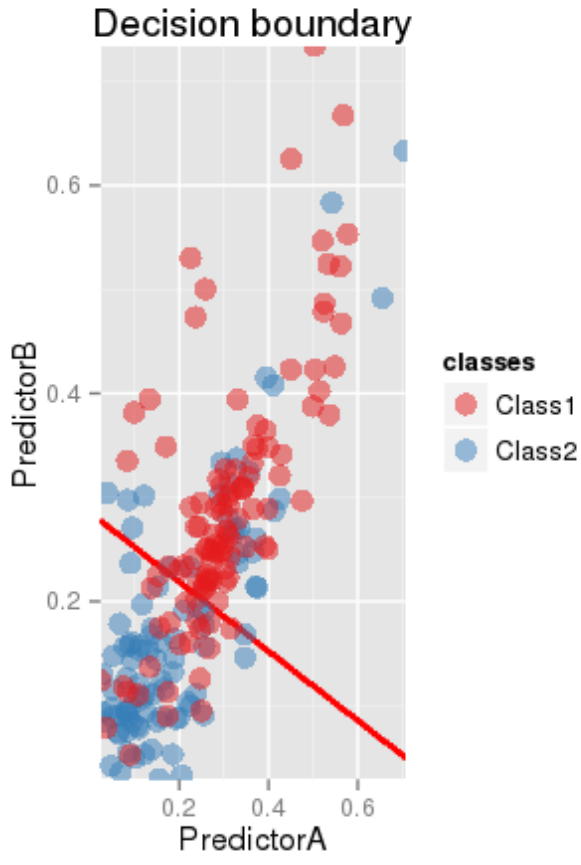
- Two features/covariates.
- Two classes.

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

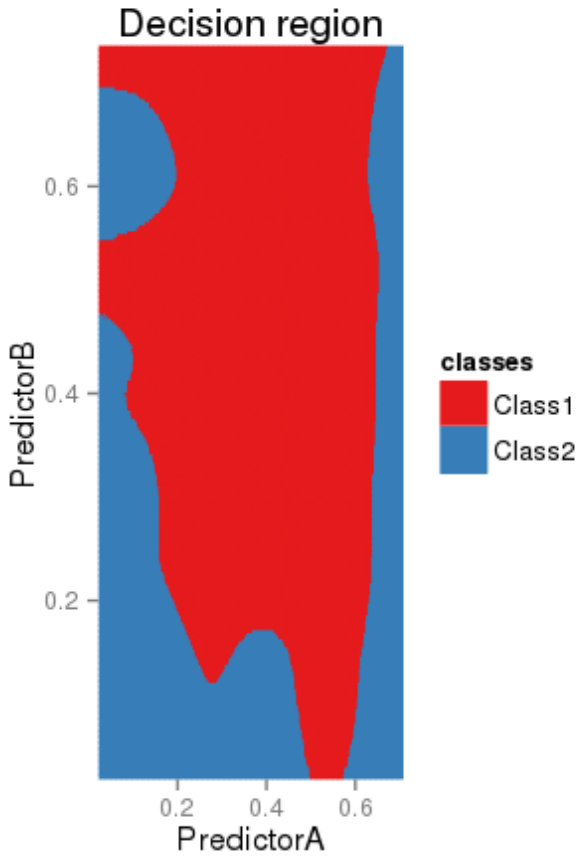


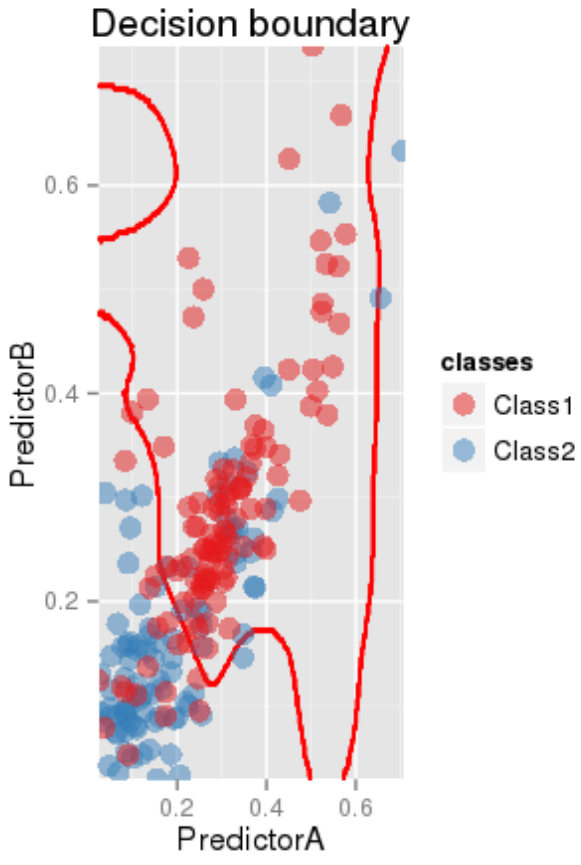
Example: Linear Discrimination



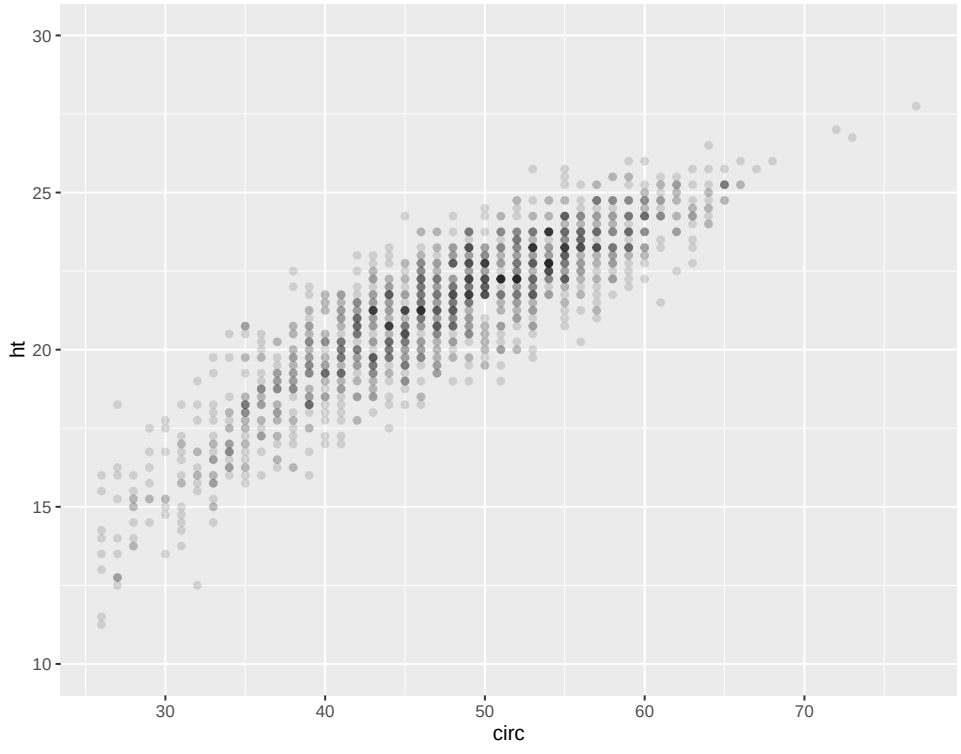


Example: More Complex Model





Eucalyptus

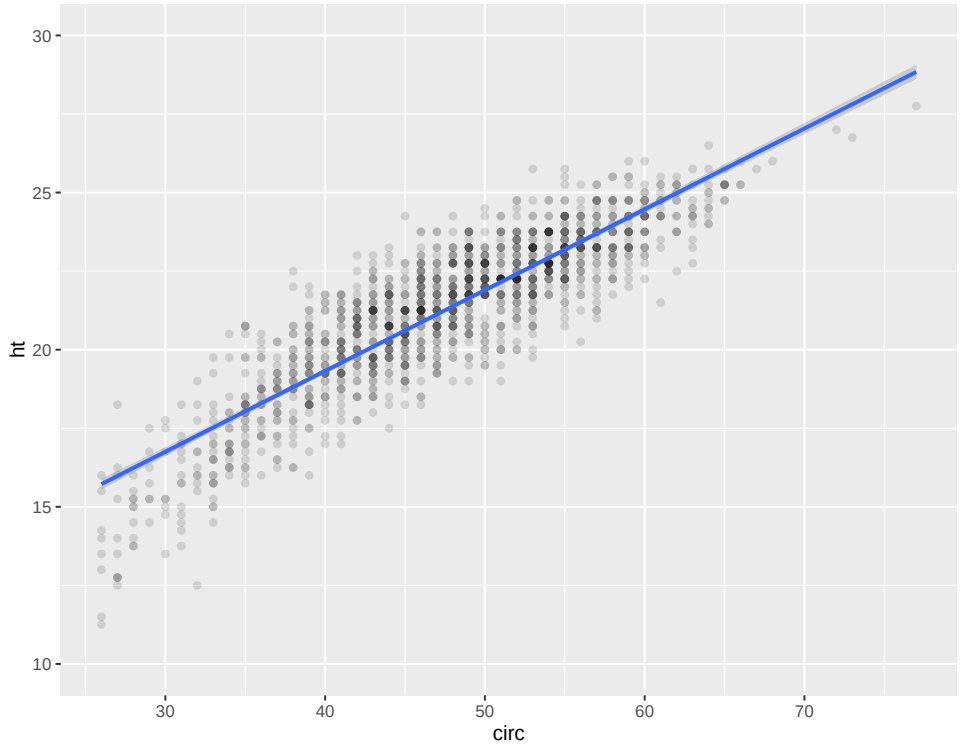


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - X: circumference / Y: height

- Can we predict the height from the circumference?

Eucalyptus

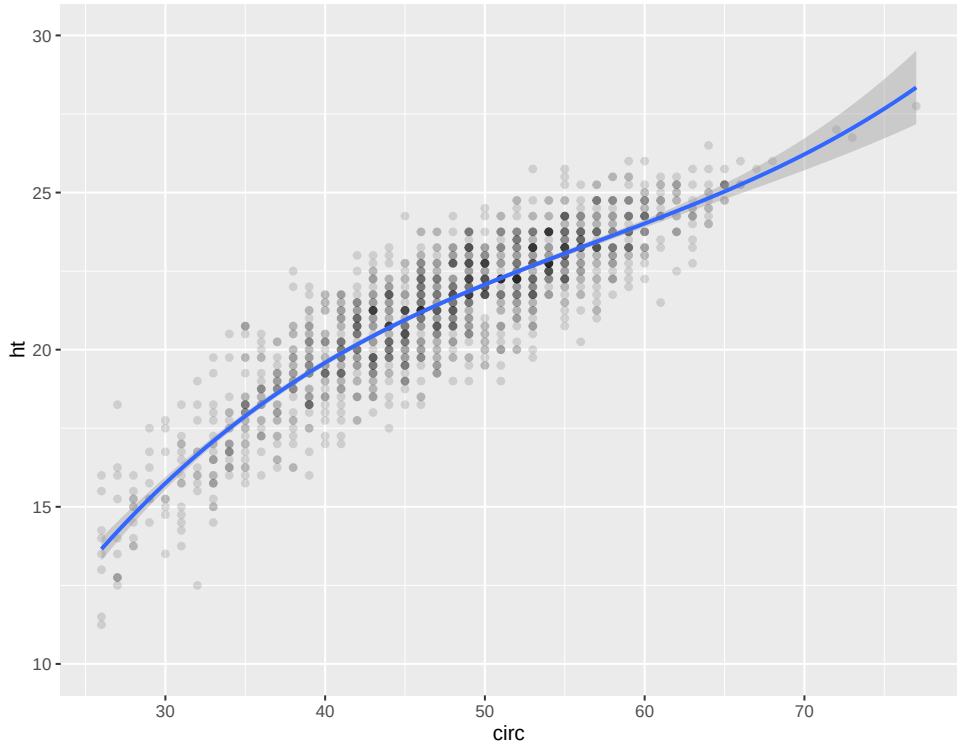


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - X: circumference / Y: height

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

Eucalyptus

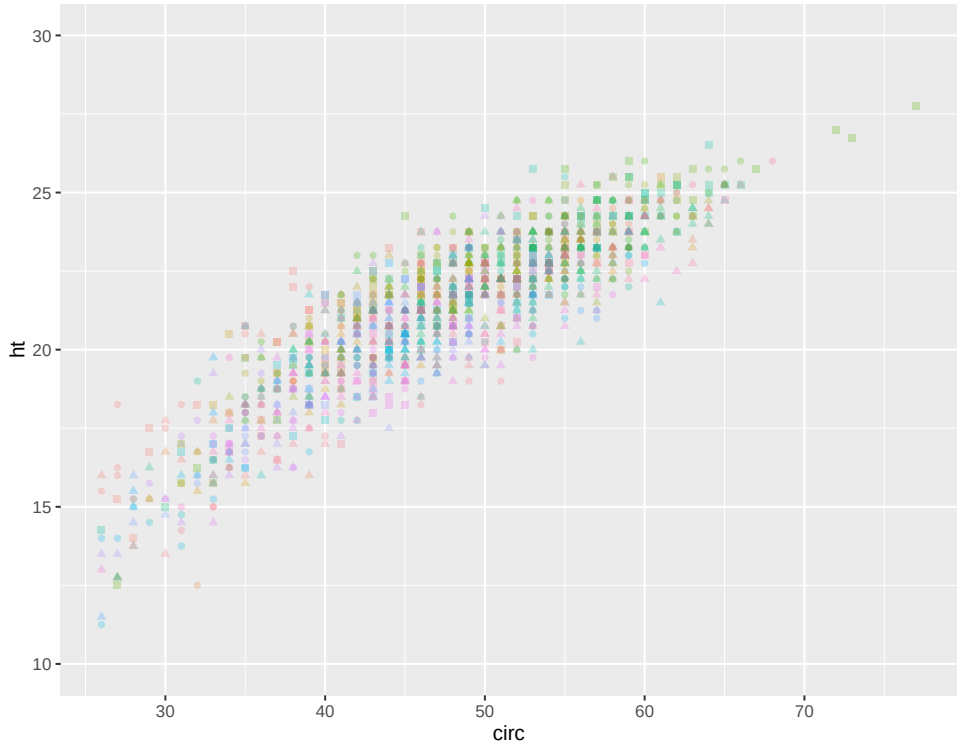


Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - X: circumference / Y: height

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

Eucalyptus



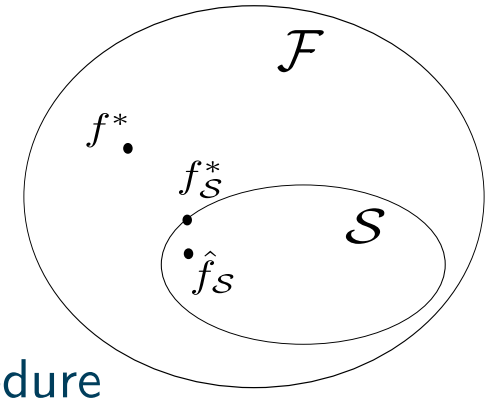
Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - **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?

Bias-Variance Dilemna

- General setting:
  - $\mathcal{F} = \{\text{measurable fonctions } \mathcal{X} \to \mathcal{Y}\}$
  - 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)

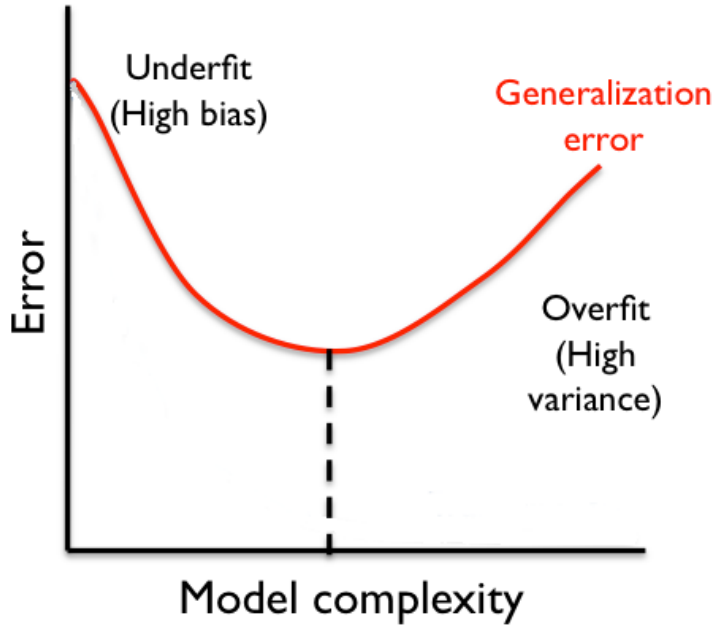
$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^*) = \underbrace{\mathcal{R}(f_{\mathcal{S}}^*) - \mathcal{R}(f^*)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f_{\mathcal{S}}^*)}_{\text{Estimation error}}$$

- 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 $(\mathbf{X}, Y)$.

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

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 $\mathbf{P}$! (Statistical Learning?)

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

Generative and Discriminative PoV

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

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

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

Problems

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

A Generative Point of View

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

A Discriminative 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

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

Bayes Classifier (explicit solution)

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

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

- In regression with the quadratic loss

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

Issue: Explicit solution requires to know $Y|\mathbf{X}$ (or rather here $\mathbb{E}\left[Y|\mathbf{X}\right]$) for all values of $\mathbf{X}$!

Plugin Classifier

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

Plugin Bayes Classifier

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

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

- In regression with the quadratic loss

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

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

Instantiation

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

Three main heuristics

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

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

Plugin Classifier

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

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

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

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

Classification Risk Analysis

Theorem

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

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

- If one estimates $\mathbb{P}\{Y = 1|\mathbf{X}\}$ 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 $\mathbb{P}\{Y = 1|\mathbf{X}\}$ while the classification task focus on values around $1/2$ for the $0/1$ loss!
- In **regression**, (often) direct control of the quadratic loss...

Risk Analysis (Proof)

- Let us denote $p_1(\mathbf{X}) = \mathbb{P}\{Y = 1|\mathbf{X}\}$
- Step 1: Let $\tilde{f}(\mathbf{X}) = \text{sign}(2\tilde{p}_1(\mathbf{X}) - 1)$

$$\mathbb{E}\left[\ell^{0/1}(Y, \tilde{f}(\mathbf{X}))\right] = \mathbb{E}_{\mathbf{X}}\left[p_1(\mathbf{X})\mathbf{1}_{\tilde{f}(\mathbf{X})=-1} + (1 - p_1(\mathbf{X}))\mathbf{1}_{\tilde{f}(\mathbf{X})=1}\right]$$
$$= \mathbb{E}_{\mathbf{X}}\left[(1 - p_1(\mathbf{X})) + (2p_1(\mathbf{X}) - 1)\mathbf{1}_{\tilde{f}(\mathbf{X})=-1}\right]$$

- Step 2:

$$\mathbb{E}\left[\ell^{0/1}(Y, \tilde{f}(\mathbf{X}))\right] - \mathbb{E}\left[\ell^{0/1}(Y, \tilde{f}^\star(\mathbf{X}))\right]$$
$$= \mathbb{E}_{\mathbf{X}}\left[(2p_1(\mathbf{X}) - 1)(\mathbf{1}_{\tilde{f}(\mathbf{X})=-1} - \mathbf{1}_{f^\star(\mathbf{X})=-1})\right]$$

using the definition of $f^\star = \text{sign}(2p(\mathbf{X} - 1)$

$$= \mathbb{E}_{\mathbf{X}}\left[|2p_1(\mathbf{X}) - 1|\mathbf{1}_{f^\star(\mathbf{X})\neq\tilde{f}(\mathbf{X})}\right]$$

and using the fact that $f^\star(\mathbf{X}) \neq \tilde{f}(\mathbf{X})$ implies that $\widehat{p}(\mathbf{X})$ and $p(\mathbf{X})$ are not on the same side with respect to $1/2$

$$\leq 2\mathbb{E}_{\mathbf{X}}\left[|p_1(\mathbf{X}) - \widehat{p}_1(\mathbf{X})|\right]) = \mathbb{E}_{\mathbf{X}}\left[\|p(\mathbf{X}) - \widehat{p}(\mathbf{X})\|_1\right]$$

using $\|P - Q\|_1 \leq \sqrt{2\text{KL}(P, Q)}$ and Jensen

$$\leq \mathbb{E}_{\mathbf{X}}\left[\sqrt{2\text{KL}(p(\mathbf{X}), \widehat{p}(\mathbf{X}))}\right] \leq \left(\mathbb{E}_{\mathbf{X}}\left[2\text{KL}(p(\mathbf{X}), \widehat{p}(\mathbf{X}))\right]\right)^{1/2}$$

Outline

Fully Generative Modeling

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

Bayes formula

- With a slight abuse of notation,

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

- **Generative Modeling:**
  - Propose a model for $(\mathbf{X}, Y)$ (or equivalently $\mathbf{X}|Y$ 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 $(\mathbf{X}, Y)$ rather than only $Y|\mathbf{X}$!
- Great flexibility in the model design but may lead to complex computation.

Fully Generative Modeling

- Simpler setting in clasification!

Bayes formula

$$\mathbb{P}\{Y = k|\mathbf{X}\} = \frac{\mathbb{P}\{\mathbf{X}|Y = k\}\mathbb{P}\{Y = k\}}{\mathbb{P}\{\mathbf{X}\}}$$

- Binary Bayes classifier (the best solution)

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

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

Discriminant Analysis (Gaussian model)

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

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

Estimation

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

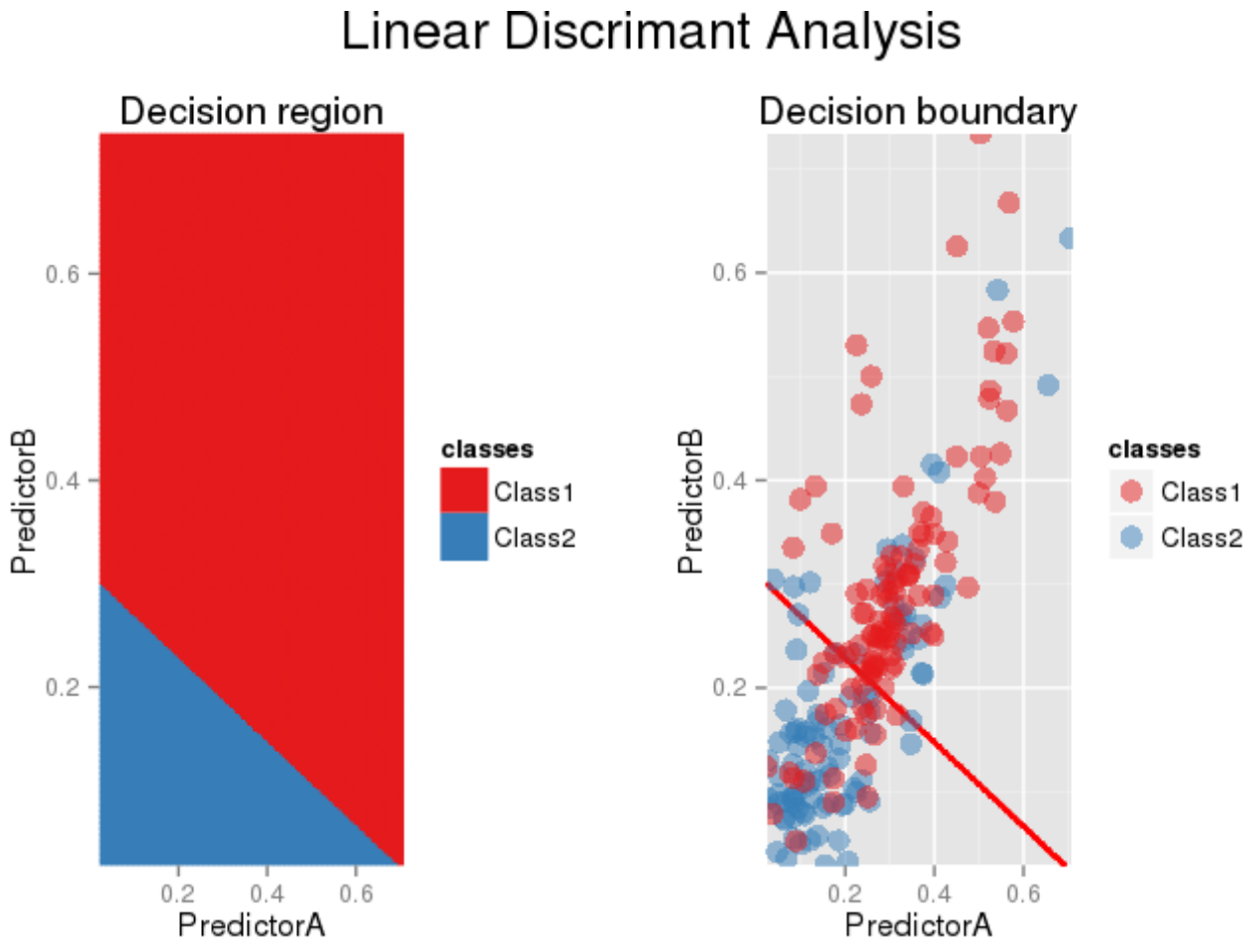
- The estimate proportion $\widehat{\mathbb{P}_k} = \frac{n_k}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{Y_i = k\}}$
- Maximum likelihood estimate of $\widehat{\mu_k}$ and $\widehat{\Sigma_k}$ (explicit formulas)

