MSc Big Data for Business - *MAP 534*
Introduction to supervised learning

Supervised classification

Linear/Quadratic discriminant analysis (LDA/QDA) & Support Vector Machines (SVM)

Outline

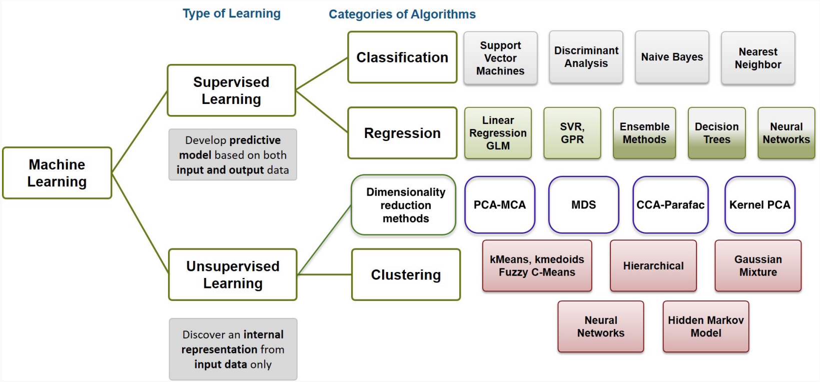
Unsupervised learning

- Dimensionality reduction methods (PCA).
- Clustering (k-means, Hierarchical clustering).
- Handling missing values/ matrix completion.
  $\Rightarrow$ Data visualization - exploratory data analysis.

Supervised learning

- Theoritical framework - Bayes risk.
- Logistic regression.
- Optimization.
- Nonparametric methods.

Machine Learning



Unsupervised and Supervised

Unsupervised Learning

- Goal: Discover a structure within a set of individuals $(\mathbf{X}_i)$.
- Data: Learning set $(\mathbf{X}_i)$

Supervised Learning:

- Goal: Learn a function f predicting a variable Y from an individual $\mathbf{X}$.
- Data: Learning set $(\mathbf{X}_i, Y_i)$

Second case is better posed.

Outline

Plan

Machine Learning everywhere

Use data to extract a prediction function

Search engine, text-mining.

Diagnosis, Fault detection.

Business analytics.

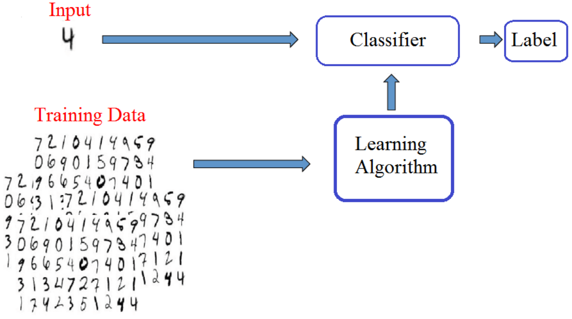
Social network analytics.

Data-mining.

Link prediction.

Robotics.

A definition of Machine Learning

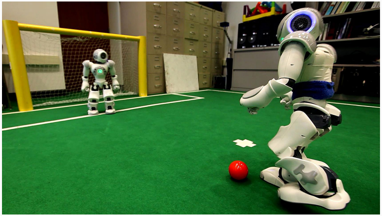


A definition by Tom Mitchell (`http://www.cs.cmu.edu/~tom/`)

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:

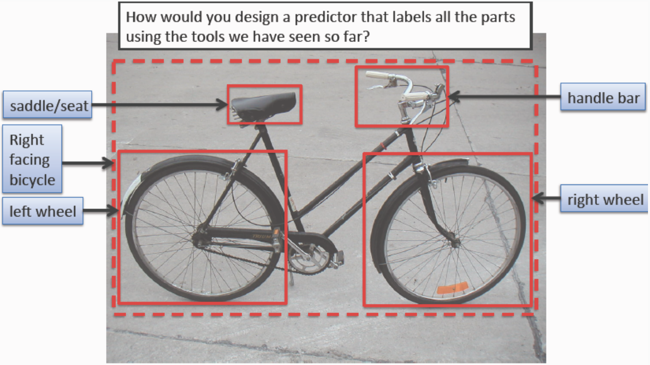


Task: play football.

Performance: score.

Experience: current environment and outcome, past games...

Object recognition in an image

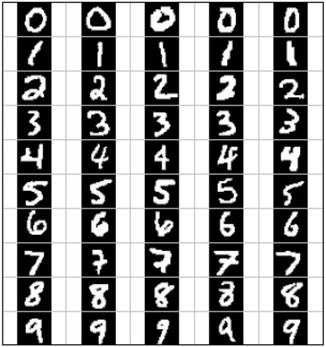


Task: say if an object is present or not in the image.

Performance: number of errors.

Experience: set of previously seen labeled image.

Number



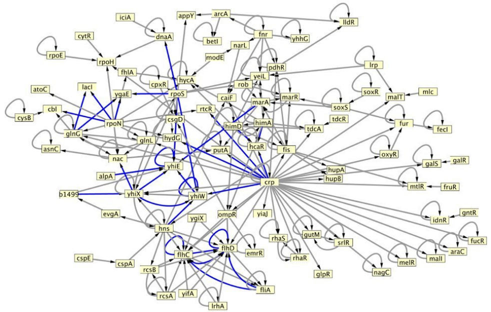
Task: Read a ZIP code from an envelop.