- DA classifier

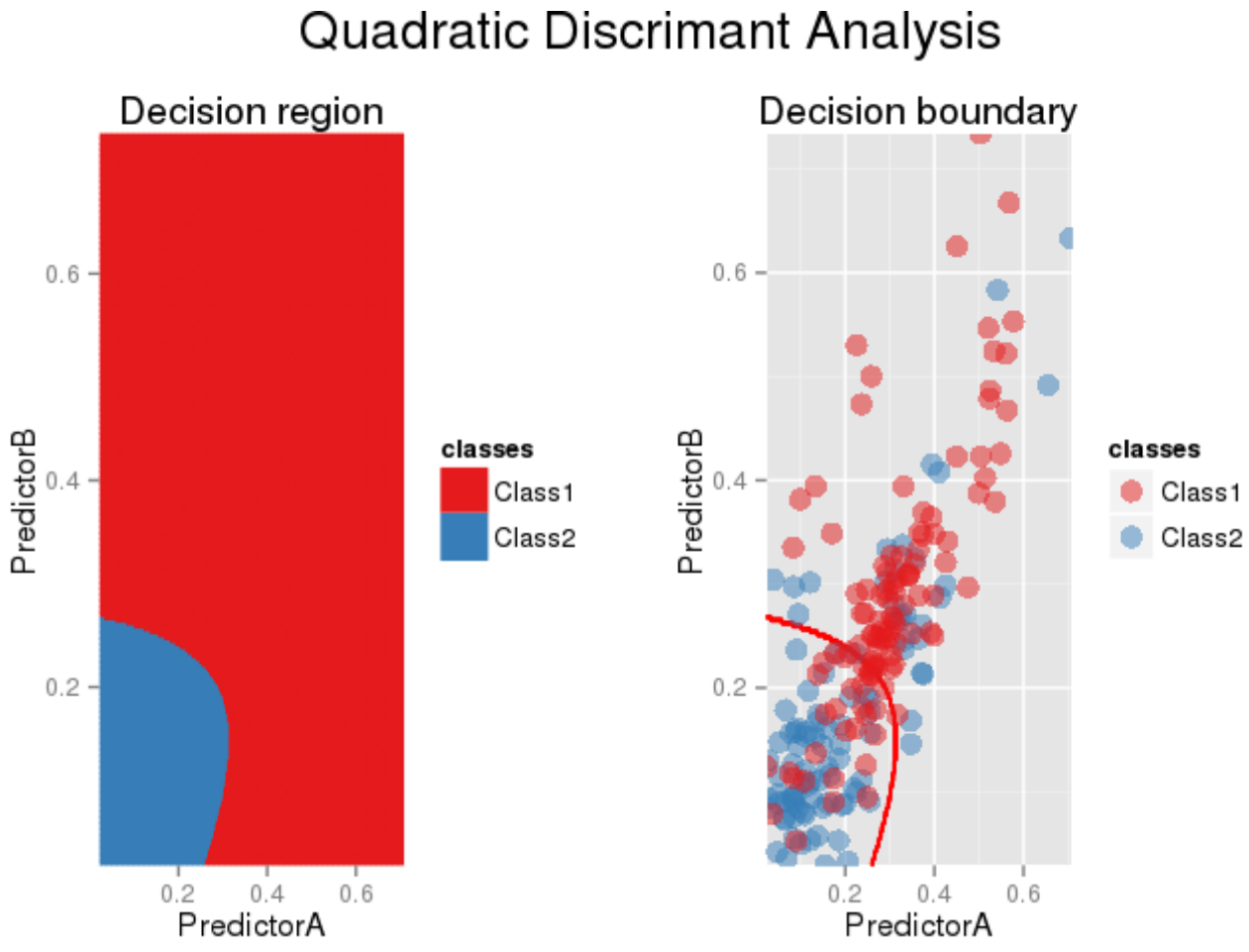
$$\widehat{f}_G(\mathbf{X}) = \begin{cases} +1 & \text{if } \widehat{g}_{+1}(\mathbf{X}) \geq \widehat{g}_{-1}(\mathbf{X}) \\ -1 & \text{otherwise} \end{cases}$$

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

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

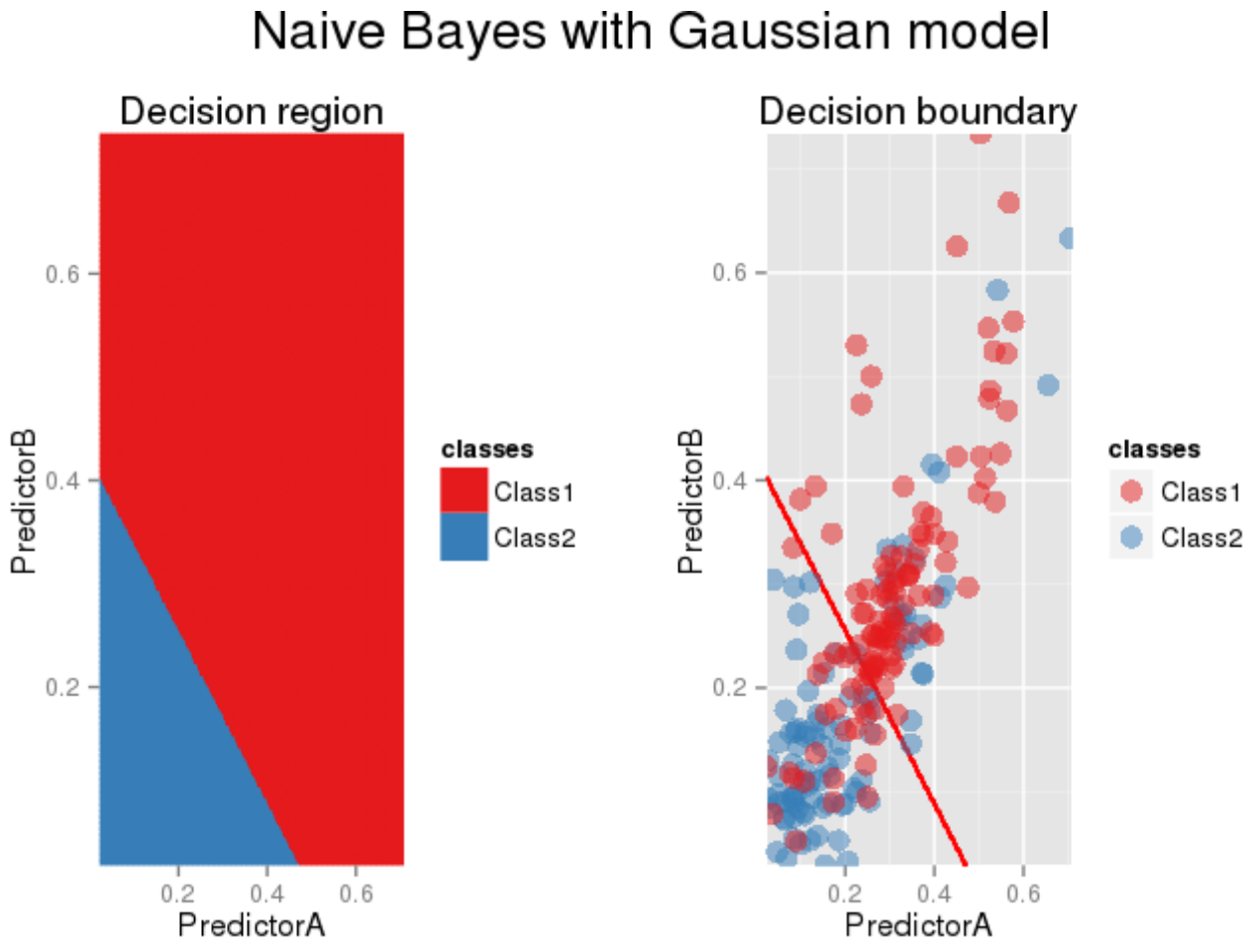
Naive Bayes

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

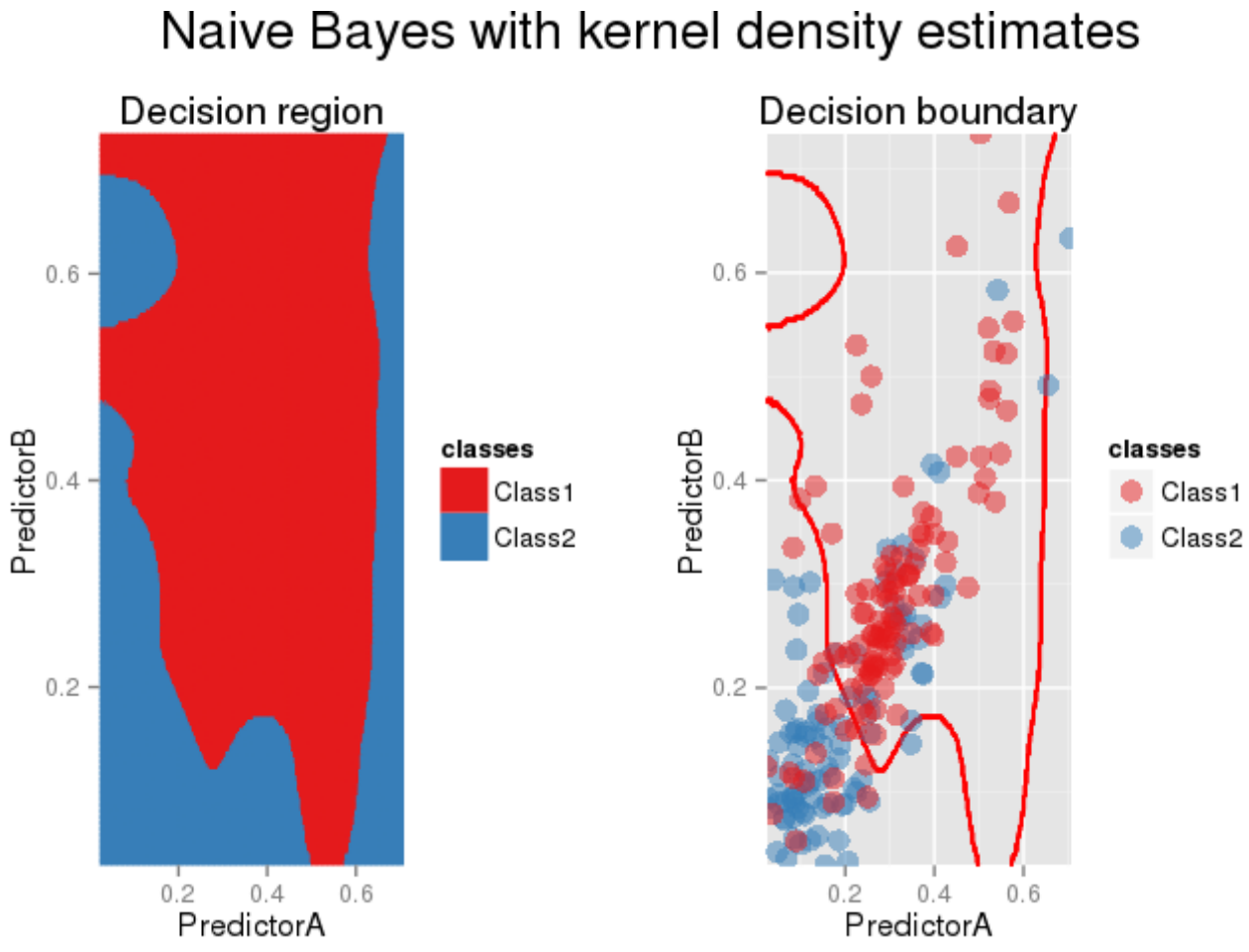
$$\mathbb{P}\{\mathbf{X}|Y\} = \prod_{i=1}^{d} \mathbb{P}\left\{\mathbf{X}^{(i)}\middle|Y\right\}$$

  - 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



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 $Y|\mathbf{X}$ by a parametric conditional density $\mathbb{P}_\theta\{Y|\mathbf{X}\}$.

Maximum Likelihood Approach

- Classical choice for θ:

$$\widehat{\theta} = \underset{\theta}{\text{argmin}} - \sum_{i=1}^{n} \log \mathbb{P}_\theta\{Y_i|\mathbf{X}_i\}$$

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

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

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

Linear Conditional Density Models

Linear Models

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

$$\mathbb{P}_\theta \{Y|\mathbf{X}\} = \mathbb{P}_{\mathbf{X}^t\beta,\varphi} \{Y\}$$

- **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 $\mathbb{P}\{Y = +1|\mathbf{X}\}$ by $h(\beta^T\mathbf{X} + \beta_0)$ with h non decreasing.
- $h(\beta^T\mathbf{X} + \beta_0) > 1/2 \Leftrightarrow \beta^T\mathbf{X} + \beta_0 - h^{-1}(1/2) > 0$
- Linear Classifier: $\text{sign}(\beta^T\mathbf{X} + \beta_0 - h^{-1}(1/2))$

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, $Y|\mathbf{X}$ follows $\mathcal{B}(\mathbb{P}\{Y = +1|\mathbf{X}\})$
- Approximation of $Y|\mathbf{X}$ by $\mathcal{B}(h(\beta^T\mathbf{X} + \beta_0))$
- *Natural* probabilistic choice for β: β minimizing the distance between $\mathcal{B}(h(\mathbf{X}^t\beta))$ and $\mathcal{B}(\mathbb{P}\{Y = 1|\mathbf{X}\})$.

KL Distance

- *Natural* distance: Kullback-Leibler divergence

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

Maximum Likelihood Estimate

log-likelihood

- KL:

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

- Empirical counterpart = opposite of the log-likelihood:

$$-\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{Y_i=1}\log(h({\mathbf{X}_i}^t\beta)) + \mathbf{1}_{Y_i=-1}\log(1-h({\mathbf{X}_i}^t\beta))\right)$$

- 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:

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

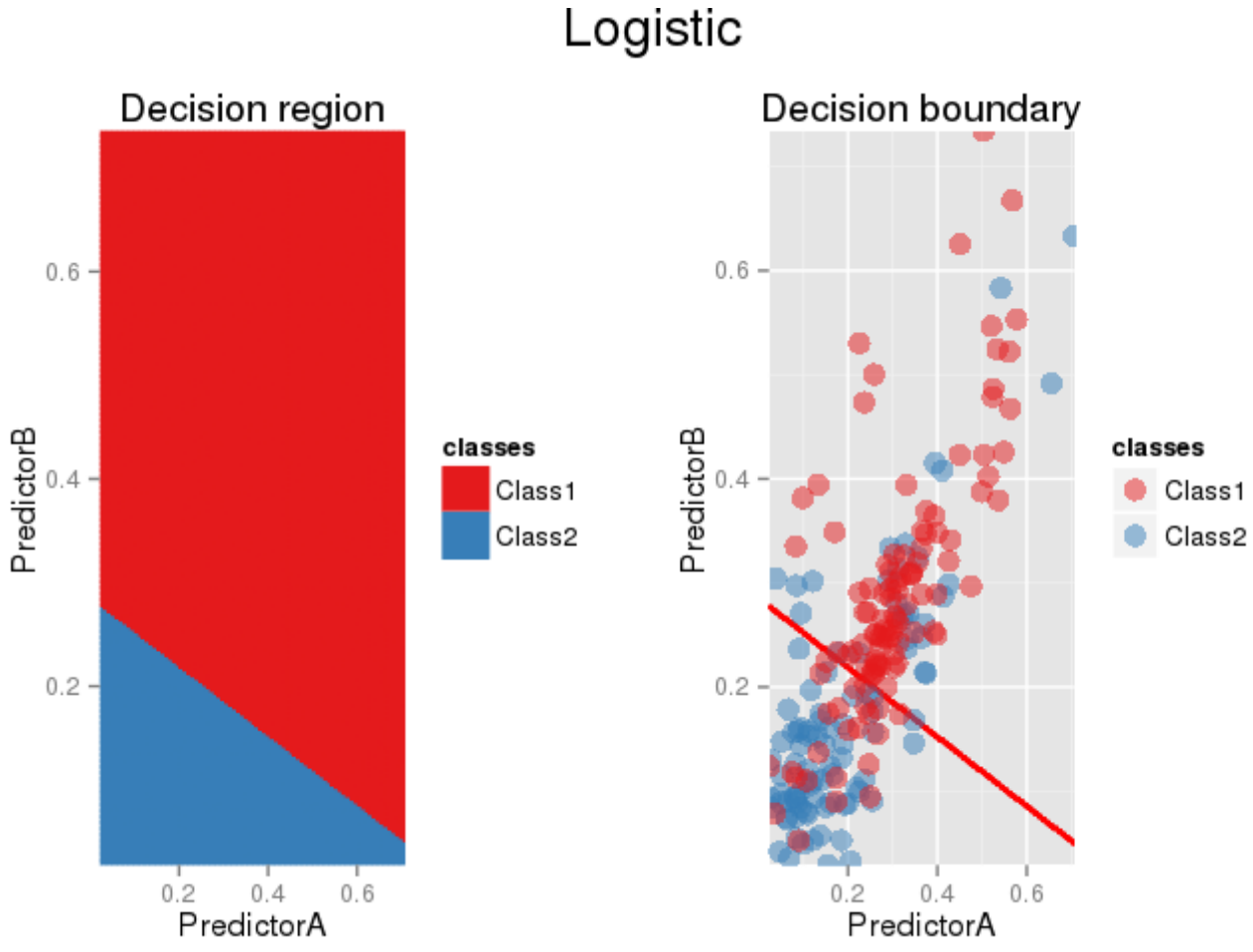
Likelikood Rewriting

- Opposite of the log-likelihood:

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

- Convex and smooth function of β
- Easy optimization.

Example: Logistic



Feature Design

Transformed Representation

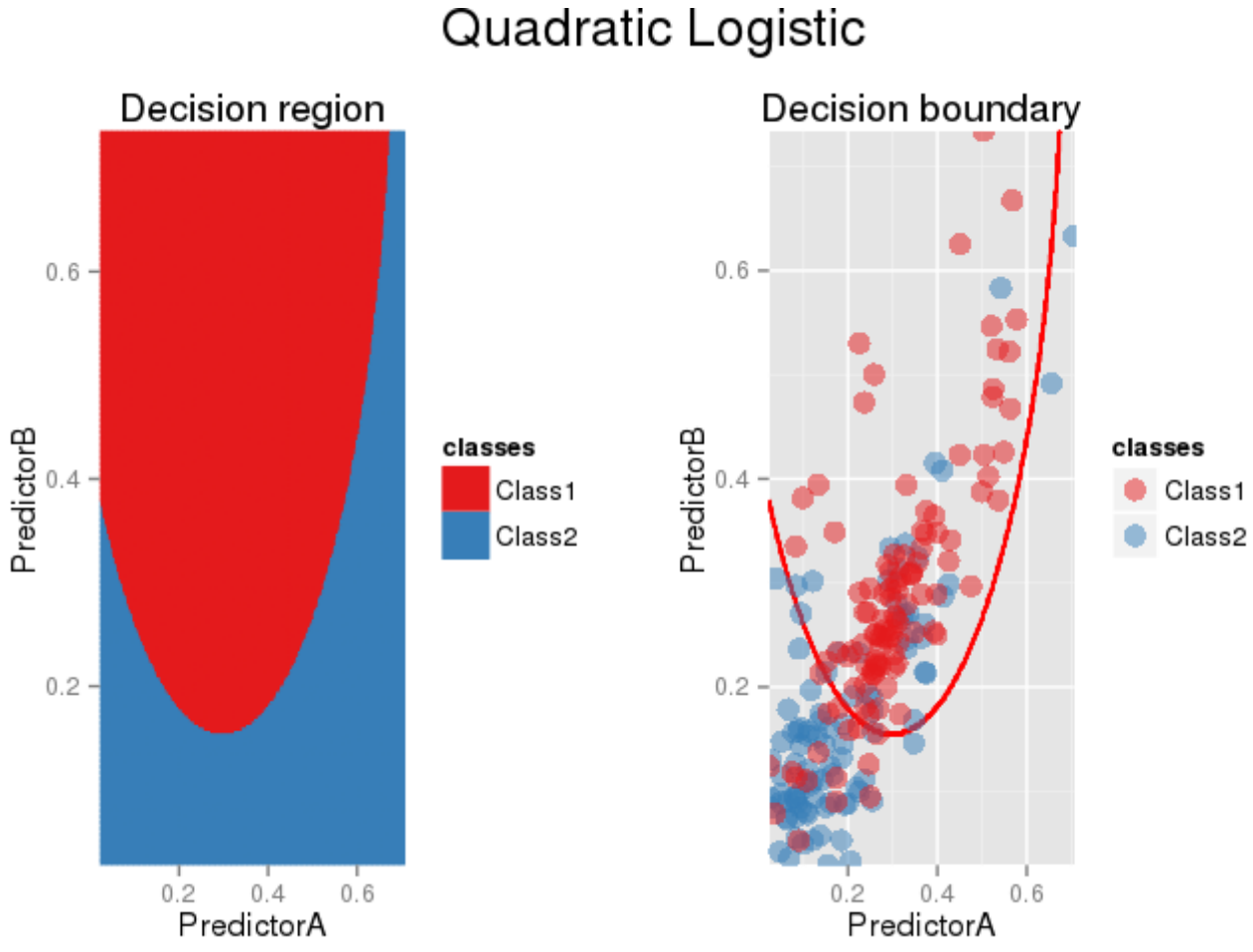
- From $\mathbf{X}$ to $\Phi(\mathbf{X})$!
- New description of $\mathbf{X}$ leads to a different linear model:

$$f_\beta(\mathbf{X}) = \Phi(\mathbf{X})^t \beta$$

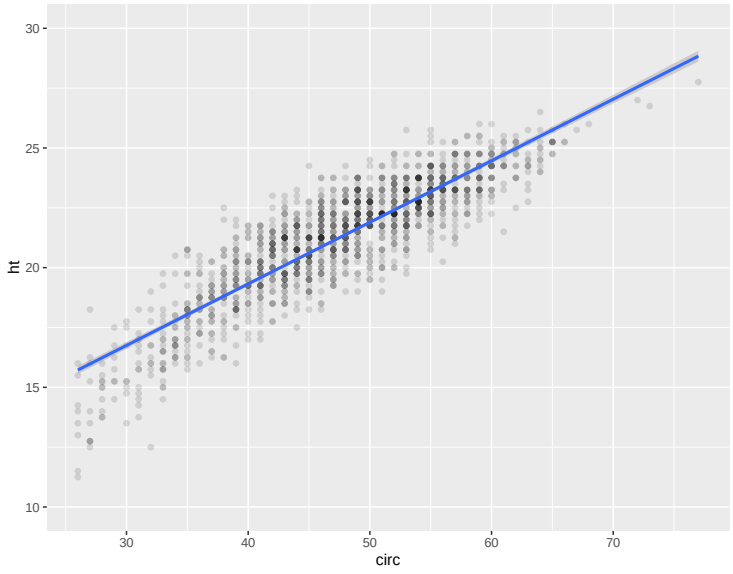
Feature Design

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

- In regression, estimation of $\mathbb{E}[Y|\mathbf{X}]$ 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: $Y|\mathbf{X} \sim P_{\mathbf{X}^t\beta}$...

- ML fit of the parameters

Outline

Non Parametric Conditional Estimation

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

Non Parametric Conditional Estimation

- Two heuristics:
  - $Y|\mathbf{X}$ (or $\mathbb{E}[Y|\mathbf{X}]$) is almost constant (or simple) in a neighborhood of $\mathbf{X}$. (Kernel methods)
  - $Y|\mathbf{X}$ (or $\mathbb{E}[Y|\mathbf{X}]$) 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

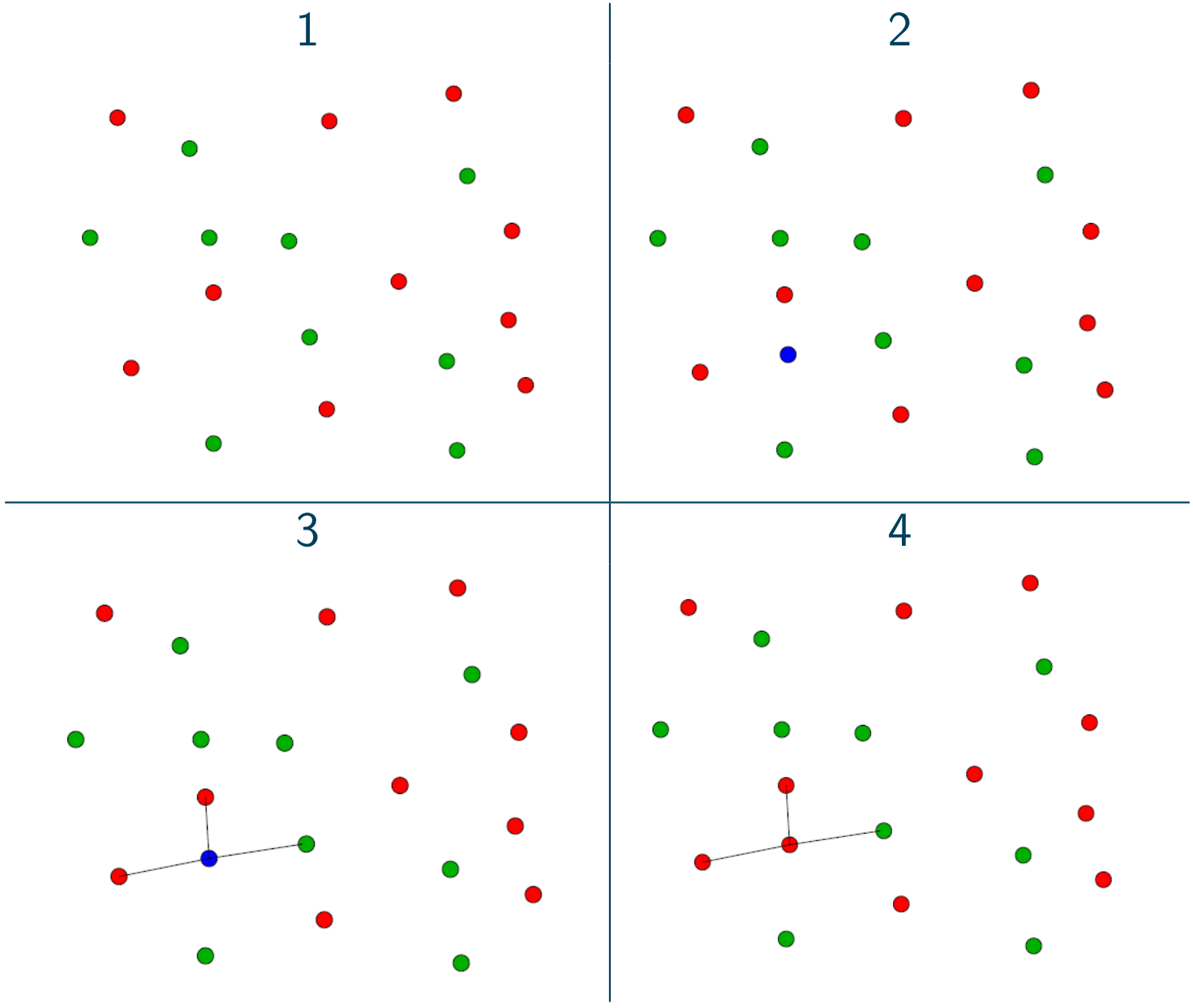
- **Idea:** The behavior of $Y|\mathbf{X}$ is locally *constant* or simple!

Kernel

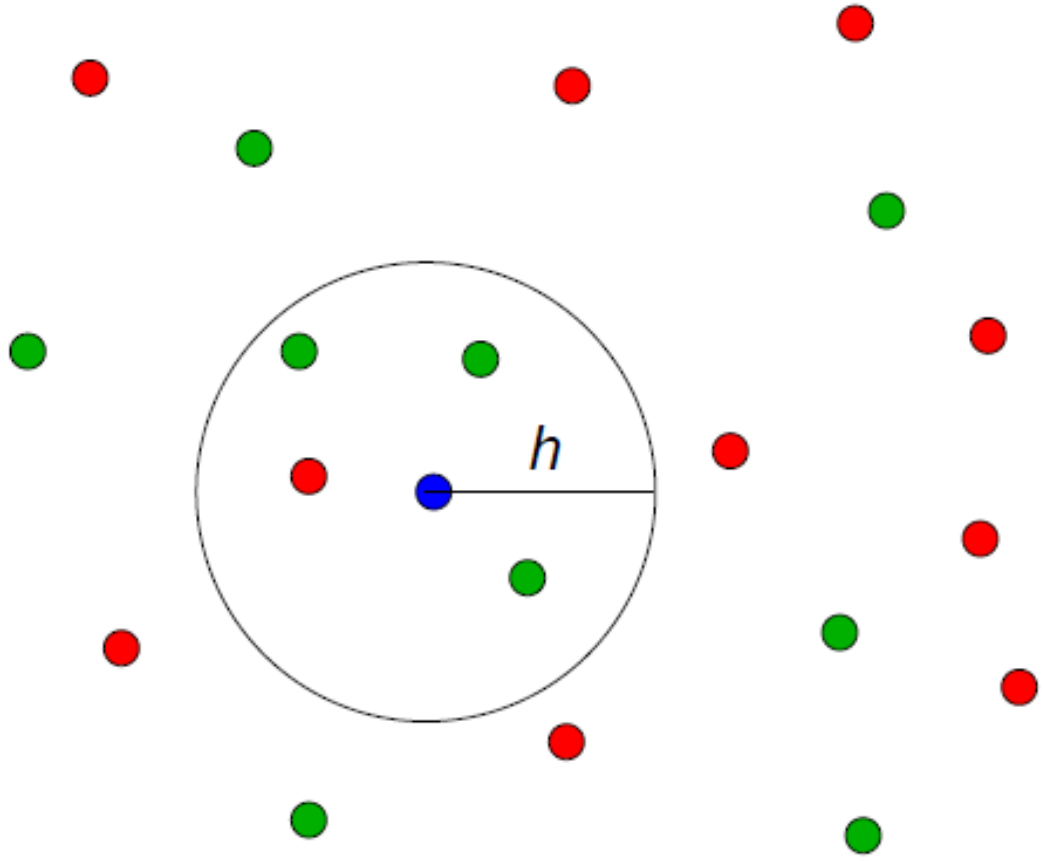
- Choose a kernel K (think of a weighted neighborhood).
- For each $\widetilde{\mathbf{X}}$, compute a simple localized estimate of $Y|\mathbf{X}$
- Use this local estimate to take the decision

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

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



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



k Nearest-Neighbors

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

k-NN as local conditional density estimate

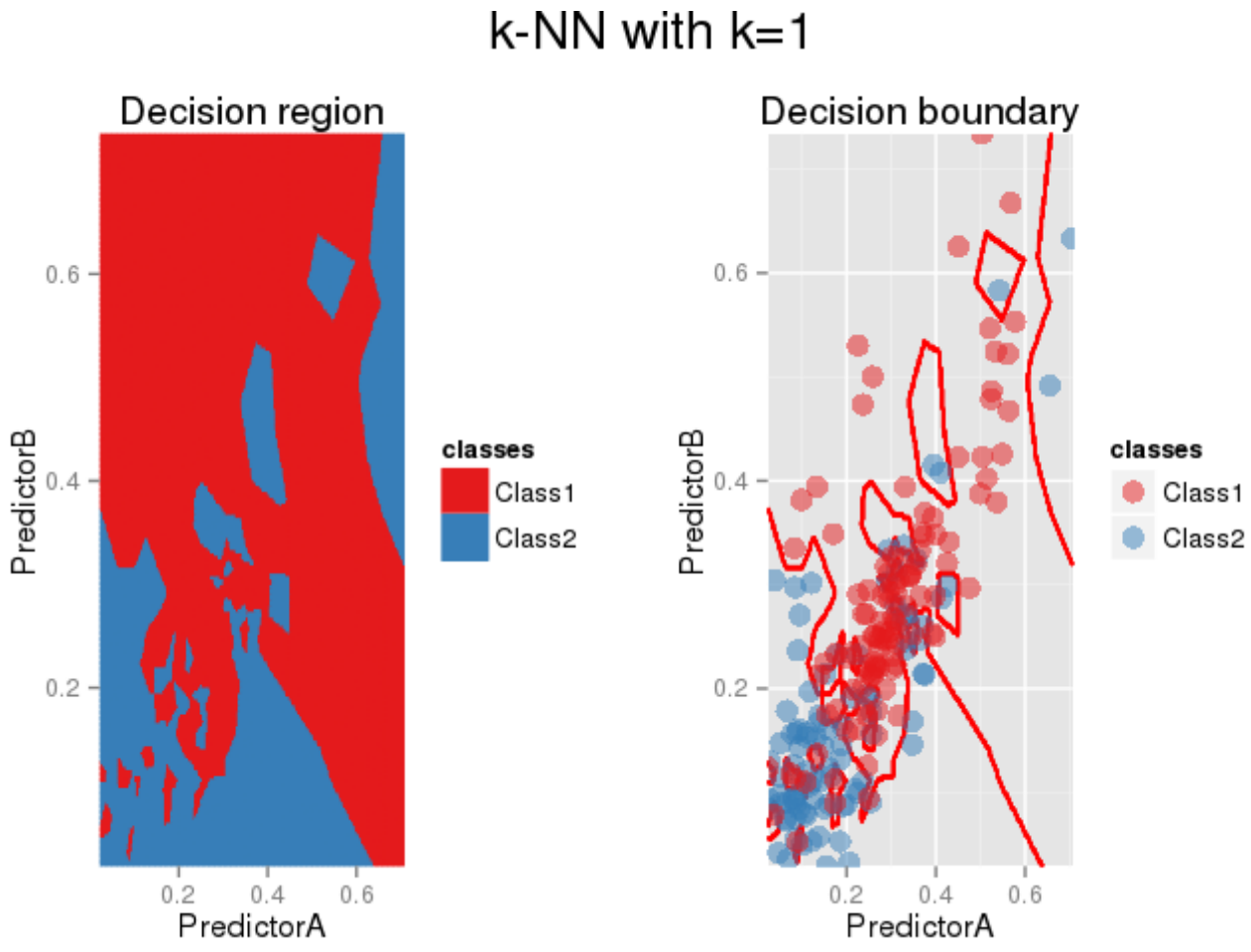
$$\mathbb{P}\left\{\widehat{Y=1}|\mathbf{X}\right\} = \frac{\sum_{\mathbf{x}_i \in \mathcal{V}_{\mathbf{x}}} \mathbf{1}_{\{Y_i=+1\}}}{|\mathcal{V}_{\mathbf{x}}|}$$

- KNN Classifier:

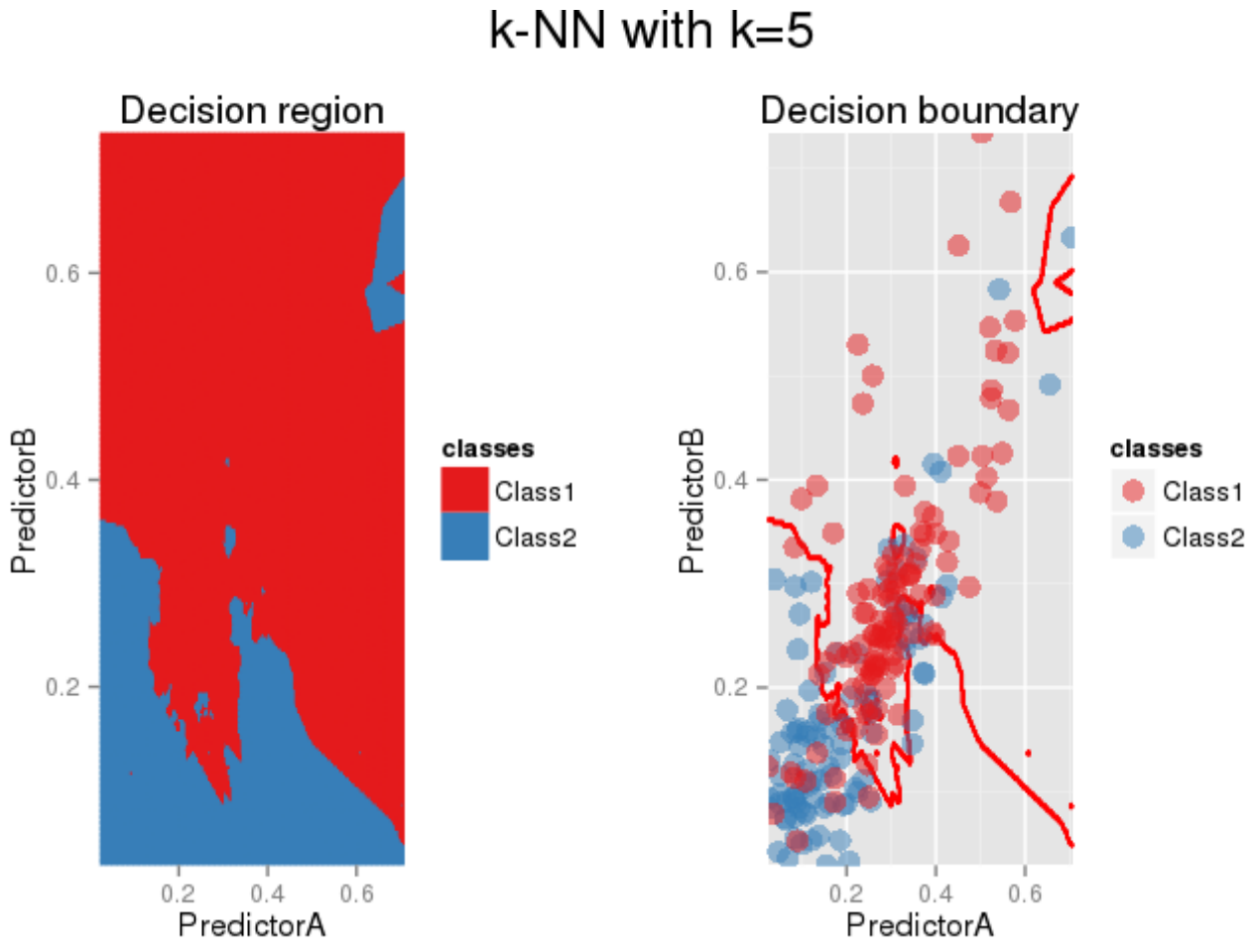
$$\widehat{f}_{KNN}(\mathbf{X}) = \begin{cases} +1 & \text{if } \mathbb{P}\left\{\widehat{Y=1}|\mathbf{X}\right\} \geq \mathbb{P}\left\{\widehat{Y=-1}|\mathbf{X}\right\} \\ -1 & \text{otherwise} \end{cases}$$

- **Remark:** You can also use your favorite kernel estimator...

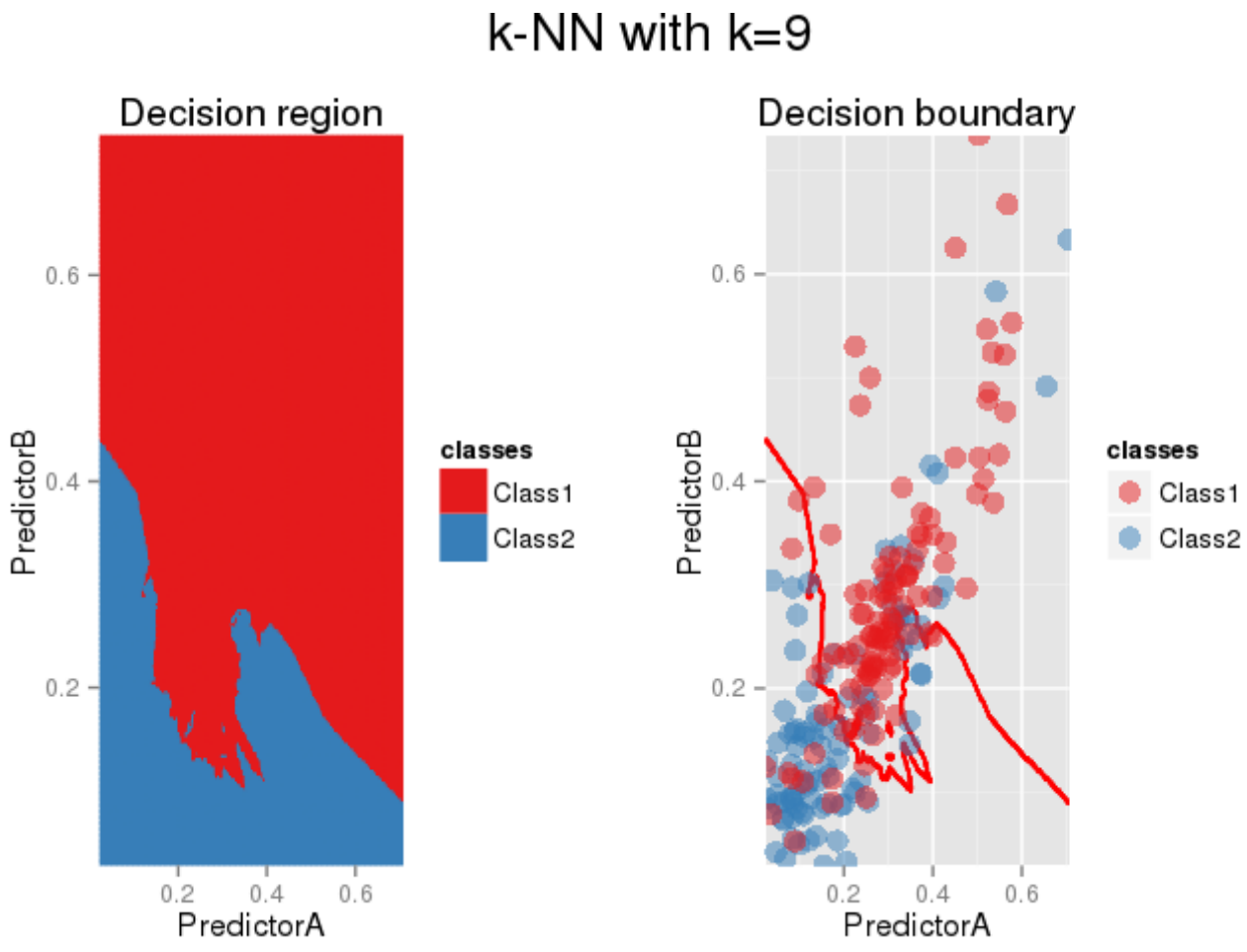
Example: KNN


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

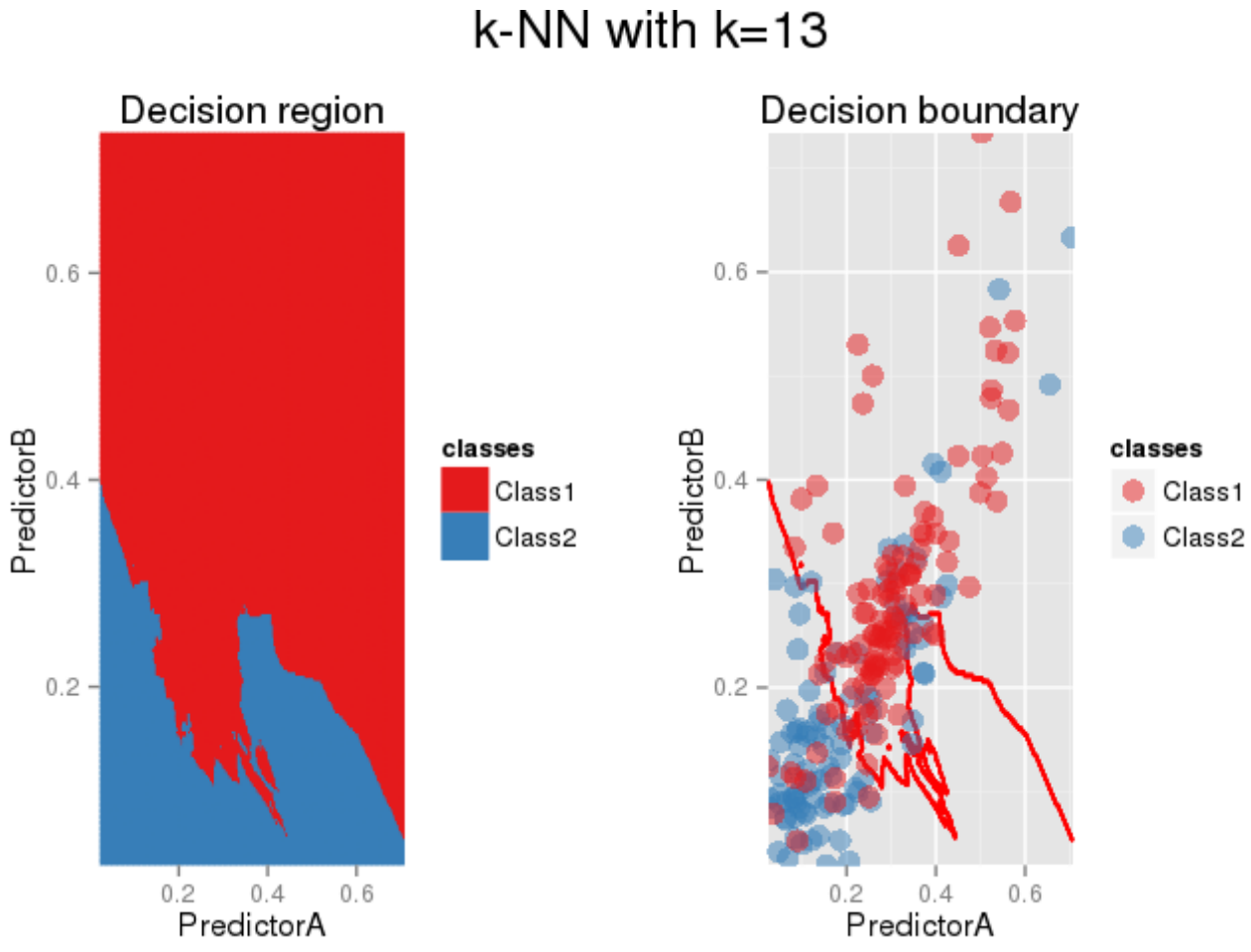
Example: KNN


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

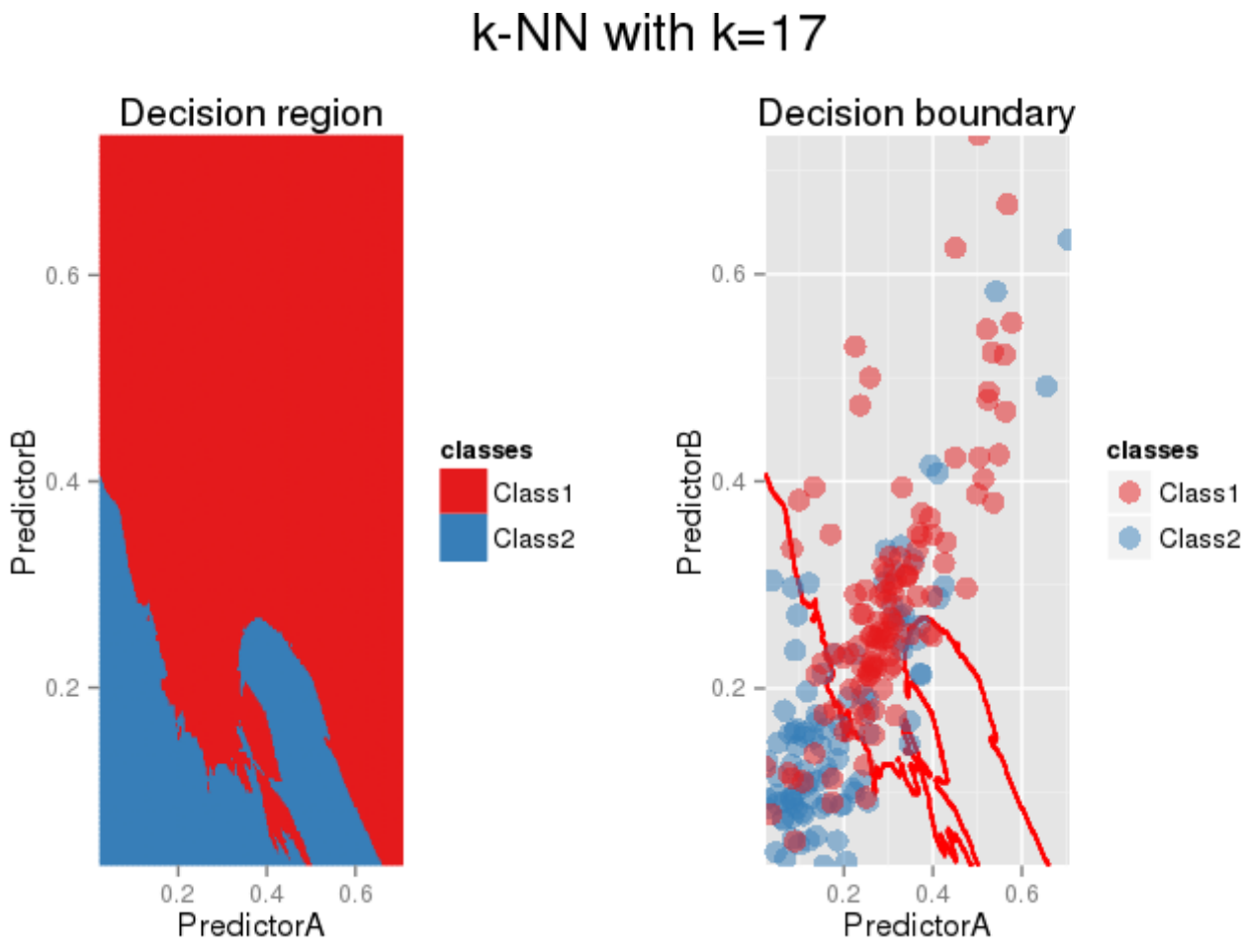
Example: KNN


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

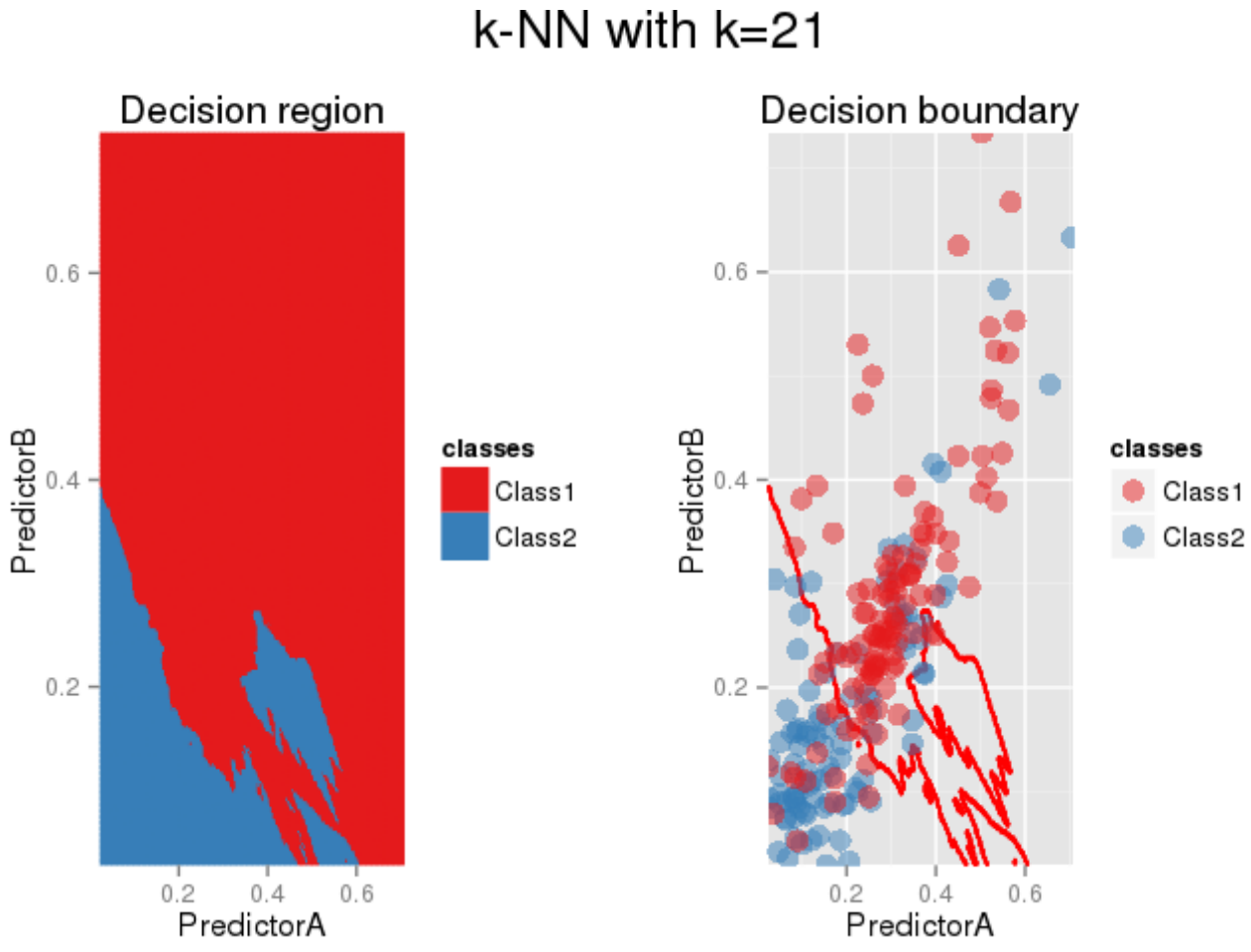
Example: KNN


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

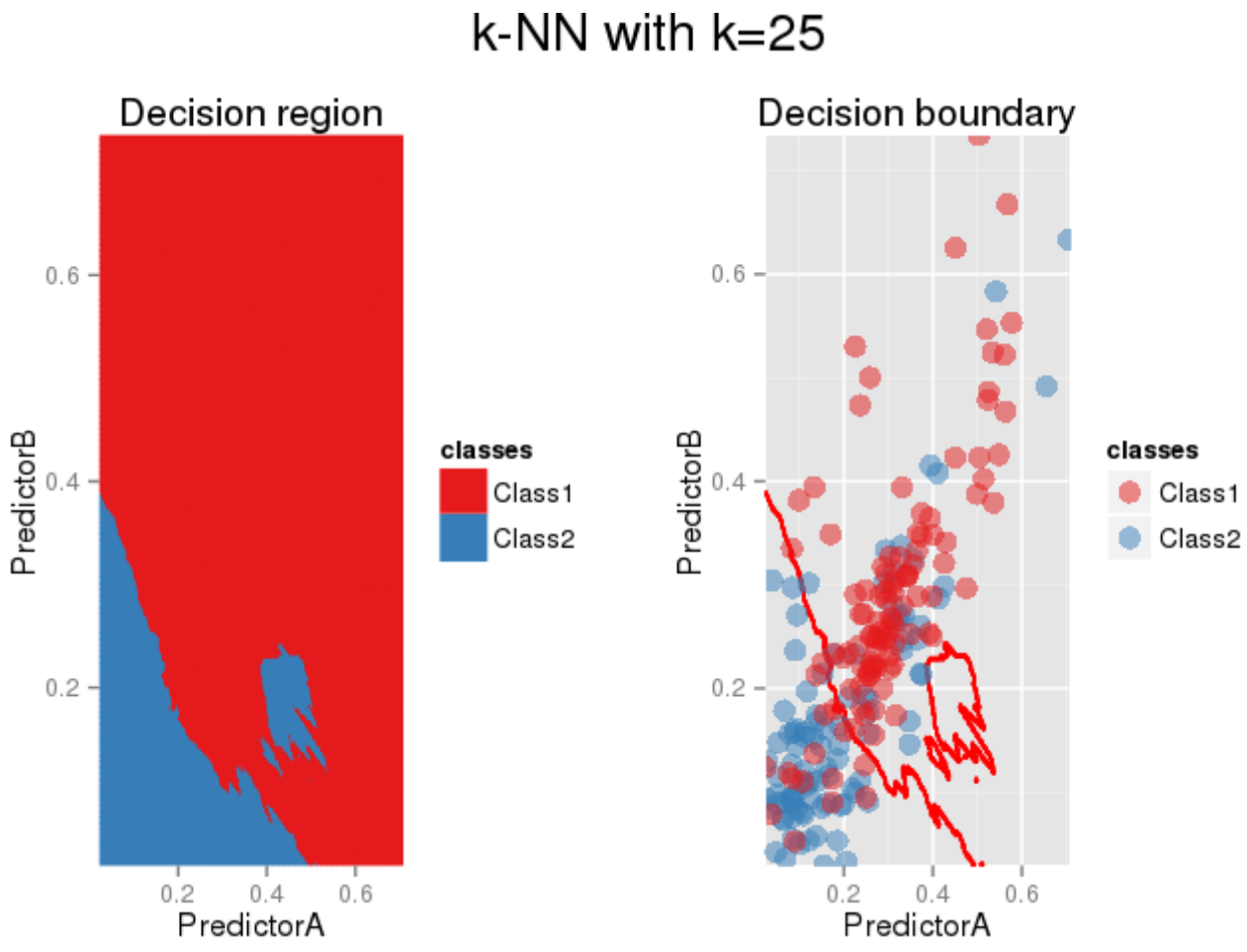
Example: KNN


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

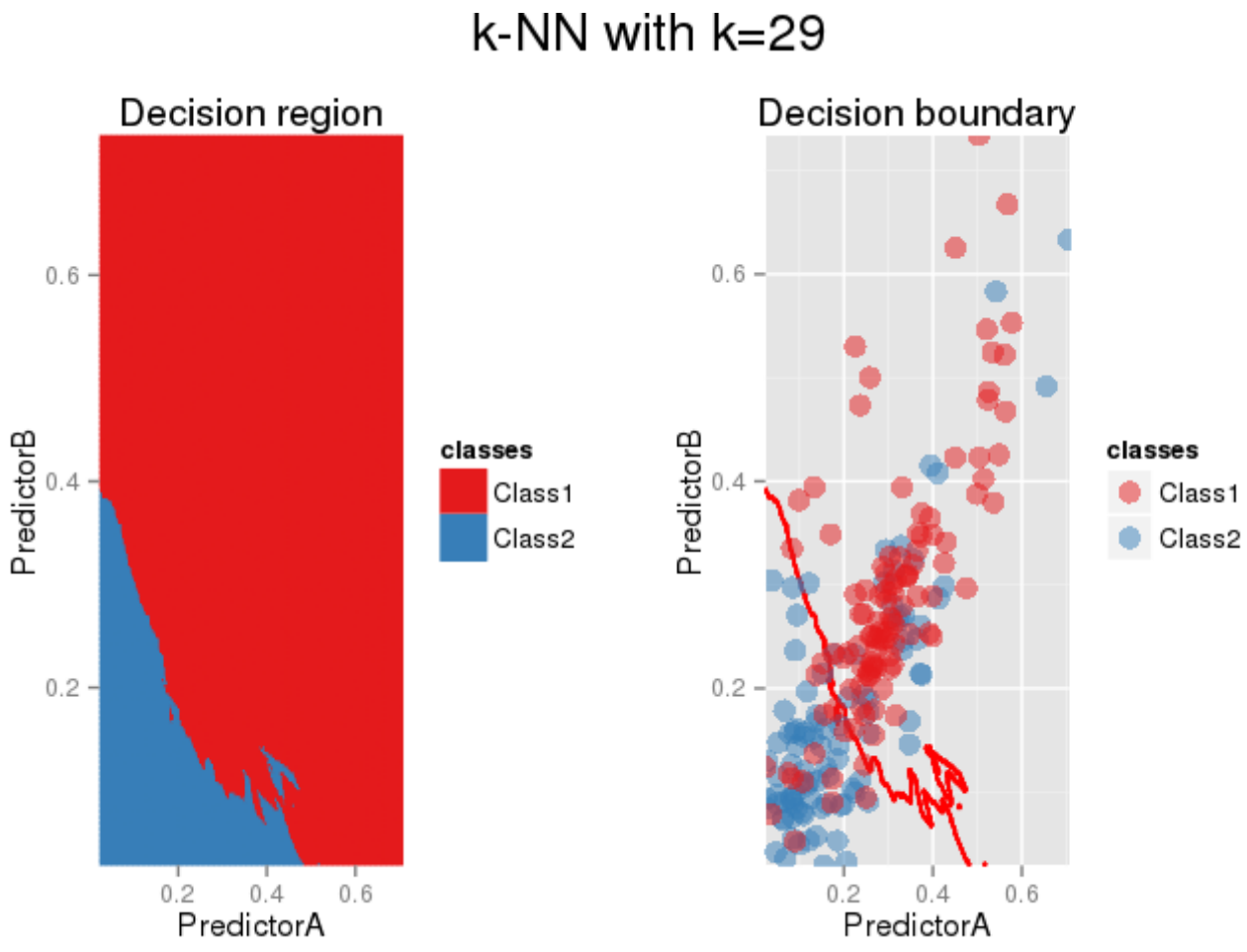
Example: KNN


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

Regression and Local Averaging

A naive idea

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

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

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

- Heuristic:
  - On the one hand, if $\mathbf{X} \to \mathbb{E}[Y|\mathbf{X}]$ is regular then
    $$\mathbb{E}[Y|\mathbf{X}] \simeq \mathbb{E}\left[\mathbb{E}\left[Y|\mathbf{X}'\right]|\mathbf{X}' \in \mathcal{N}(\mathbf{X})\right] = \mathbb{E}\left[Y|\mathbf{X}' \in \mathcal{N}(\mathbf{X})\right]$$
  - On the other hand,
    $$\mathbb{E}\left[Y|\mathbf{X}' \in \mathcal{N}(\mathbf{X})\right] \simeq \frac{1}{|\{\mathbf{X}_i \in \mathcal{N}(\mathbf{X})\}|} \sum_{\mathbf{X}_i \in \mathcal{N}(\mathbf{X})} Y_i$$

Regression and Local Averaging

Neighborhood and Size

- Most classical choice: $\mathcal{N}(\mathbf{X}) = \{\mathbf{X}', \|\mathbf{X} - \mathbf{X}'\| \leq h\}$ where $\|.\|$ is a (pseudo) norm and h a size (bandwidth) parameter.
- In principle, the norm and h could vary with $\mathbf{X}$, 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 $\mathbb{E}[Y|\mathbf{X}] \simeq \mathbb{E}\left[Y|\mathbf{X}' \in \mathcal{N}(\mathbf{X})\right]$ is more accurate.

Weighted Local Averaging

Weighted Local Average

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

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

Kernel

- Most classical choice: $w(\mathbf{X}, \mathbf{X}') = K\left(\frac{\mathbf{X}-\mathbf{X}'}{h}\right)$ 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.

Link with Density Estimation

Density Estimation

- How to estimate the density p of $\mathbf{X}$ with respect to the Lebesgue measure from an i.i.d. sample $(\mathbf{X}_1, \ldots, \mathbf{X}_n)$.
- Parametric approach: assume that the density has a known parametrized shape and estimate those parameters.
- Nonparametric approach: do not assume that the density has a known parametrized shape and
  - Approximate it by a parametric one, whose parameters can be estimated
  - Estimate directly the density
- Important nonparametric statistic topic!

Link with Density Estimation

Kernel Density Estimation (Parzen)

- Choose a positive kernel K such that $\int K(x)dx = 1$
- Use as an estimate

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

- If $K = \frac{1}{Z_h}\mathbf{1}_{\|t\| \leq h}$ this can be easily interpreted in term of local empirical density of samples!
- General K corresponds to the same smoothing idea we have used before.
- We will often use $K_h(t) = \frac{1}{h^d}K(t/h)$ and let

$$\widehat{p}_h(\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\mathbf{X} - \mathbf{X}_i)$$

Link with Density Estimation

Bandwidth choice

- A small h leads to a small bias but a large variance...
- A large h leads to a small variance but a large bias...
- Theoretical analysis possible!

Nadaraya-Watson Heuristic

- Provided all the densities exist

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

- Replace the unknown densities by their estimates:

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

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

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

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

From Density Estimation to Regression

Nadaraya-Watson

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

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

- Same local weighted average estimator!

Bandwidth Choice

- Bandwidth h of K allows to balances between bias and variance.
- Similar but more complex theoretical analysis of the error is possible.
- Same conclusion: the smoother the densities the easier the estimation but the optimal bandwidth depends on the unknown regularity!

Local Linear Estimation

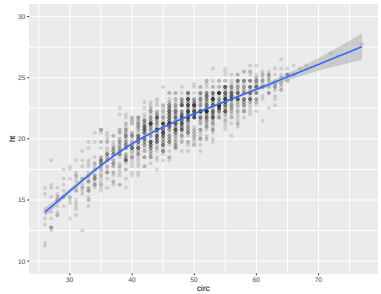
Another Point of View on Kernel

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

Local Linear Model

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

LOESS: LOcal polynomial regrESSion



1D Nonparametric Regression

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

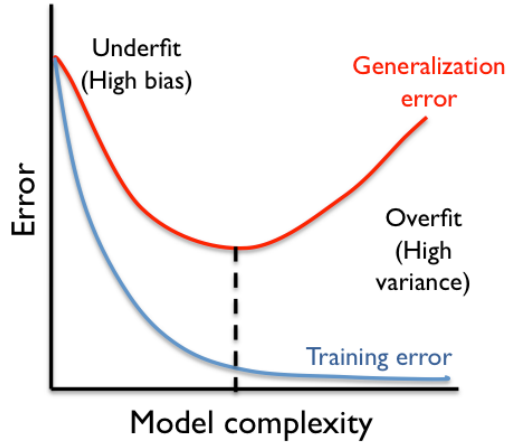
$$\sum_{i=1}^{n} |Y_i - \sum_{j=1}^{d} \beta \mathbf{X}_i^j|^2 K_h(\mathbf{X} - \mathbf{X}_i).$$

- 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.

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 an other criterion than the training error!

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.
- **Penalization approach:** use empirical loss criterion but penalize it by 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 penalized 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 $(1-\epsilon)n$ observations to train and ϵn to verify!
- 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.

Hold Out

Principle

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

$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} |Y_i - \widehat{f}^{HO}(\mathbf{X}_i)|^2$$

Model Selection by Cross Validation

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

- Does not take into account the variability of $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ ($\widehat{f}$ and error estimate)
- Biased toward simpler model as the estimation does not use

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:

$$\mathcal{R}_n^{-v}(\widehat{f}^{-v}) = \frac{1}{n_v} \sum_{(\mathbf{X}_i, Y_i) \in \mathcal{D}_v} |Y_i - \widehat{f}^{-v}(\mathbf{X}_i)|^2$$

- Compute the average empirical error:

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

- Leave One Out : V = n.

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:

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

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

- 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 $V = 10$!

Cross Validation and Confidence Interval

A Generative POV

- 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 (Quasi independence assumption)

- 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 (Small risk estimation error assumption)

- Compute the raw medians or (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.

Cross Validation

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

Cross Validation

- Use $(1-\epsilon)n$ observations to train and ϵn to verify!
- 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:
  - Leave One Out,
  - V-fold cross validation.

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:

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

- Compute the average empirical error:

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

V-fold Cross Validation

Analysis (when n is a multiple of V)

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

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

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

- Average risk for a sample of size $(1-\frac{1}{V})n$.
- Variance term much more complex to analyse!
- 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

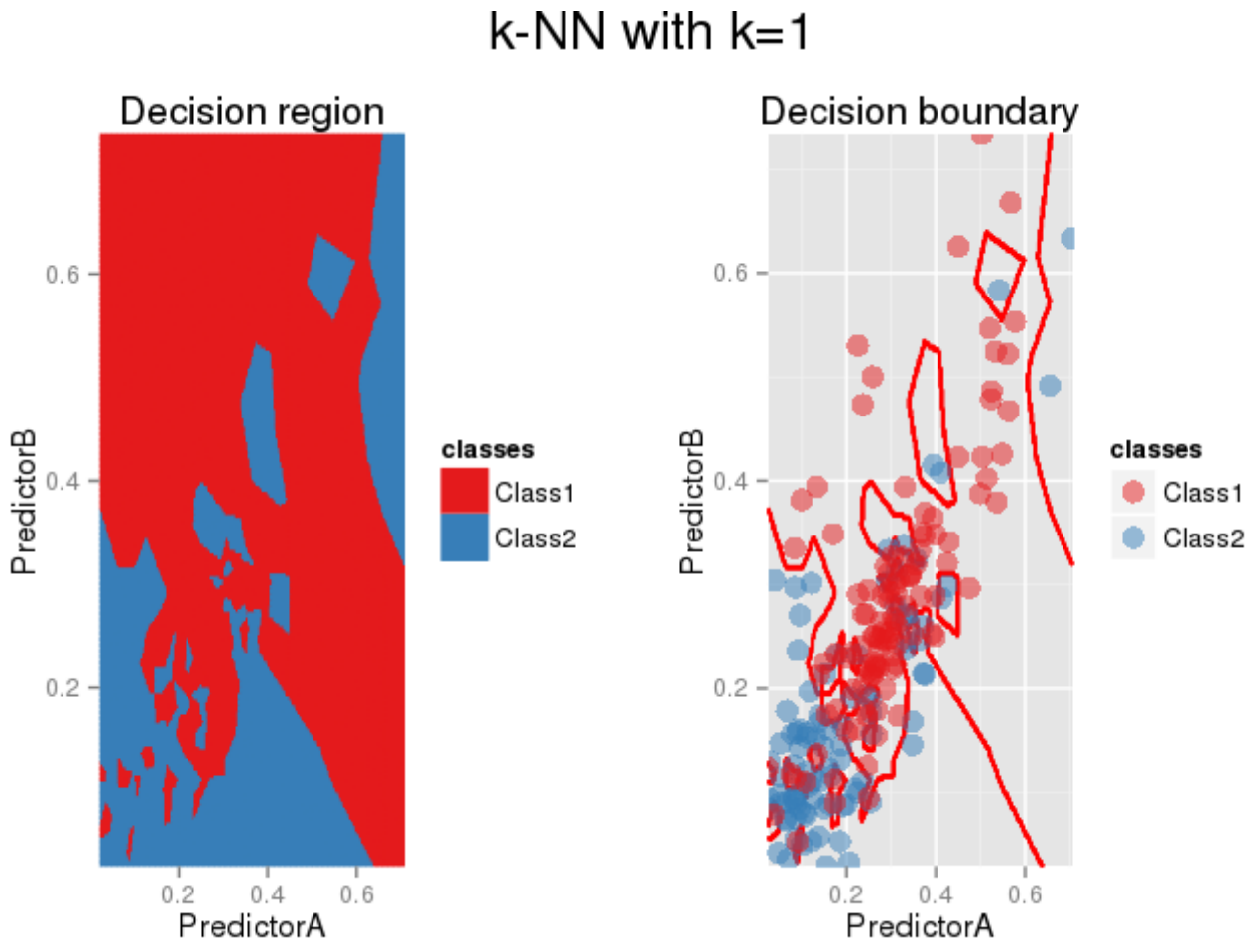
- **Prop:** for the least squares linear regression,

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

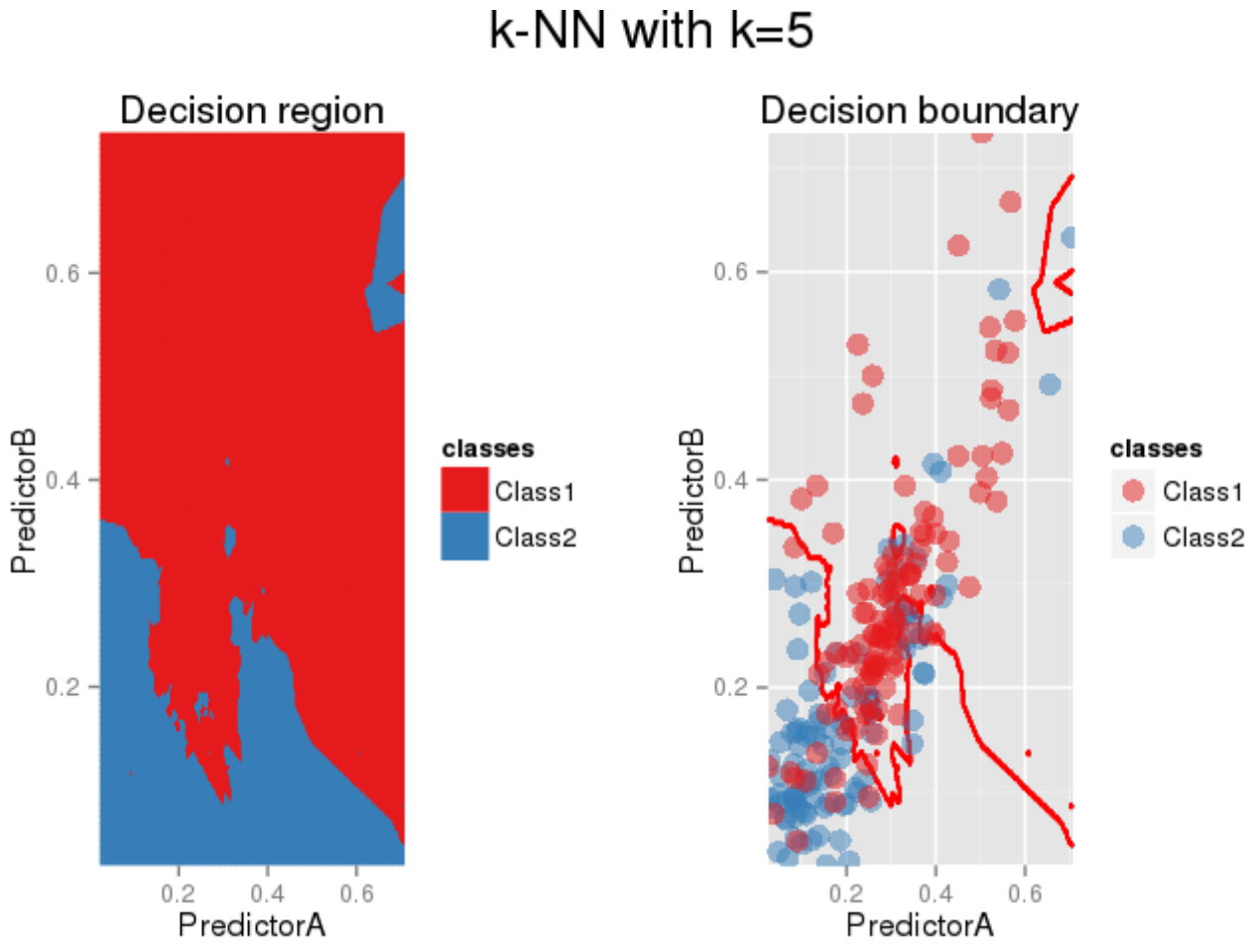
  with h_{ii} the ith diagonal coefficient of the hat (projection) matrix.
- Proof based on linear algebra!
- Leads to a fast formula for LOO:

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

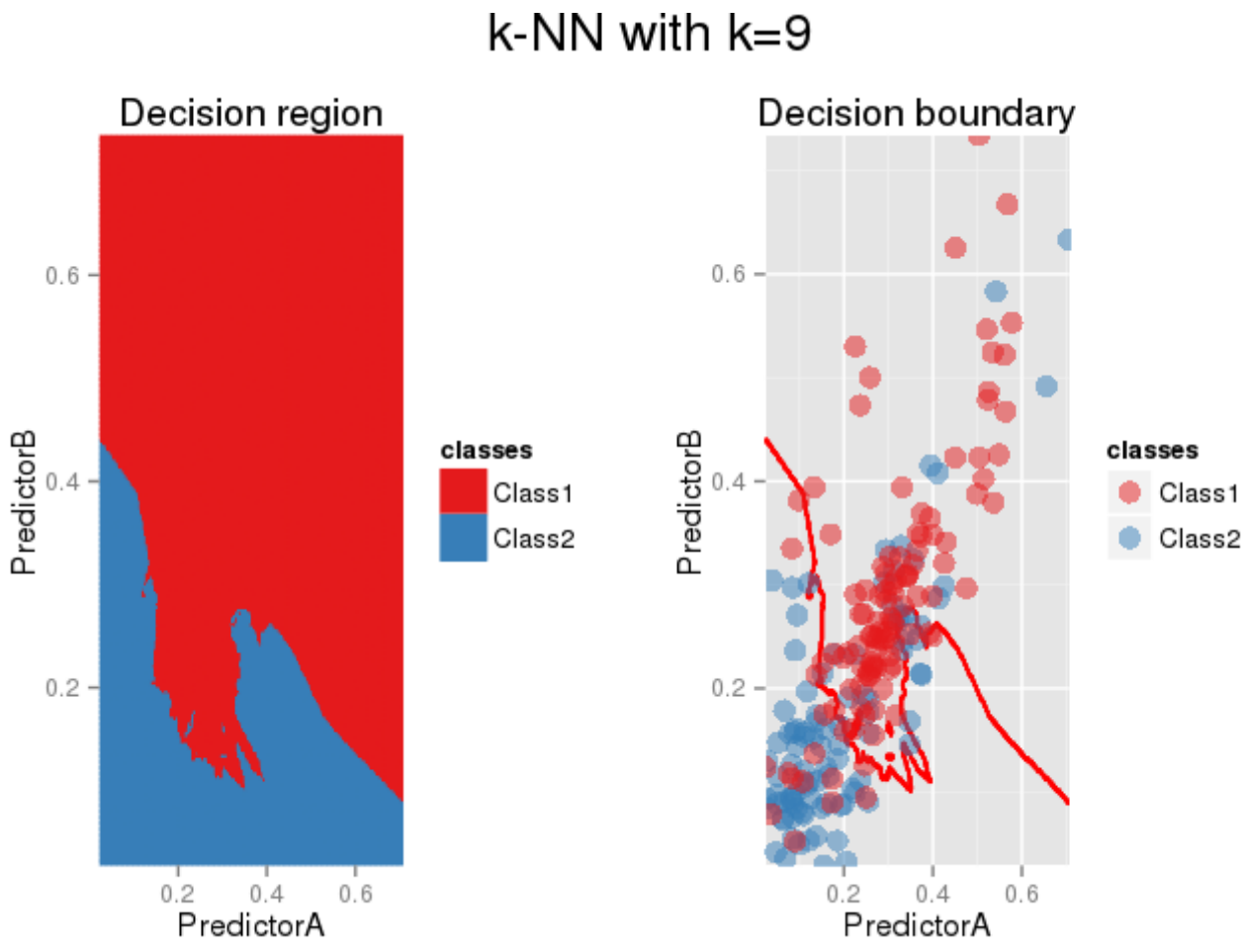
Example: KNN


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

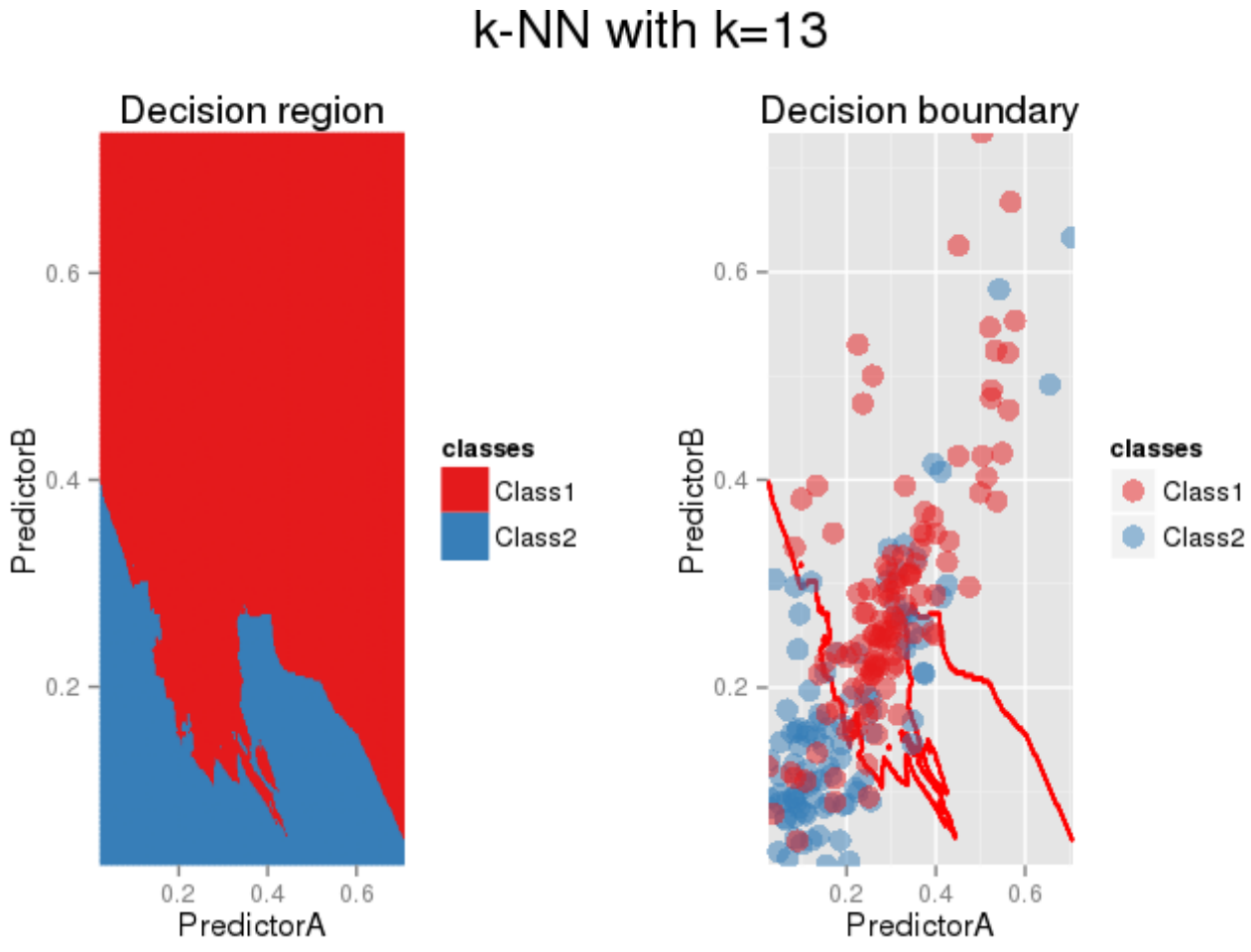
Example: KNN


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

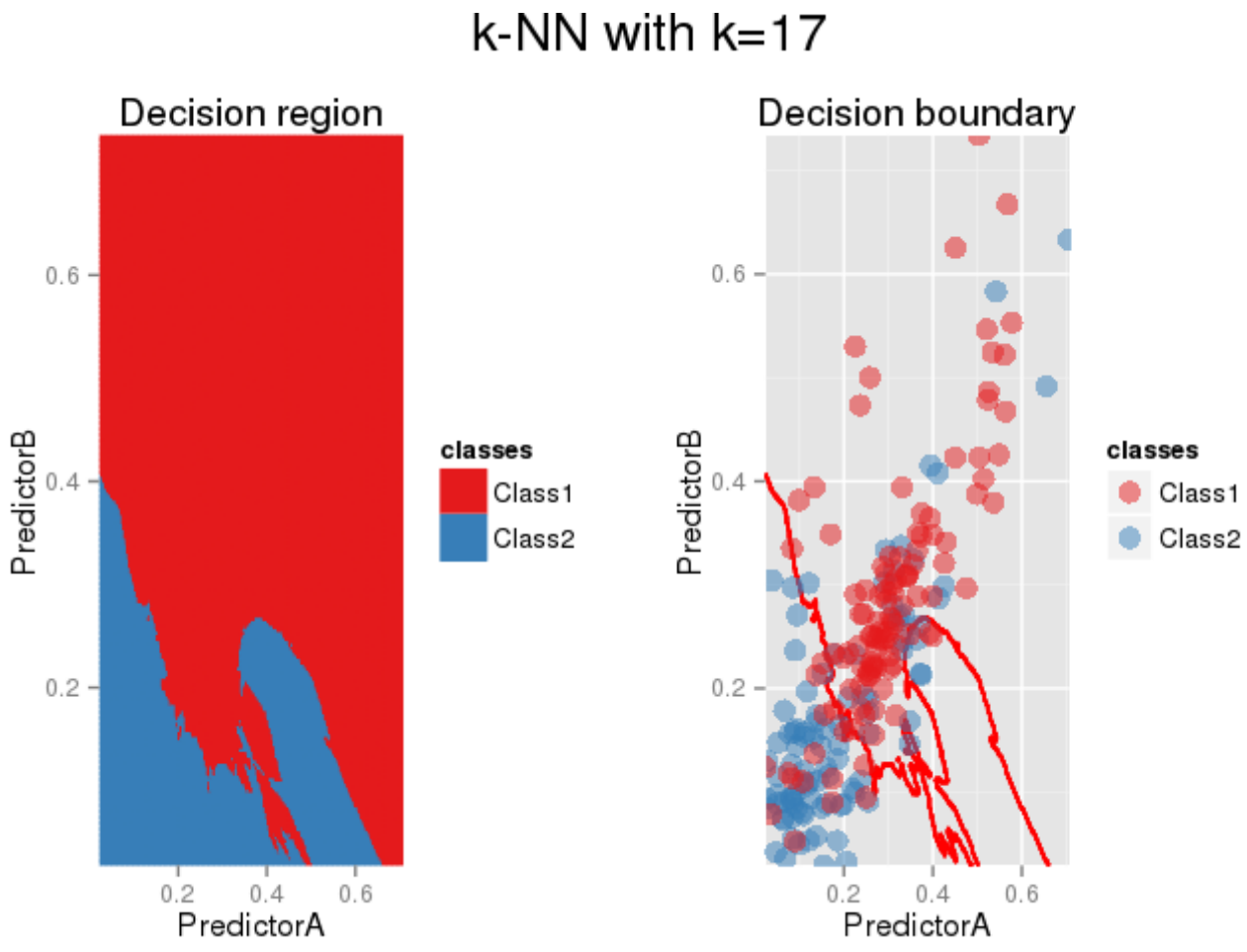
Example: KNN


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

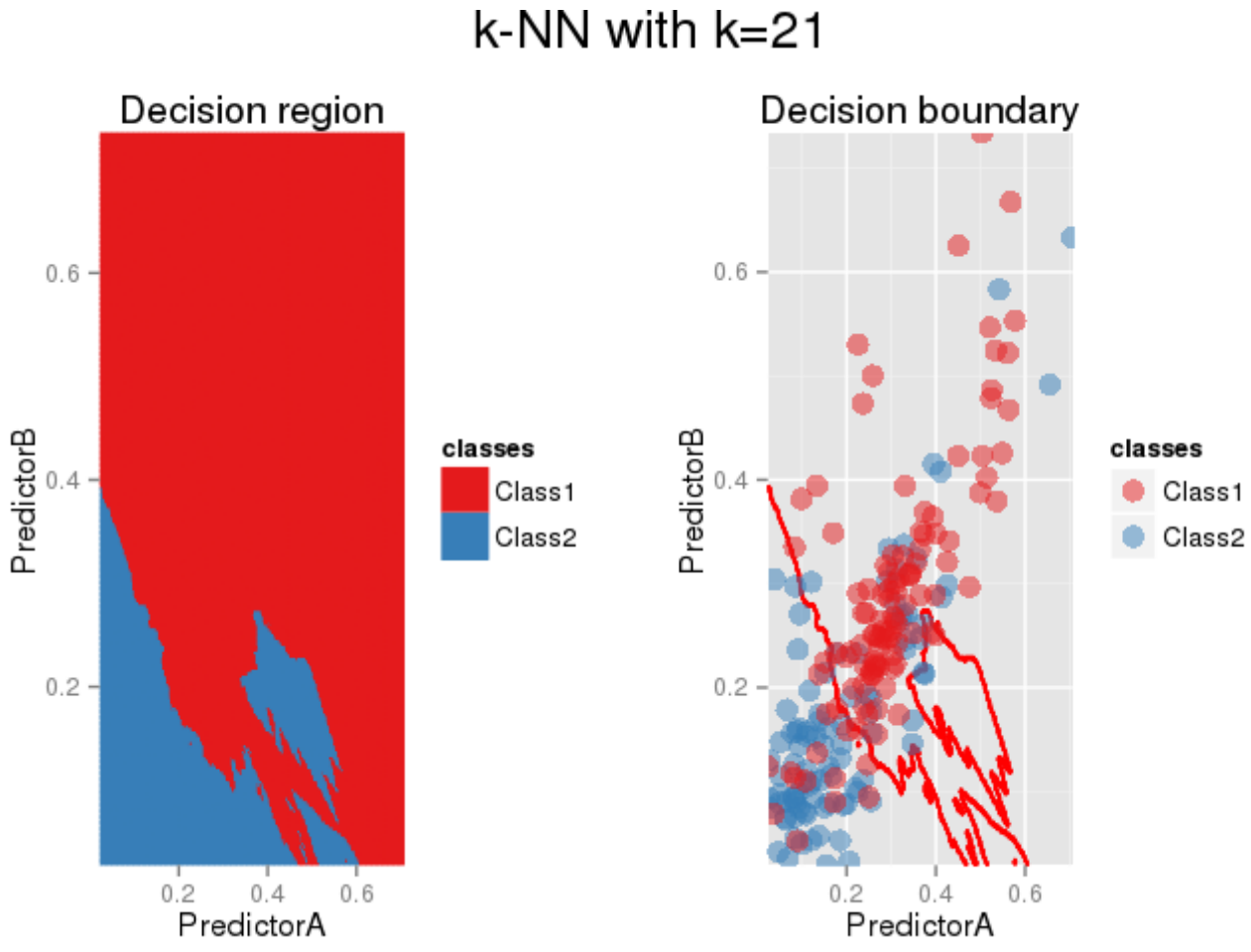
Example: KNN


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

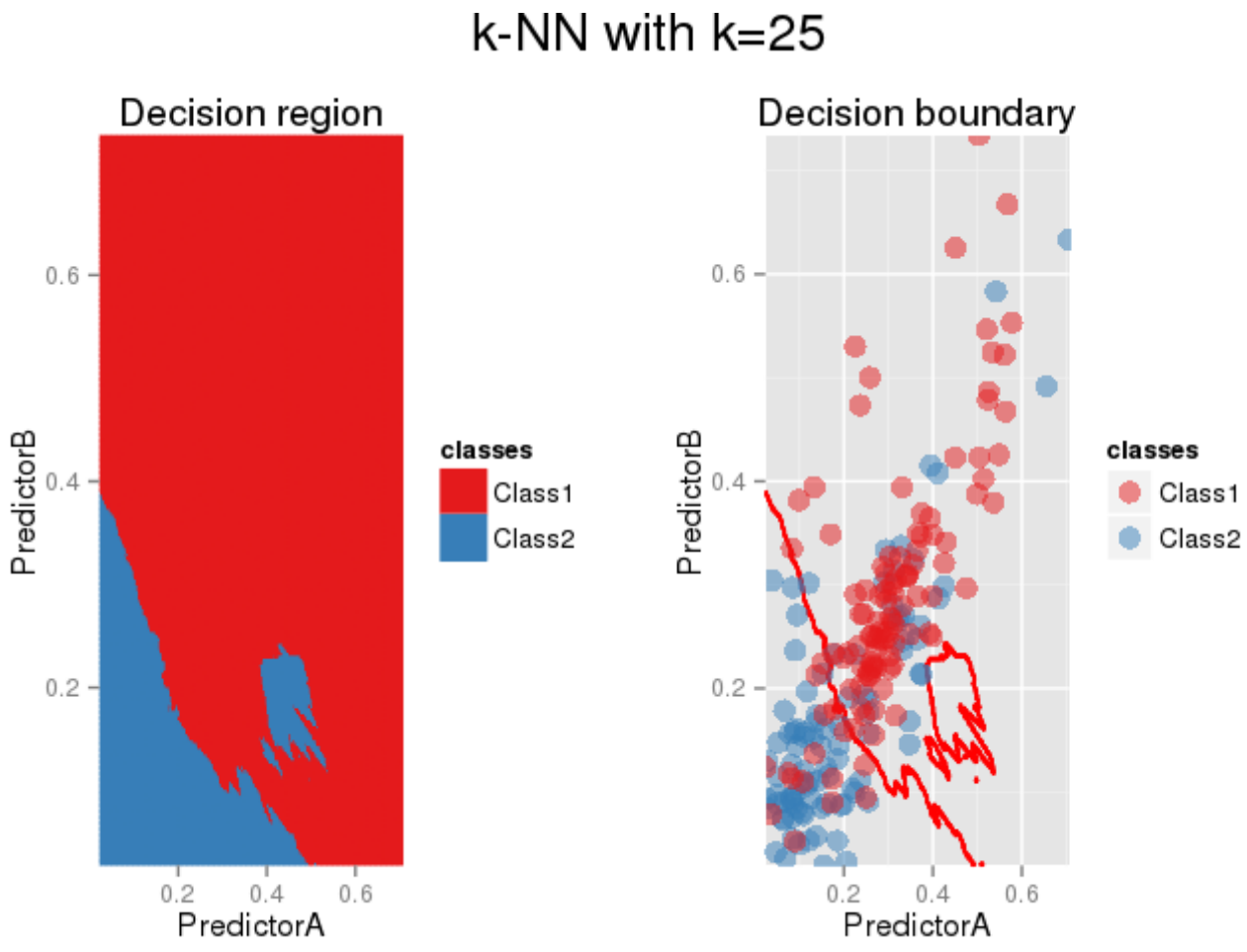
Example: KNN


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

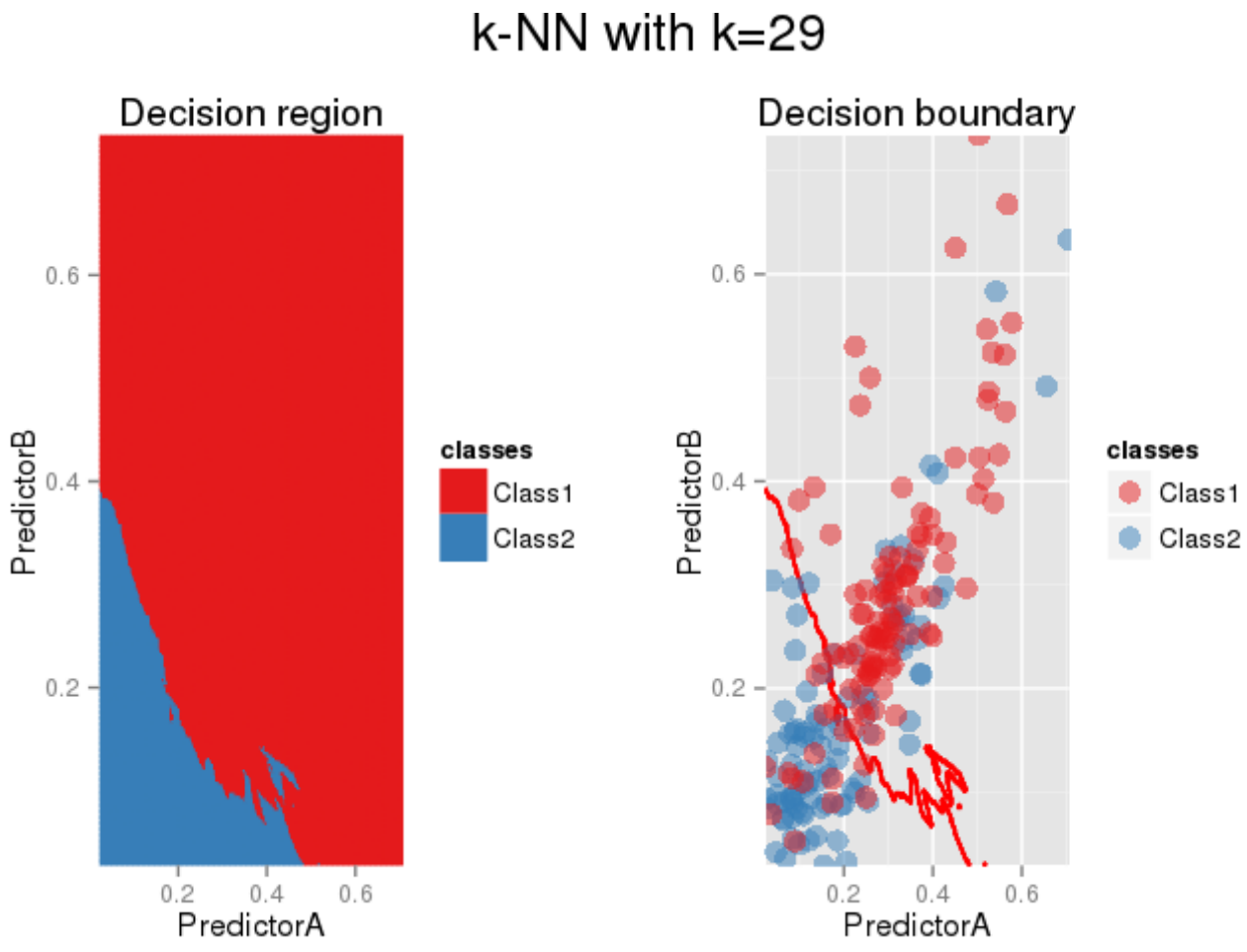
Example: KNN


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

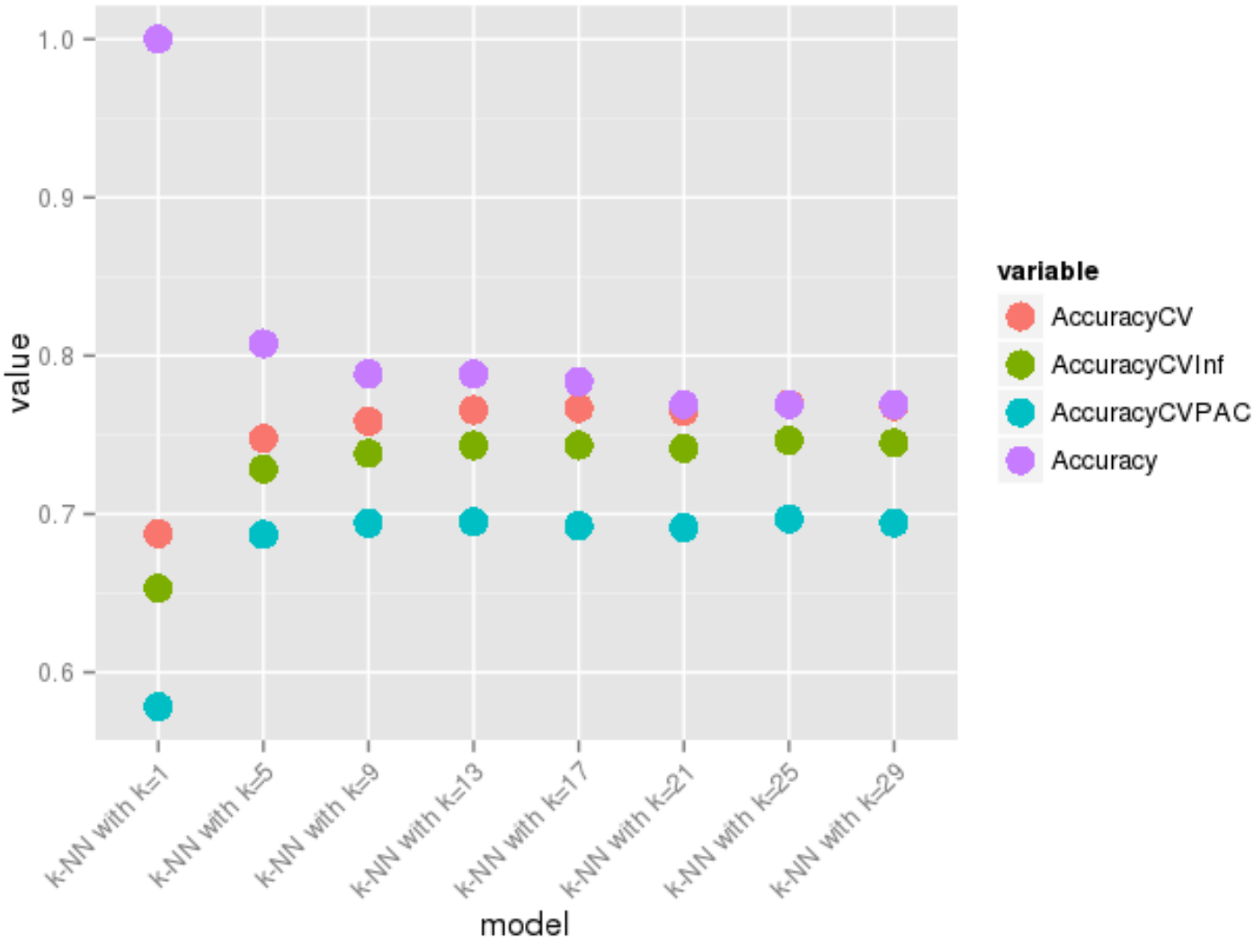
Example: KNN


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

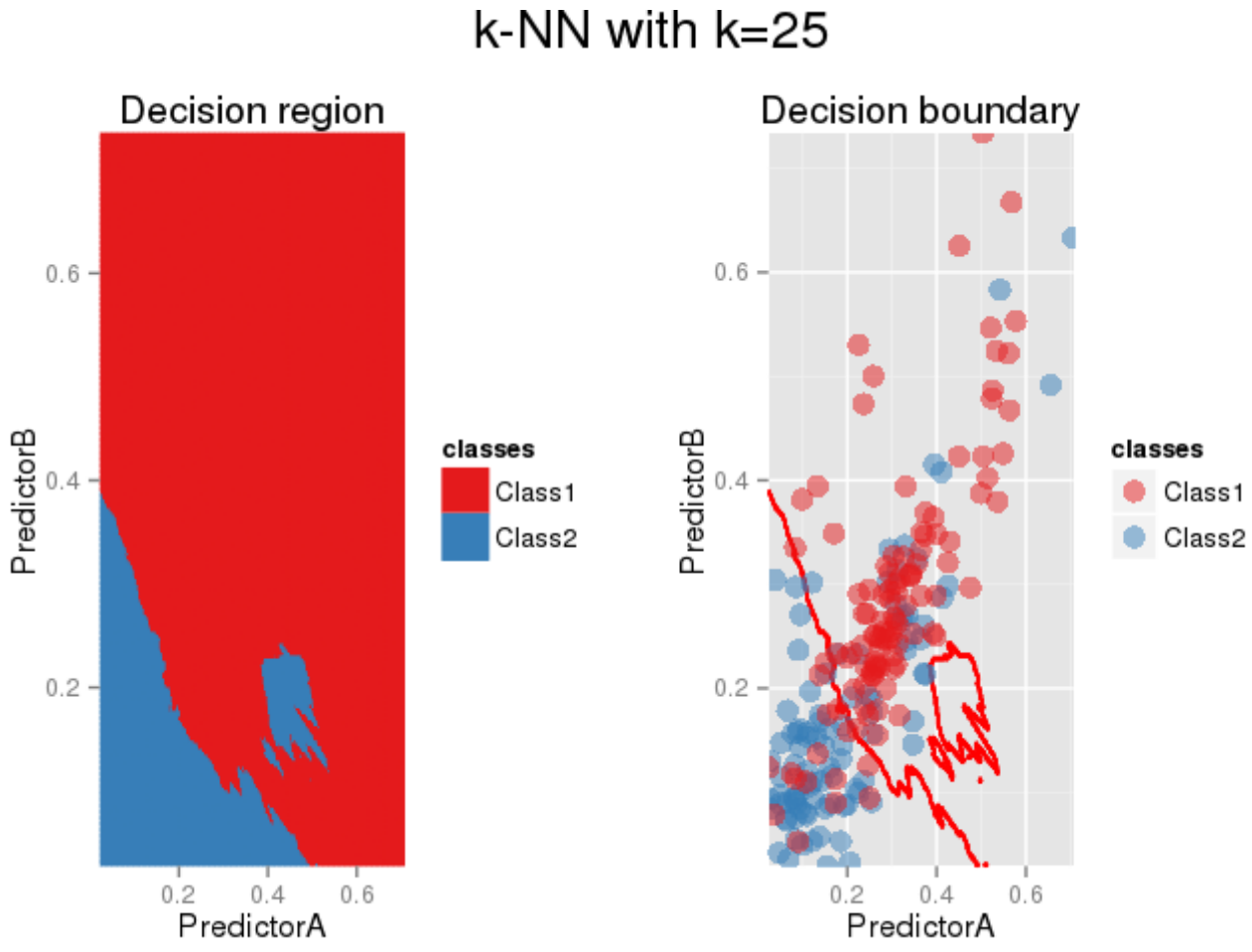
Example: KNN


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

Cross Validation



Example: KNN ($\widehat{k} = 25$ using cross-validation)



Outline

Statistical and Optimization Framework

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

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

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

Problems

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

A Generative Point of View

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

A Dicriminative 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

Empirical Risk Minimization

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

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

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

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

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

Convexification Strategy

Risk Convexification

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

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

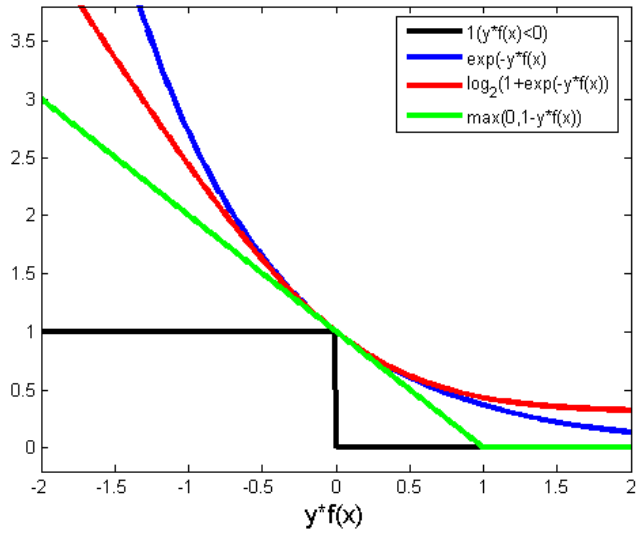
- Use $\widehat{f} = \operatorname{sign}(\tilde{f})$

- Much easier optimization.

Instantiation

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

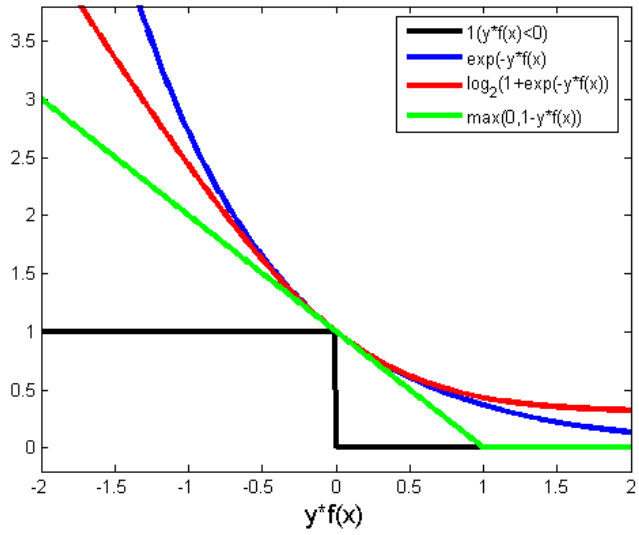
Classification Loss and Convexification



Convexification

- Replace the loss $\ell^{0/1}(Y, f(\mathbf{X}))$ by
$$\ell'(Y, f(\mathbf{X})) = l(Y_i f(\mathbf{X}))$$
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: $\ell'(Y, f(\mathbf{X})) = \log(1 + e^{-Yf(\mathbf{X})})$ (Logistic / NN)
- Hinge loss: $\ell'(Y, f(\mathbf{X})) = (1 - Yf(\mathbf{X}))_+$ (SVM)
- Exponential loss: $\ell'(Y, f(\mathbf{X})) = e^{-Yf(\mathbf{X})}$ (Boosting...)

Properties

The Target is the Bayes Classifier

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

Control of the Excess Risk

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

- Theoretical guarantee!

Proof

- By definition,
$$\mathbb{E}\left[l(Yf)|\mathbf{X}\right] = \eta(\mathbf{X})l(f) + (1-\eta(\mathbf{X}))l(-f)$$
- Let $H(f,\eta) = \eta l(f) + (1-\eta)l(-f)$, the optimal value for $\tilde{f}$ satisfies
$$\delta H(\tilde{f},\eta) = -\eta\delta l(\tilde{f}) + (1-\eta)\delta l(-\tilde{f}) \ni 0.$$
- With a slight abuse of notation, we denote by $\delta l(\tilde{f})$ and $\delta l(-\tilde{f})$ the two subgradients such that
$$\eta\delta l(\tilde{f}) - (1-\eta)\delta l(-\tilde{f}) = 0$$
- Now we discuss the sign of $\tilde{f}$:
  - If $\tilde{f} > 0$, $\delta l(-\tilde{f}) < \delta l(\tilde{f})$ and thus $\eta > (1-\eta)$, i.e. $2\eta - 1 > 0$.
  - Conversely, if $\tilde{f} < 0$ then $2\eta - 1 < 0$
- Thus $\text{sign}(\tilde{f}) = \text{sign}(2\eta - 1)$ i.e. the minimizer of $\mathbb{E}\left[l(yf)|\mathbf{X}\right]$ is $f^*(\mathbf{X}) = \text{sign}(2\eta(\mathbf{X}) - 1)$

Proof

- We define $H(\eta) = \inf_f H(f, \eta) = \inf_f (\eta l(f) + (1-\eta) l(-f))$. By construction, H is a concave function satisfying $H(1/2 + x) = H(1/2 - x)$.
- Furthermore, one verify that if we consider the minimimum over the *wrong sign classifiers*, $inf_{f, f(2\eta-1)<0} H(f, \eta) = l(0)$.
- Indeed,

$$\begin{aligned}
&\inf_{f, f(2\eta-1)<0} H(f, \eta) \\
&\quad = \inf_{f, f(2\eta-1)<0} (\eta l(f) + (1-\eta) l(-f)) \\
&\quad \geq \inf_{f, f(2\eta-1)<0} (\eta (l(0) + l'(0) f) + (1-\eta)(l(0) - l'(0) f)) \\
&\quad \geq l(0) + \inf_{f, f(2\eta-1)<0} l'(0) f (2\eta - 1) = l(0)
\end{aligned}$$

- Furthermore,

$$\mathbb{E}\left[\ell'(Y, f(\mathbf{X})\right] = \mathbb{E}_{\mathbf{X}}\left[H(f, \eta(\mathbf{X})\right]$$
$$\mathbb{E}\left[\ell'(Y, f^\star(\mathbf{X}))\right] = \mathbb{E}_{\mathbf{X}}\left[H(\eta(\mathbf{X})\right]$$

Proof

- We define then
$$\Psi(\theta) = I(0) - H\left(\frac{1+\theta}{2}\right)$$
which is thus a convex function satisfying $\Psi(0) = 0$ and $\Psi(\theta) > 0$ for $\theta > 0$.

Proof

- Recall that

$$\mathbb{E}\left[\ell^{0/1}(Y, \text{sign}(f(\mathbf{X})))\right] - \mathbb{E}\left[\ell^{0/1}(Y, f^\star(\mathbf{X}))\right]$$
$$= \mathbb{E}_{\mathbf{X}}\left[|2\eta(\mathbf{X}) - 1|\mathbf{1}_{f^\star(\mathbf{X}) \neq \text{sign}(f(\mathbf{X}))}\right]$$

- Using Jensen inequality, we derive

$$\Psi\left(\mathbb{E}\left[\ell^{0/1}(Y, \text{sign}(f(\mathbf{X})))\right] - \mathbb{E}\left[\ell^{0/1}(Y, f^\star(\mathbf{X}))\right]\right)$$
$$\leq \mathbb{E}_{\mathbf{X}}\left[\Psi\left(|2\eta(\mathbf{X}) - 1|\mathbf{1}_{f^\star(\mathbf{X}) \neq \text{sign}(f(\mathbf{X}))}\right)\right]$$

- Using $\Psi(0) = 0$ and the symmetry of H,

$$\Psi\left(\mathbb{E}\left[\ell^{0/1}(Y, \text{sign}(f(\mathbf{X})))\right] - \mathbb{E}\left[\ell^{0/1}(Y, f^\star(\mathbf{X}))\right]\right)$$
$$\leq \mathbb{E}_{\mathbf{X}}\left[\left(I(0) - H\left(\left(\frac{1 + |2\eta(\mathbf{X}) - 1|}{2}\right)\right)\right)\mathbf{1}_{f^\star(\mathbf{X}) \neq \text{sign}(f(\mathbf{X}))}\right]$$
$$\leq \mathbb{E}_{\mathbf{X}}\left[\left(I(0) - H(\eta(\mathbf{X}))\right)\mathbf{1}_{f^\star(\mathbf{X}) \neq \text{sign}(f(\mathbf{X}))}\right]$$
$$\leq \mathbb{E}_{\mathbf{X}}\left[\left(I(0) - H(\eta(\mathbf{X}))\right)\mathbf{1}_{f(\mathbf{X})(2\eta(\mathbf{X})-1)<0}\right]$$

Proof

- Using the property of the wrong sign classifiers

$$\begin{aligned}
&\Psi\left(\mathbb{E}\left[\ell^{0/1}(Y, \mathrm{sign}(f(\mathbf{X})))\right] - \mathbb{E}\left[\ell^{0/1}(Y, f^\star(\mathbf{X}))\right]\right) \\
&\quad \leq \mathbb{E}_{\mathbf{X}}\left[\left(H(f, \eta(\mathbf{X})) - H(f^\star, \eta(\mathbf{X}))\right) \mathbf{1}_{f(\mathbf{X})(2\eta(\mathbf{X})-1)<0}\right] \\
&\quad \leq \mathbb{E}_{\mathbf{X}}\left[\left(H(f, \eta(\mathbf{X})) - H(f^\star, \eta(\mathbf{X}))\right)\right] \\
&\quad \leq \mathbb{E}\left[\ell'(Y, f(\mathbf{X})\right] - \mathbb{E}\left[\ell'(Y, f^\star(\mathbf{X}))\right]
\end{aligned}$$

Logistic Revisited

- Ideal solution:

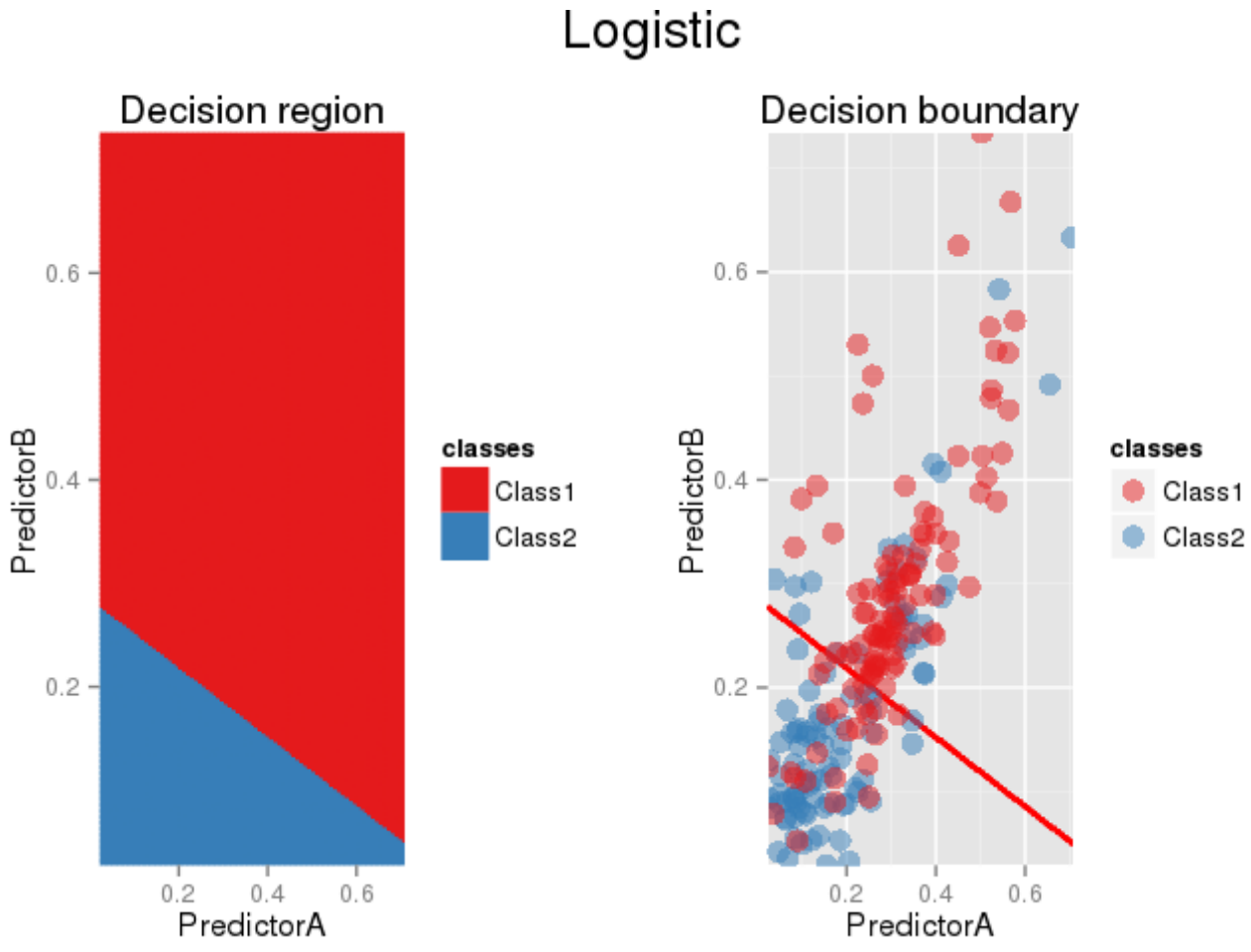
$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\mathbf{X}_i))$$

Logistic regression

- Use $f(\mathbf{X}) = \mathbf{X}^t \beta + b$.
- Use the logistic loss $\ell(Y, f) = \log_2(1 + e^{-Yf})$, i.e. the -log-likelihood.

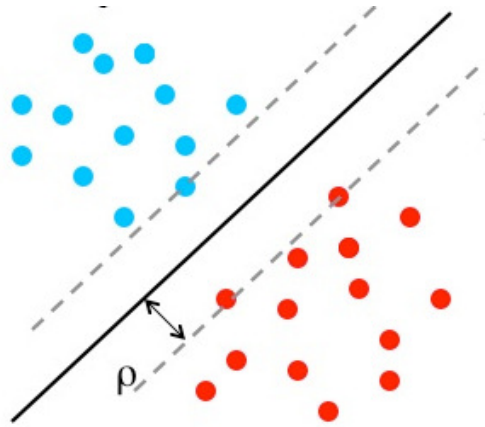
- Different vision than the statistician but same algorithm!

Logistic Revisited



Outline

Ideal Separable Case

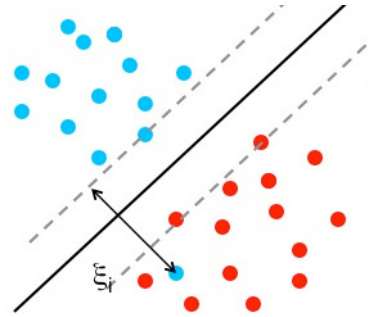


- Linear classifier: $\text{sign}(\mathbf{X}^t\beta + b)$
- Separable case: $\exists(\beta, b), \forall i,\ Y_i(\mathbf{X}_i^t\beta + b) > 0$!

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

- Strict separation: $\exists(\beta, b), \forall i,\ Y_i(\mathbf{X}_i^t\beta + b) \geq 1$
- Maximize the distance between $\mathbf{X}^t\beta + b = 1$ and $\mathbf{X}^t\beta + b = -1$.
- Equivalent to the minimization of $\|\beta\|^2$.

Non Separable Case



- What about the non separable case?
- Relax the assumption that $\forall i, Y_i(\mathbf{X}_i^t\beta + b) \geq 1$.
- Naive attempt:

$$\text{argmin } \|\beta\|^2 + C\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{Y_i(\mathbf{X}_i^t\beta+b)\leq 1}$$

- Non convex minimization.

SVM: better convex relaxation!

$$\text{argmin } \|\beta\|^2 + C\frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\mathbf{X}_i^t\beta + b), 0)$$

SVM as a Penalized Convex Relaxation

- Convex relaxation:

$$\text{argmin } \|\beta\|^2 + C\frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\mathbf{X}_i^t\beta + b), 0)$$

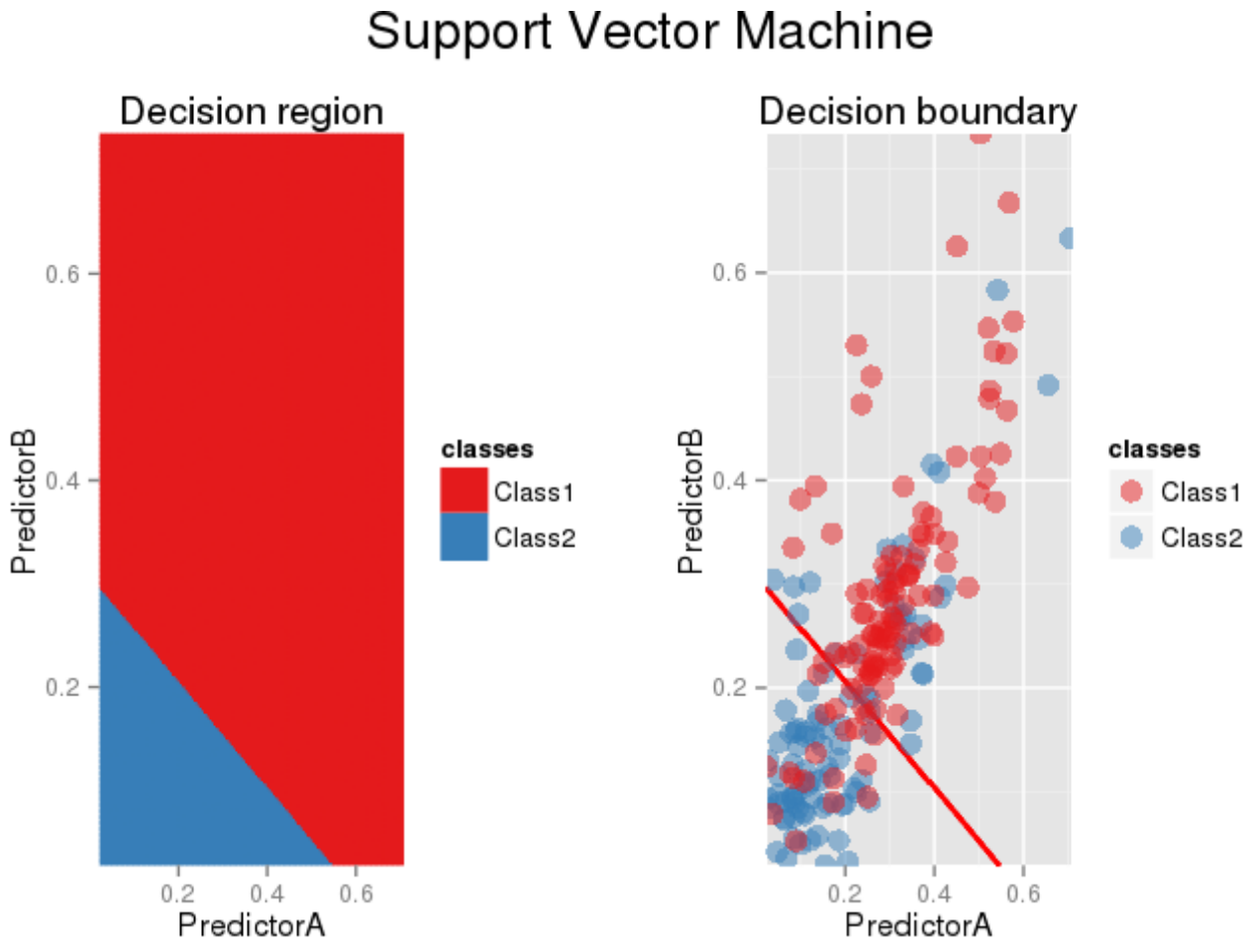
$$= \text{argmin } \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\mathbf{X}_i^t\beta + b), 0) + \frac{1}{C}\|\beta\|^2$$

- **Prop:** $\ell^{0/1}(Y_i, \text{sign}(\mathbf{X}_i^t + b)) \leq \max(1 - Y_i(\mathbf{X}_i^t\beta + b), 0)$

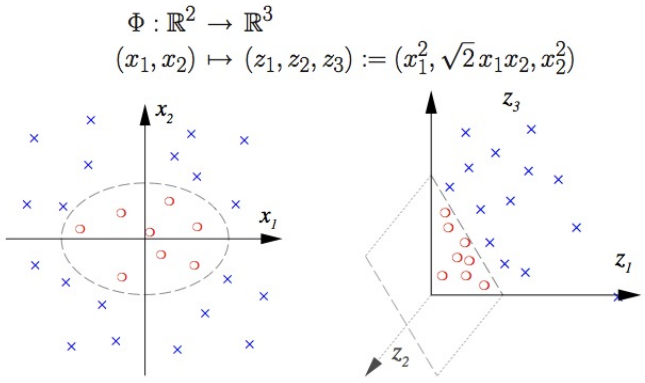
Penalized convex relaxation (Tikhonov!)

$$\frac{1}{n}\sum_{i=1}^{n}\ell^{0/1}(Y_i, \text{sign}(\mathbf{X}_i^t\beta + b))$$

$$\leq \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\mathbf{X}_i^t\beta + b), 0) + \frac{1}{C}\|\beta\|^2$$


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

The Kernel Trick

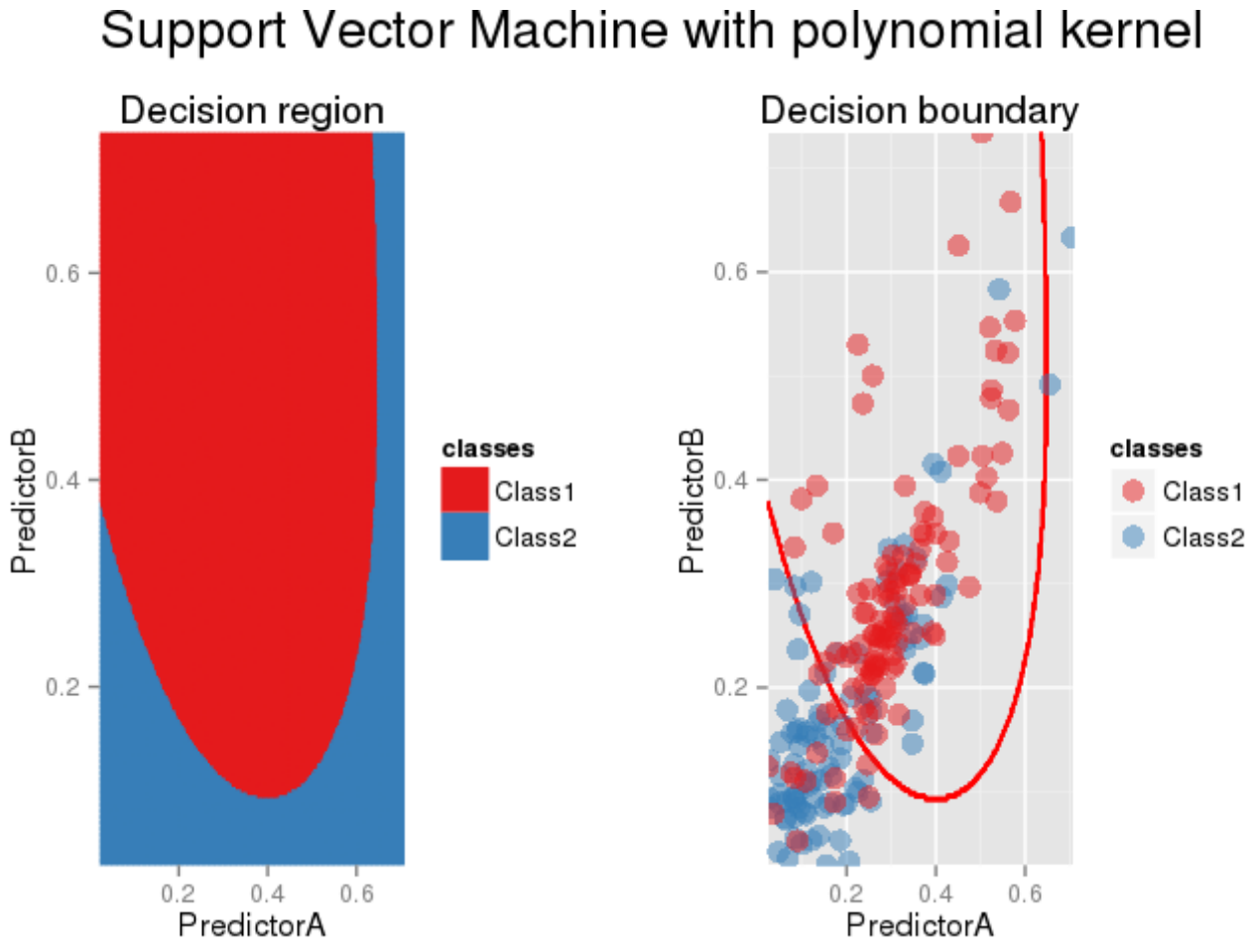


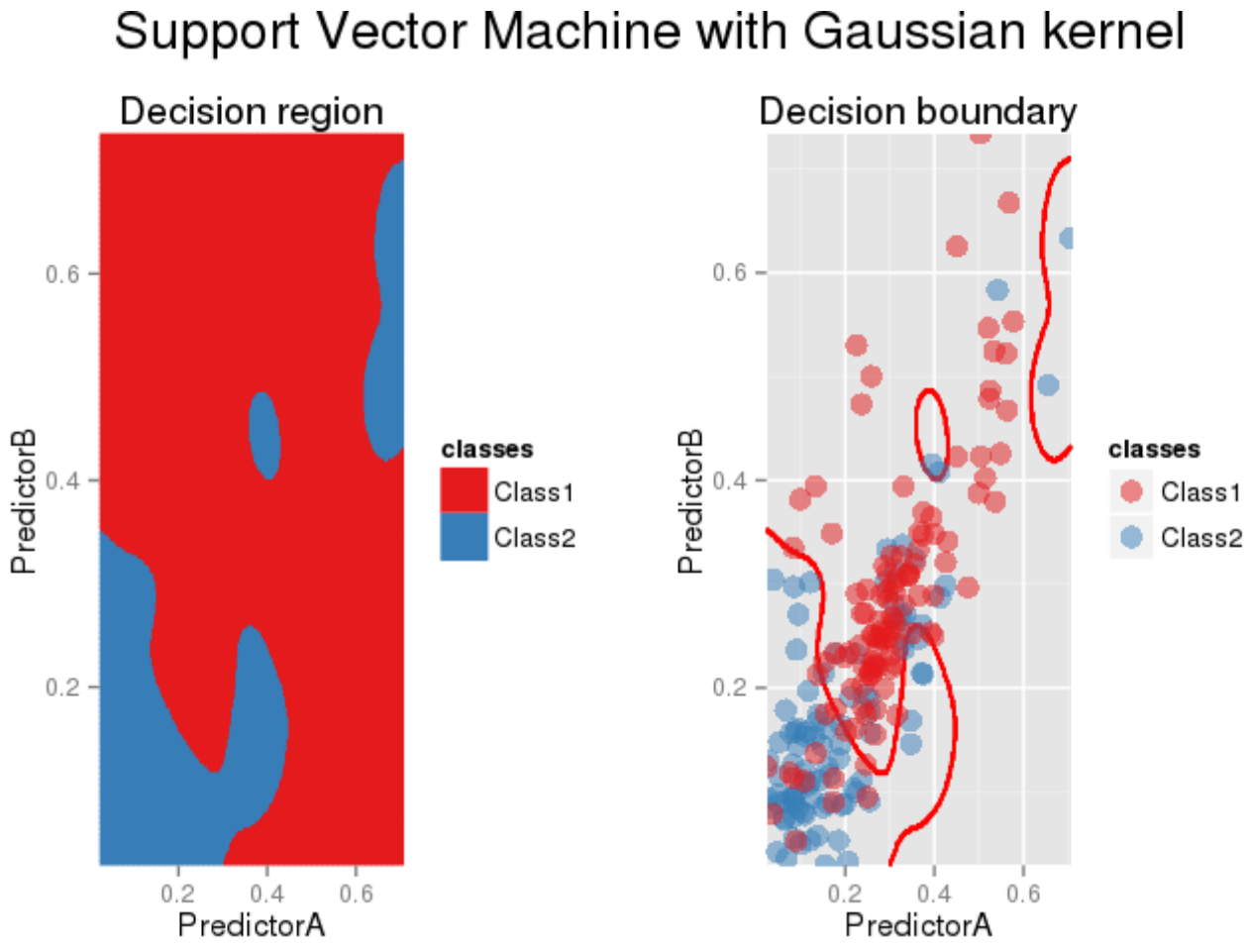
- Non linear separation: just replace x by a non linear $\Phi(x)$...

Kernel trick

- Computing $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ may be easier than computing $\Phi(x)$, $\Phi(x')$ and then the scalar product!
- Φ can be specified through its definite positive kernel k.

- Examples: Polynomial kernel $k(x, x') = (1 + \langle x, x' \rangle)^d$, Gaussian kernel $k(x, x') = e^{-\|x-x'\|^2/2}$,...
- RKHS setting!
- Can be used in (logistic) regression and more...


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

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.
- **Penalization approach:** use empirical loss criterion but penalize it by 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 penalized risk.

Which loss to use?

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

Penalization

Penalization as a risk correction

- The empirical loss computed on an estimator selected in a family according to the data is biased!
- Optimistic estimation of the risk...
- Estimate an upper bound of this optimism for a given family, called the penalty.
- Add it to the empirical loss

- One can also think of the penalty as a way to force the use of *simple* models...
- **Rk:** Interpretability with both the statistical and the optimization point of view.

Penalization

Penalized Loss

- Minimization of

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

where $\text{pen}(\theta)$ is a penalty.

Penalties

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

Instantiation

- Penalized Loss for Linear Model
- Structural Risk Minimization

Variable Selection

- **Setting**: Gen. linear model = prediction of Y by $h(\mathbf{X}^t\beta)$.

Model coefficients

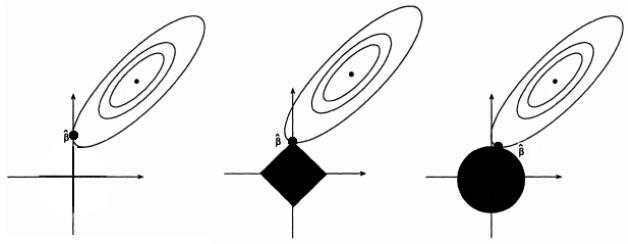
- Model entirely specified by β.
- Coefficientwise:
  - $\beta_i = 0$ means that the ith covariate is not used.
  - $\beta_i \sim 0$ means that the ith covariate as a *low* influence...

- 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.

Constraint and Penalization

Constrained Optimization

- Choose a constant C.
- Compute β as

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

Lagrangian Reformulation

- Choose λ and compute β as

$$\underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, h(\mathbf{X}_i^t \beta)) + \lambda \|\beta\|_p^{p'}$$

  with $p' = p$ except if $p = 0$ where $p' = 1$.

- Easier calibration...

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

Penalization

Penalized Linear Model

- Minimization of

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

  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!**

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 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 parameters with a penalty $\widehat{\lambda}\widetilde{\text{pen}}$.

Why not using only CV?

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

Penalization

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

- Minimization of

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

  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

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

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

- No 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 $\widehat{\lambda}\widetilde{\text{pen}}$.

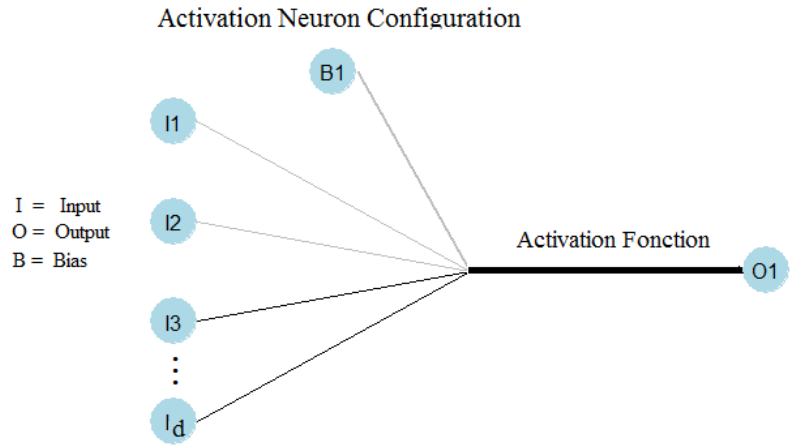
Why not using only CV?

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

Outline

Artificial Neuron and Logistic Regression

A Discriminative PoV



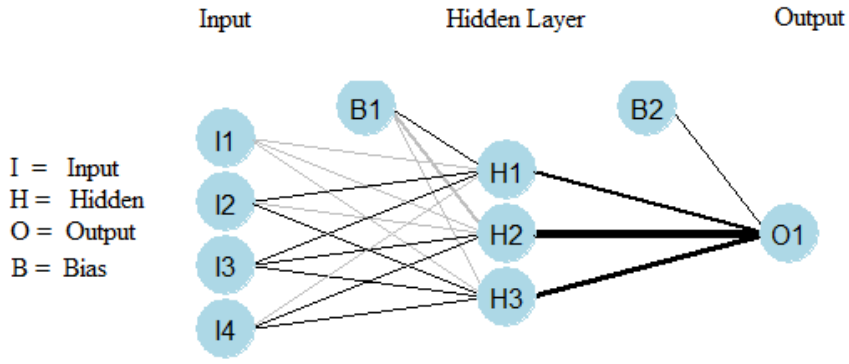
Artificial neuron

- 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 the 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.

Neural network

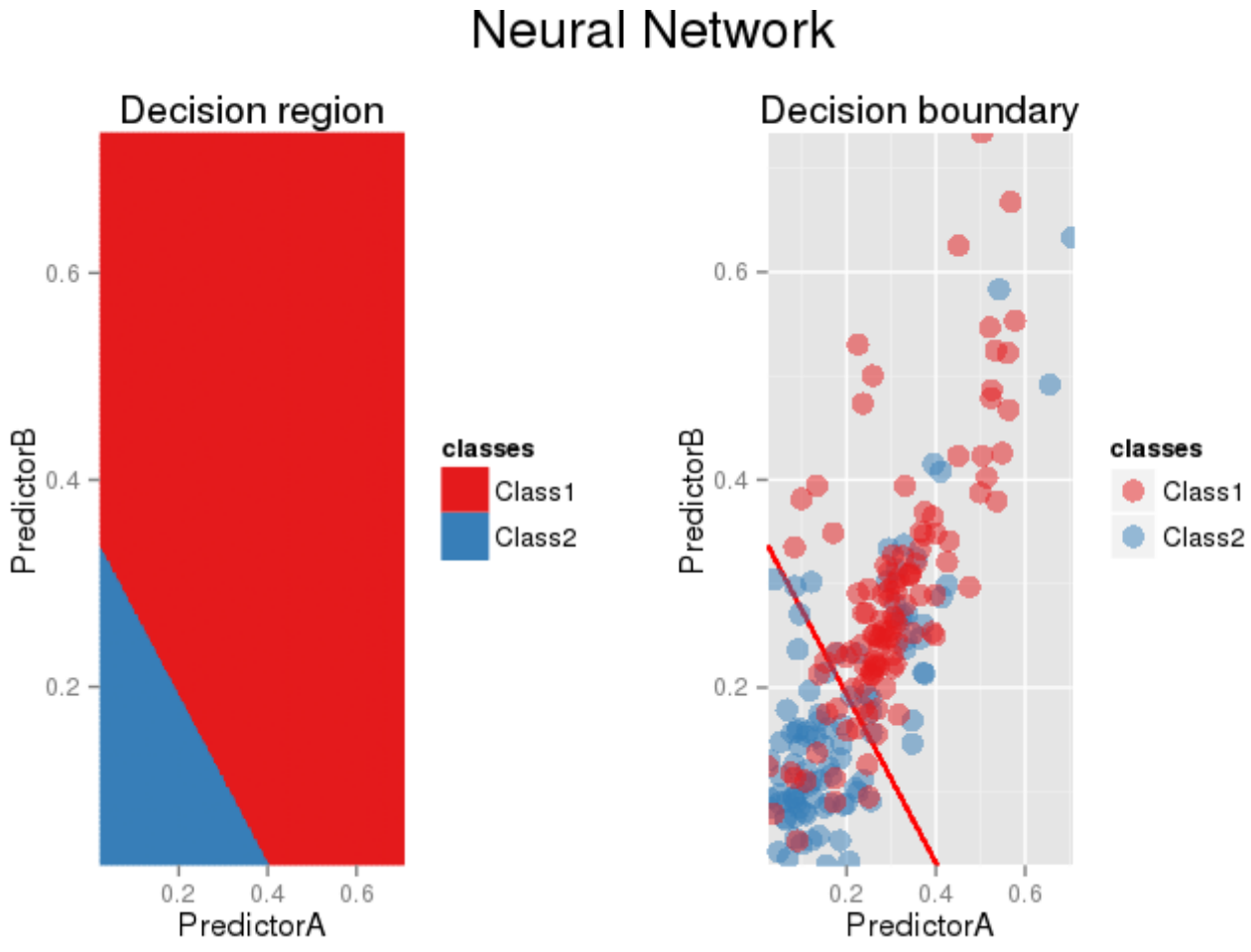


Neural network structure

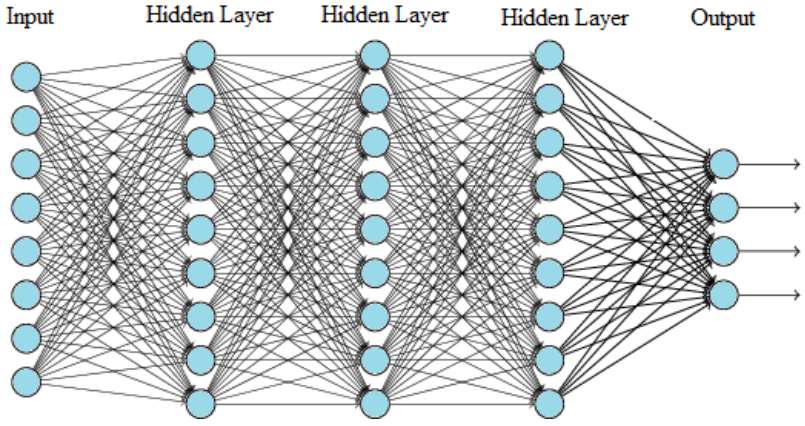
- Cascade of artificial neurons organized in layers
- Thresholding decision only at the output layer

- Most classical case use logistic neurons and the -log-likelihood as the criterion to minimize.
- Classical (stochastic) gradient descent algorithm (Back propagation)
- Non convex and thus may be trapped in local minima.

Neural network



Deep Neural Network

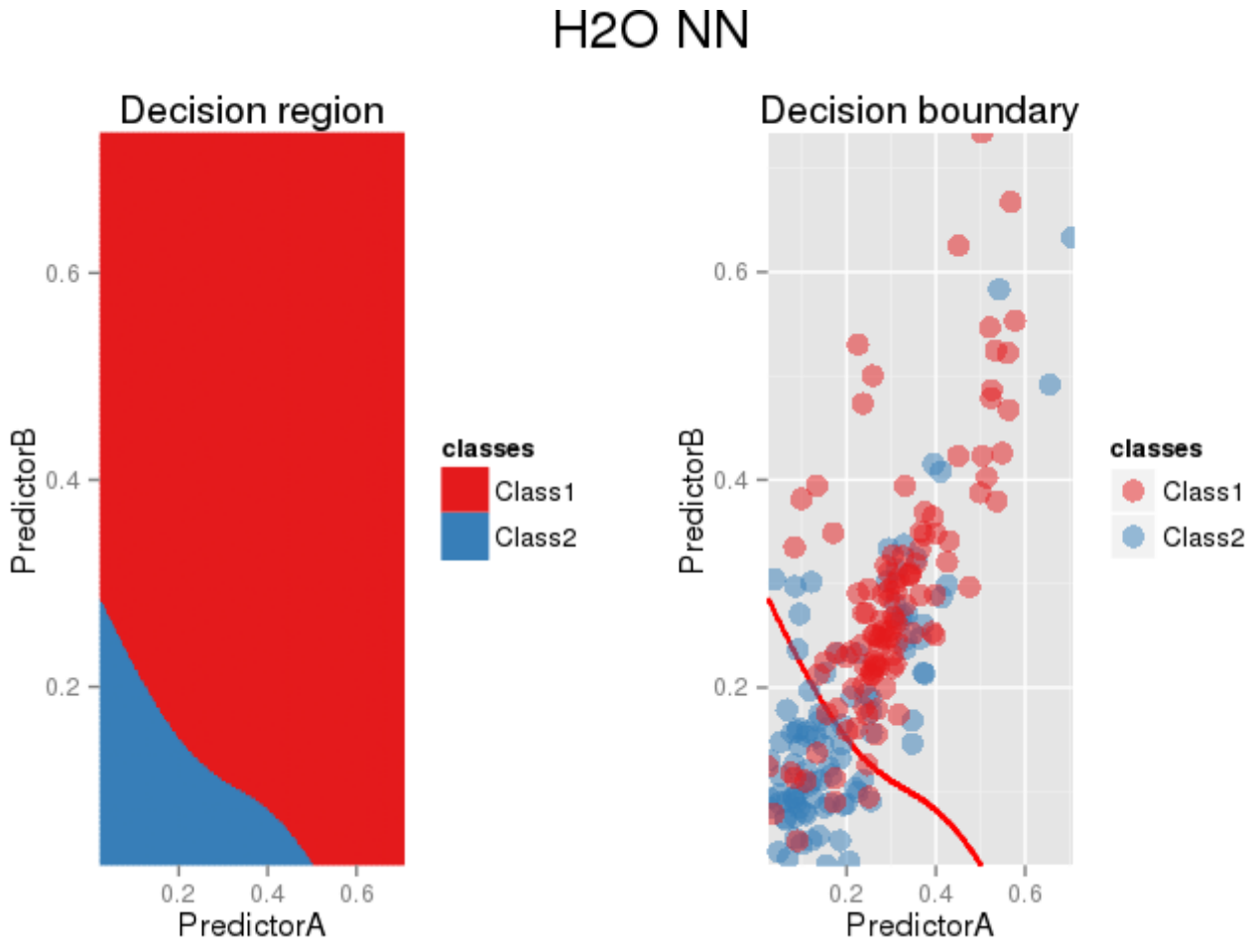


Deep Neural Network structure

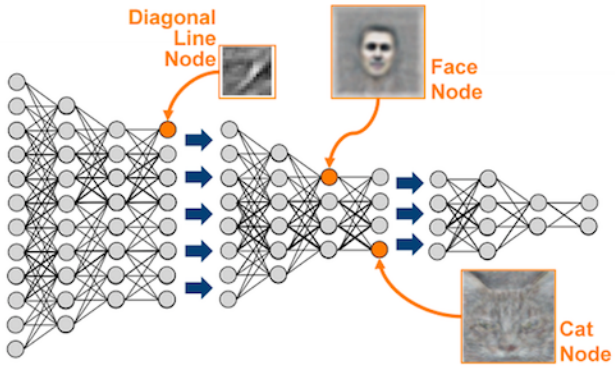
- Deep cascade of layers!

- No conceptual novelty but initialization becomes a crucial issue.
- Bunch of solutions proposed on a greedy initialization of the layers starting from the deepest one.
- Very impressive results!

Deep Neural Network



Deep Learning



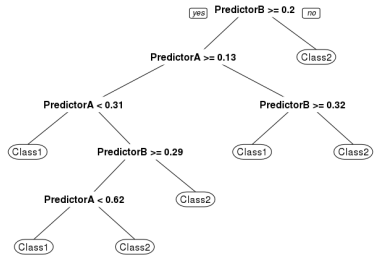
Family of Machine Learning algorithm combining:

- a (deep) multilayered structure,
- a clever (often unsupervised) initalization,
- a more classical final fine tuning optimization.

- Examples: Deep Neural Network, Deep (Restricted) Boltzman Machine, Stacked Encoder...
- Appears to be very efficient but lack of theoretical fundation!

Outline

Regression Trees

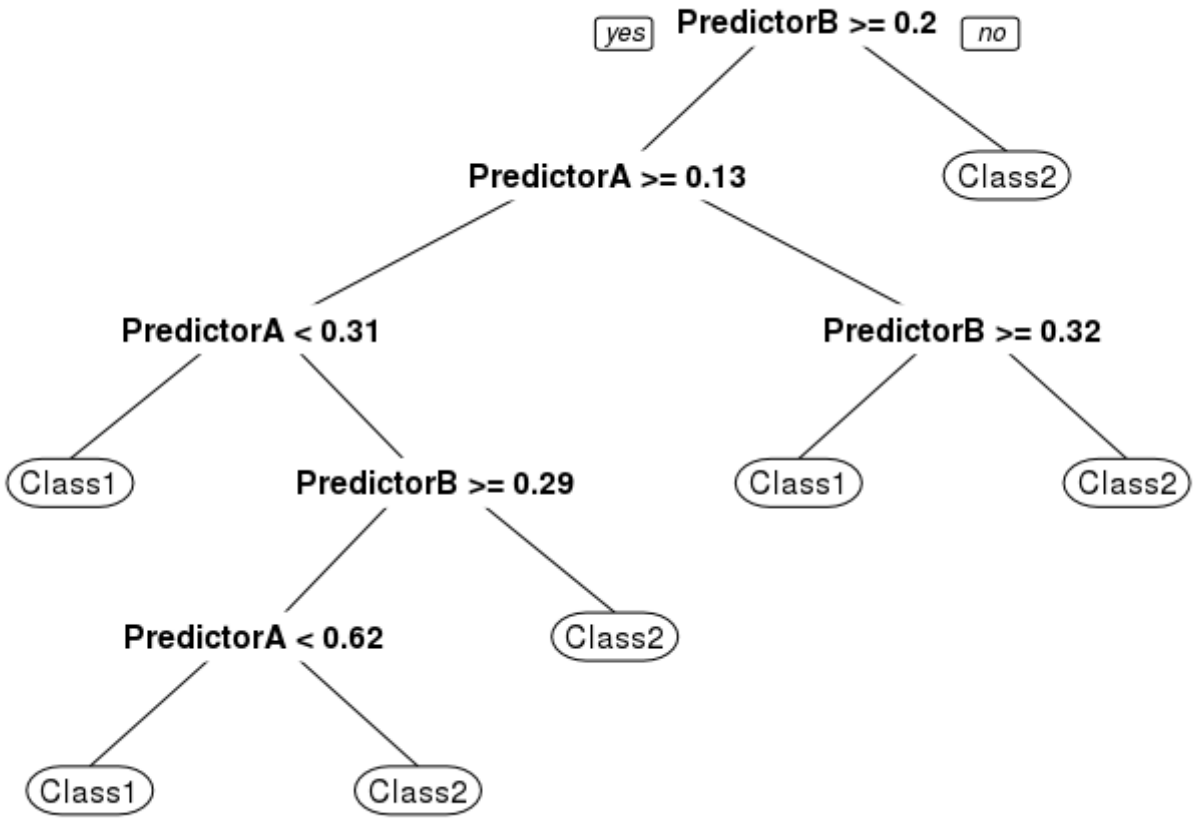


Tree principle

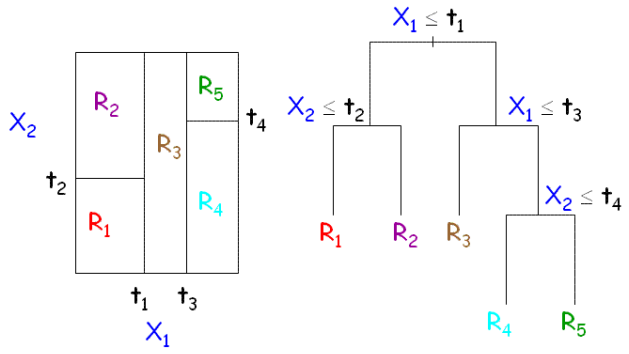
- Construction of a recursive partition through a tree structured set of questions (splits around a given value of a variable)
- For a given partition, statistical approach **and** optimization approach yields the same classifier!
- A simple majority vote in each leaf

- Quality of the prediction depends on the tree (the partition).
- 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

Various definition of *homogeneous*

- CART: empirical loss based criterion
$$C(R, \overline{R}) = \sum_{x_i \in R} \ell(Y_i, y(R)) + \sum_{x_i \in \overline{R}} \ell(Y_i, y(\overline{R}))$$
- CART: Gini index (classification)
$$C(R, \overline{R}) = \sum_{x_i \in R} p(R)(1 - p(R)) + \sum_{x_i \in \overline{R}} p(\overline{R})(1 - p(\overline{R}))$$
- C4.5: entropy based criterion (Information Theory)
$$C(R, \overline{R}) = \sum_{x_i \in R} H(R) + \sum_{x_i \in \overline{R}} H(\overline{R})$$

- 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 (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!

Pruning

- Model select. 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:

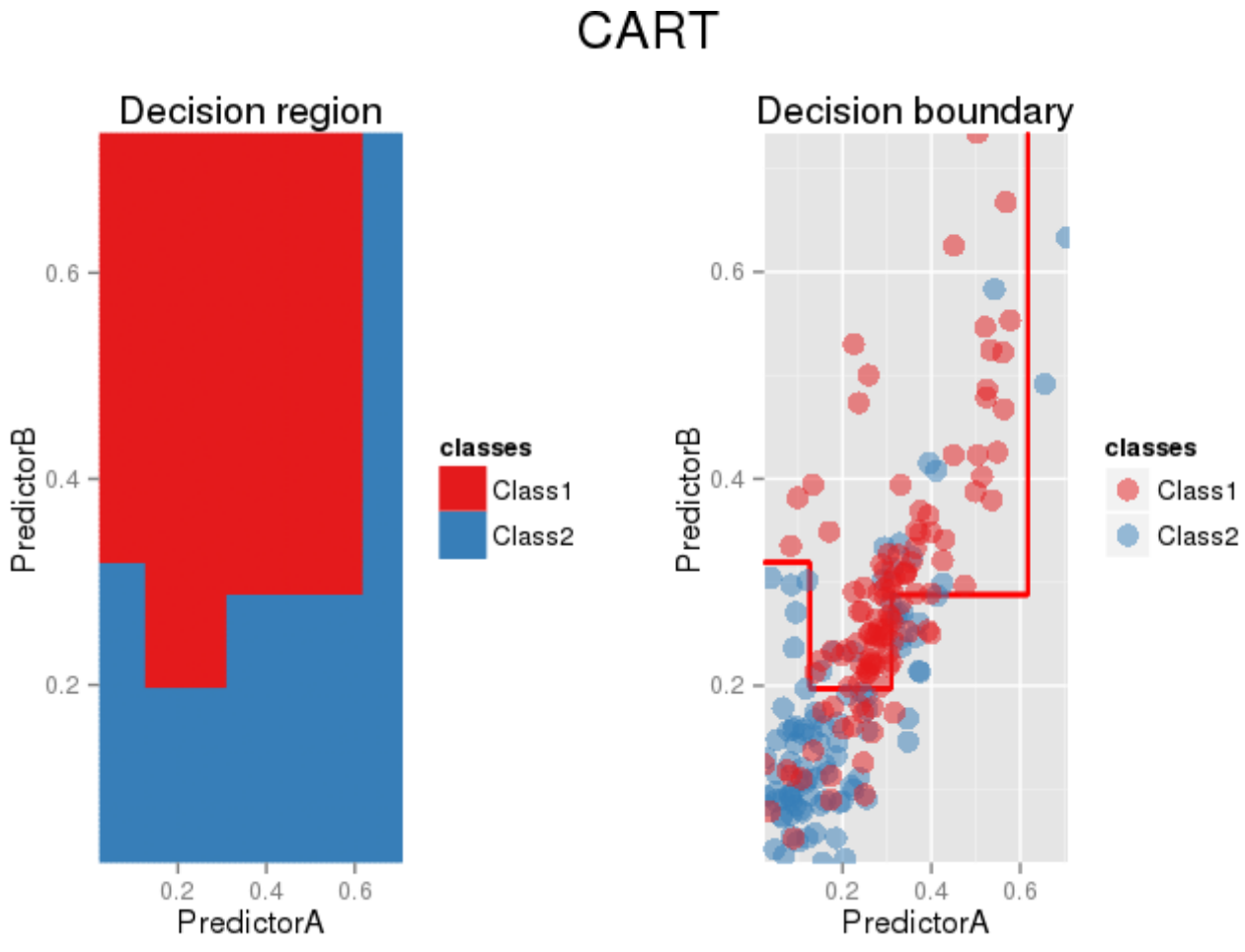
$$\sum_{i=1}^{n} \ell'(Y_i, f_{\mathcal{L}(\mathbf{x}_i)}(\mathbf{X}_i) + \lambda|\mathcal{T}| = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\mathbf{X}_i\in\mathcal{L}} \ell'(Y_i, f_{\mathcal{L}}(\mathbf{X}_i) + \lambda \right)$$

- Simple cross-Validation (with $(\mathbf{X}'_i, \mathbf{X}'_i)$ a different dataset):

$$\sum_{i=1}^{n'} \ell'(Y'_i, f_{\mathcal{L}}(\mathbf{X}'_i) = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\mathbf{X}'_i\in\mathcal{L}} \ell'(Y'_i, f_{\mathcal{L}}(\mathbf{X}'_i) \right)$$

- Limits over-fitting...

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

Ensemble methods

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

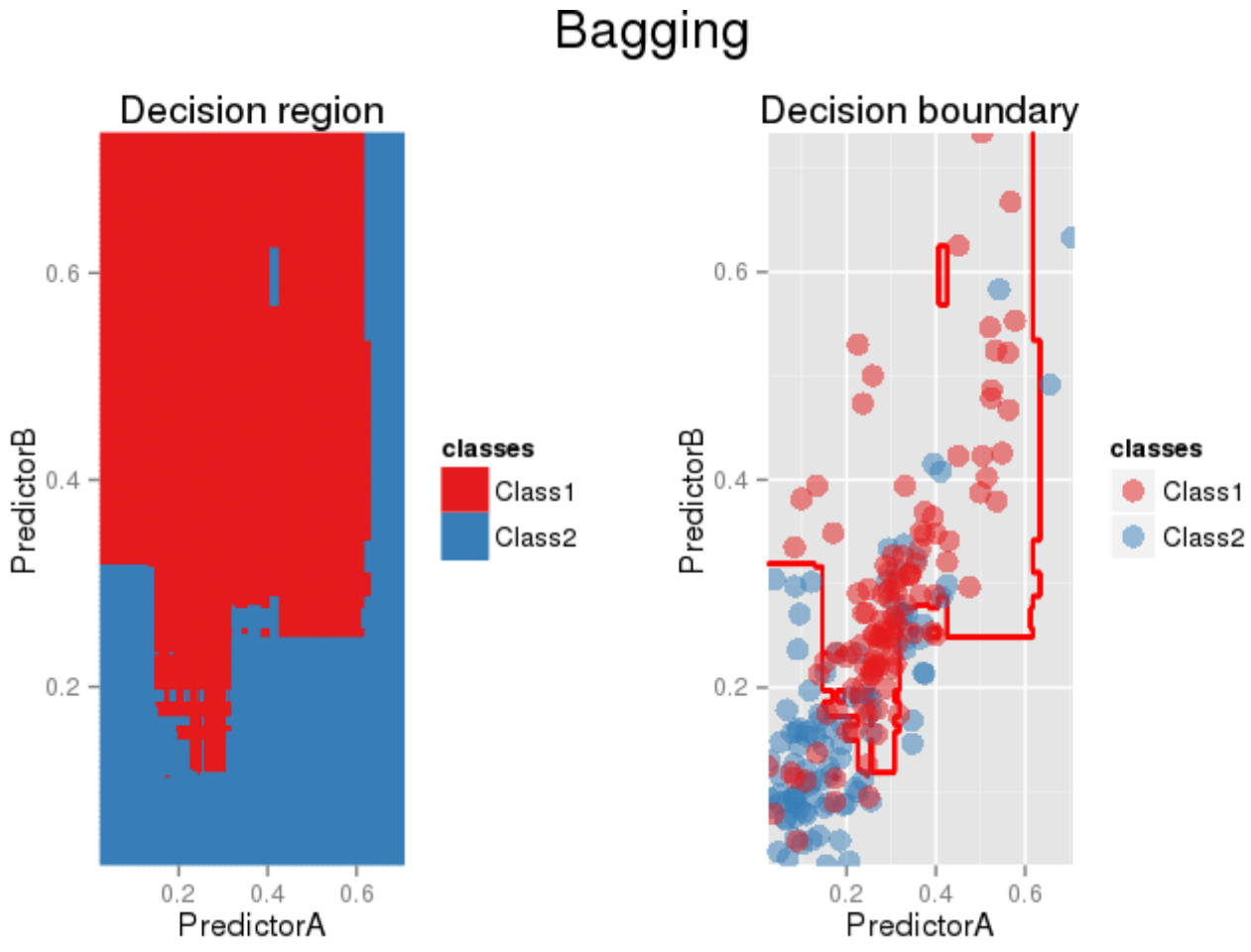
Parallel construction

- Construct several trees from bootstrapped samples and average the responses (bagging)
- Add more randomness in the tree construction (random forests)

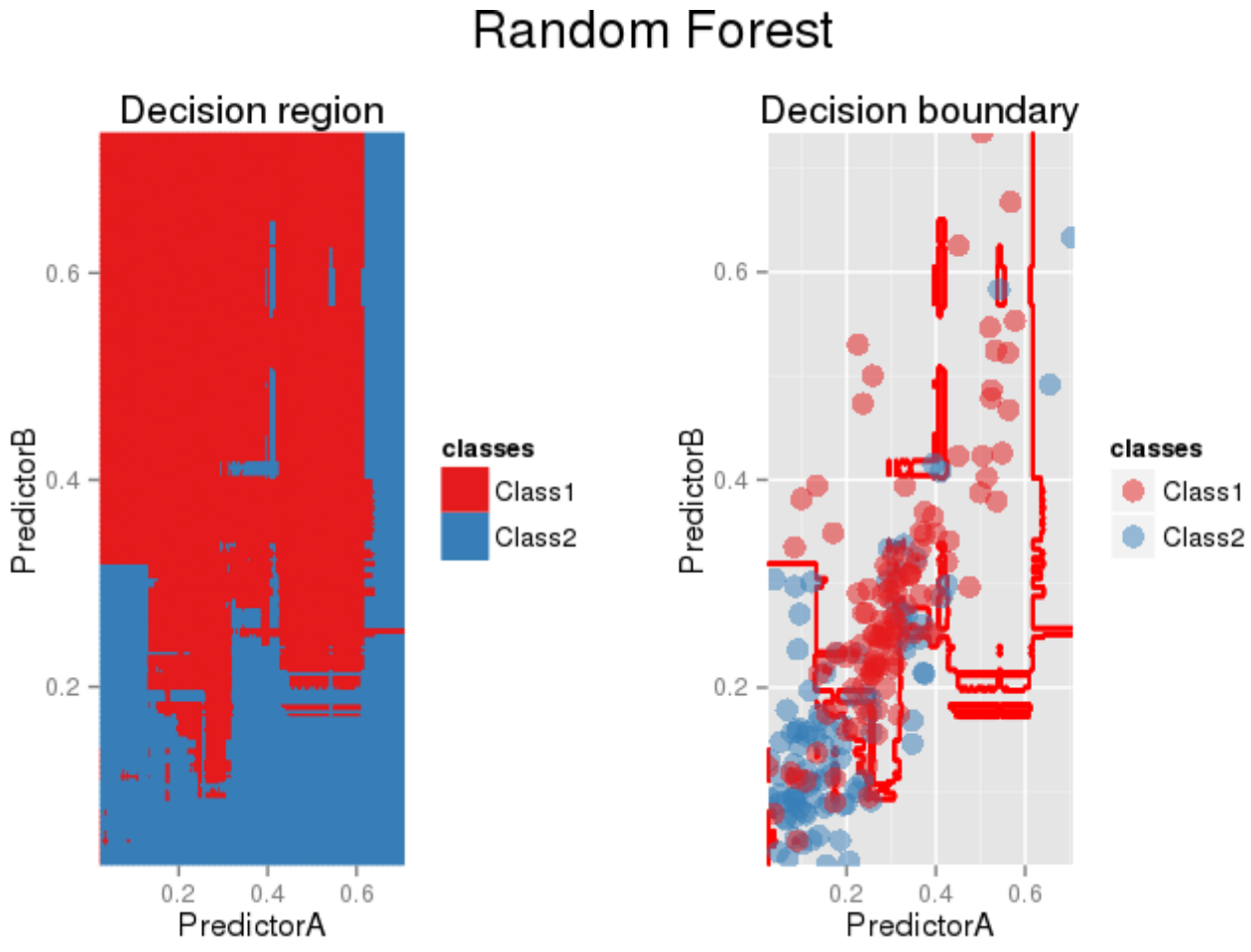
Sequential construction

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

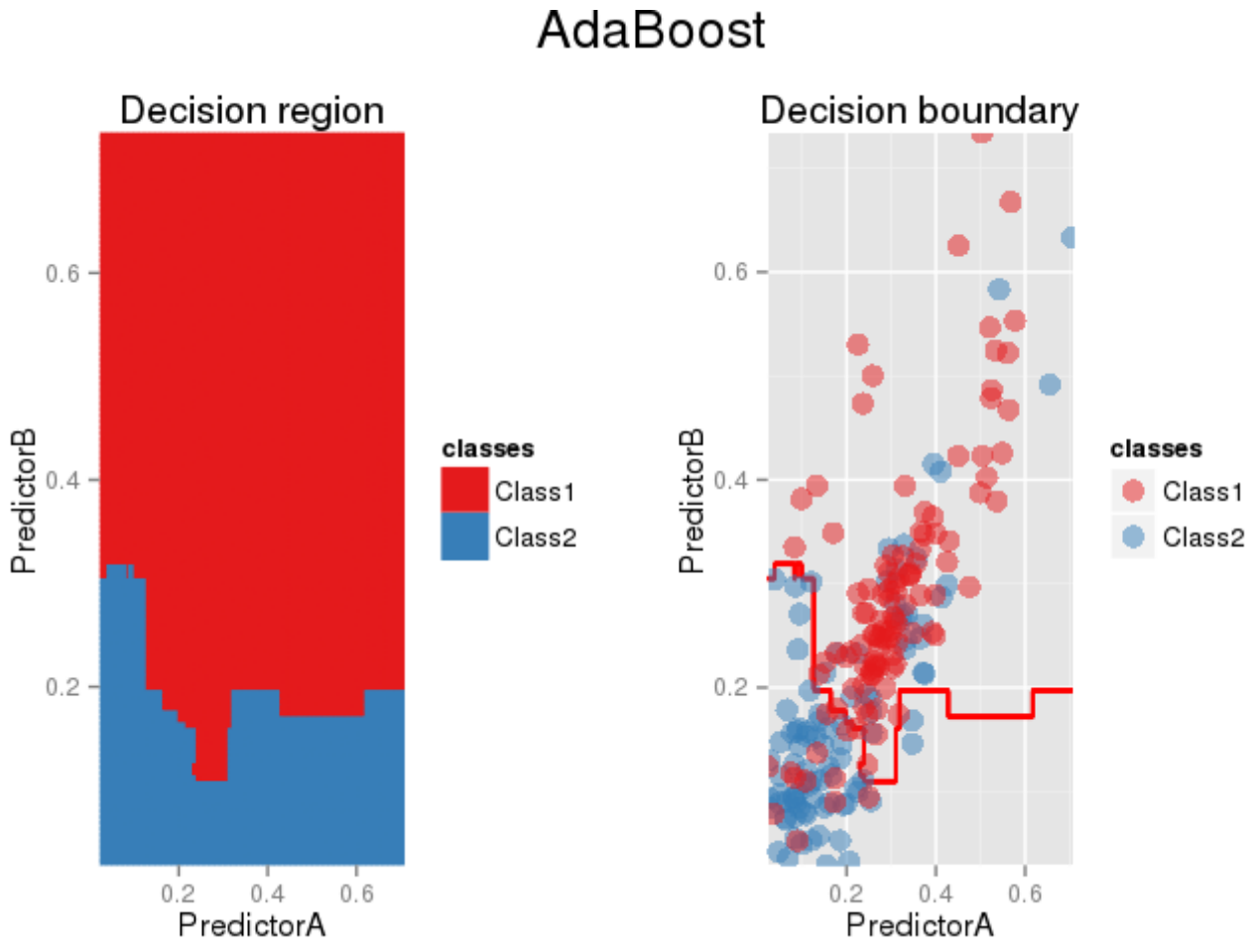
Ensemble methods



Ensemble methods



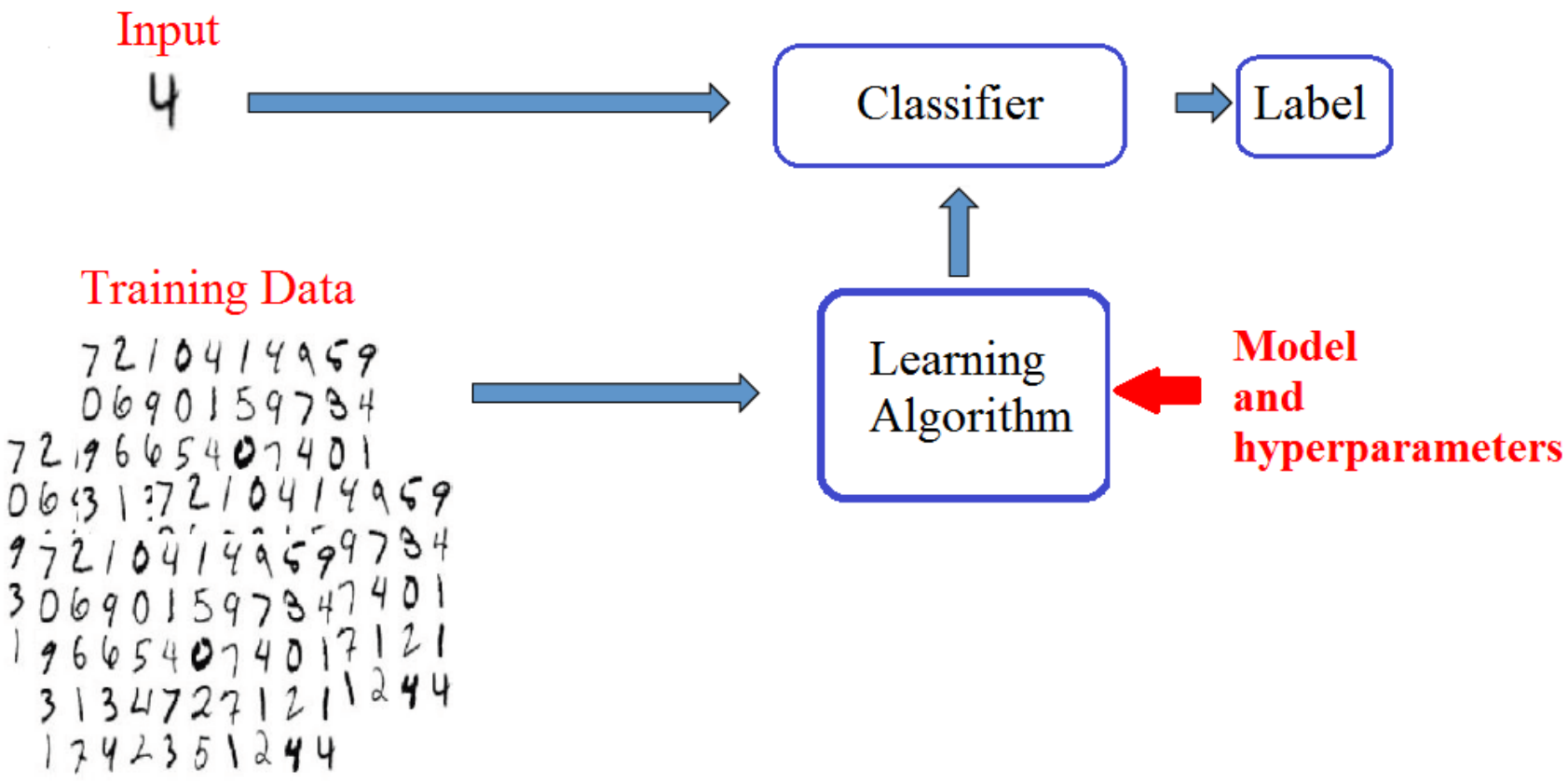
Ensemble methods



Outline

Outline

Model and Hyperparameters



Logistic Regression

- Ideal solution:

$$f^*(\mathbf{X}) = \text{argmax}\, \mathbb{P}\{Y|\mathbf{X}\}$$

Logistic

- Model $Y|\mathbf{X}$ 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:

$$f^*(\mathbf{X}) = \text{argmax}\, \mathbb{P}\{Y|\mathbf{X}\}$$

Generative Modeling

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

- Model hyperparameters:
  - Features
  - Generative model

Kernel Method

- Ideal solution:

$$f^*(\mathbf{X}) = \text{argmax}\, \mathbb{P}\{Y|\mathbf{X}\}$$

Kernel methods

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

- Model hyperparameters:
  - Features
  - Bandwidth and kernel

Logistic Regression

- Ideal solution:

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

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:

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

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:

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

NN

- Neuron: $\mathbf{X} \mapsto \sigma(\mathbf{X}^t \beta + b)$
- 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:
  $f^*(\mathbf{X}) = \text{argmax}\, \mathbb{P}\{Y|\mathbf{X}\}$ and $f^* = \underset{f \in \mathcal{S}}{\text{argmin}}\, \mathbb{E}\left[\ell^{0/1}(Y, f(\mathbf{X}))\right]$

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

$$f_\beta(\mathbf{X}) = \Phi(\mathbf{X})^t \beta$$

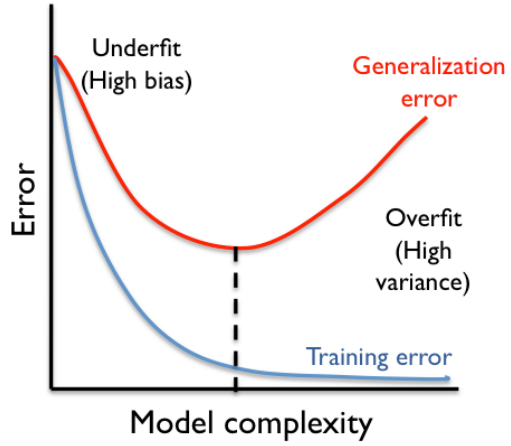
Feature Design

- 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 an other criterion than the training error!

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.
- **Penalization approach:** use empirical loss criterion but penalize it by 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 penalized risk.

Which loss to use?

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

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 parameters with a penalty $\widehat{\lambda}\widetilde{\text{pen}}$.

Why not using only CV?

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

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...