Goal: give a number from an image.

$\mathbf{X}$ = image.

Y = corresponding number.

Applications in biology



Task: protein interaction network prediction.

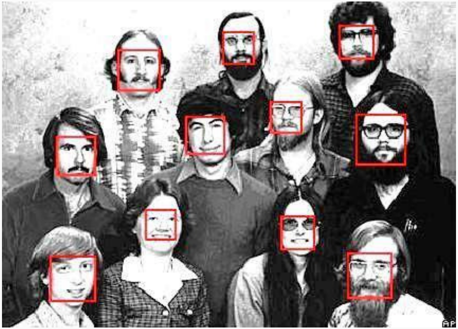
Goal: predict (unknown) interactions between proteins.

$\mathbf{X}$ = pair of proteins.

Y = existence or no of interaction.

Numerous similar questions in bio(informatics), genomic, etc.

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...

Supervised learning methods

Support Vector Machine

Linear Discriminant Analysis

Logistic Regression

Trees/ Random Forests

Kernel methods

Neural Networks

Many more...

Plan

Supervised Learning Framework

Input measurement $\mathbf{X} \in \mathcal{X}$ (often $\mathcal{X} \subset \mathbb{R}^d$).

Output measurement $Y \in \mathcal{Y}$.

The joint distribution of $(\mathbf{X}, Y)$ is p with p unknown.

Training data : $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ (i.i.d. $\sim p$).

$Y \in \{-1, 1\}$ (classification) or $Y \in \mathbb{R}$ (regression).

A predictor is a measurable function in $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$.

Goal

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

Need to specify the meaning of good.

Loss and Probabilistic Framework

Loss function for a generic predictor

- Loss function: $\ell(Y, f(\mathbf{X}))$ measures the goodness of the prediction of Y by $f(\mathbf{X})$.
- 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 function

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

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

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

Best Solution

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

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

Bayes Predictor (explicit solution)

In binary classification with $0 - 1$ loss:

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

In regression with the quadratic loss

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

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

Plugin Classifier

Estimate $\eta(\mathbf{x}) = \mathbb{P}[Y = 1 | \mathbf{X} = \mathbf{x}]$ by $\widehat{\eta}_n(\mathbf{x})$ using the training dataset and plug it the Bayes classifier.

Plugin Bayes Classifier

In binary classification with $0 - 1$ loss:

$$\widehat{f}_n(\mathbf{X}) = \begin{cases} +1 & \text{if } \widehat{\eta}_n(\mathbf{x}) \geq 1/2\,, \\ -1 & \text{otherwise}\,. \end{cases}$$

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 $\widehat{f}_n : \mathbb{R}^d \to \{-1, 1\}$

$$\widehat{f}_n(\mathbf{X}) = \begin{cases} +1 & \text{if } \widehat{\eta}_n(\mathbf{X}) \geq 1/2\,, \\ -1 & \text{otherwise}\,. \end{cases}$$

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

Classification Risk Analysis

The missclassification error satisfies:

$$0 \leq \mathbb{P}(\widehat{f}_n(\mathbf{X}) \neq Y) - L^\star \leq 2\mathbb{E}\left[|\eta(\mathbf{X}) - \hat{\eta}_n(\mathbf{X})|^2\right]^{1/2},$$

where

$$L^\star = \mathbb{P}[f^\star(\mathbf{X}) \neq Y]$$

and $\widehat{\eta}_n(\mathbf{x})$ is an empirical estimate based on the training dataset of

$$\eta(\mathbf{x}) = \mathbb{P}[Y = 1|\mathbf{X} = \mathbf{x}].$$

Instantiation: how to estimate $Y|\mathbf{X}$?

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 $\mathcal{L}_\theta(\mathbf{X})$: *linear regression, logistic regression...*

Nonparametric conditional modeling.

Estimate the law of $Y|\mathbf{X}$ by a **non parametric** estimate: *kernel methods, nearest neighbors...*

Plan

Plan

Fully Generative Modeling

If the law of $(\mathbf{X}, Y)$ **is known** everything can be easy!

Bayes formula

With a slight abuse of notation,

$$\mathbb{P}\left\{Y|\mathbf{X}\right\} = \frac{\mathbb{P}\left\{(\mathbf{X}, Y)\right\}}{\mathbb{P}\left\{\mathbf{X}\right\}} = \frac{\mathbb{P}\left\{\mathbf{X}|Y\right\}\mathbb{P}\left\{Y\right\}}{\mathbb{P}\left\{\mathbf{X}\right\}}$$

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*.

Remark: 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

Simplest setting in classification!

Bayes formula

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

Binary Bayes classifier (the best solution).

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

Heuristic: estimate those quantities and plug the estimations. Choosing different models/estimators for $\mathbb{P}\left\{\mathbf{X} | Y\right\}$ leads to different classifiers.

Remark: no need to renormalize by $\mathbb{P}\left\{\mathbf{X}\right\}$ to take the decision!

Naive Bayes

Classical algorithm using a crude modeling for $\mathbb{P}\left\{\mathbf{X}|Y\right\}$:

- Feature independence assumption:

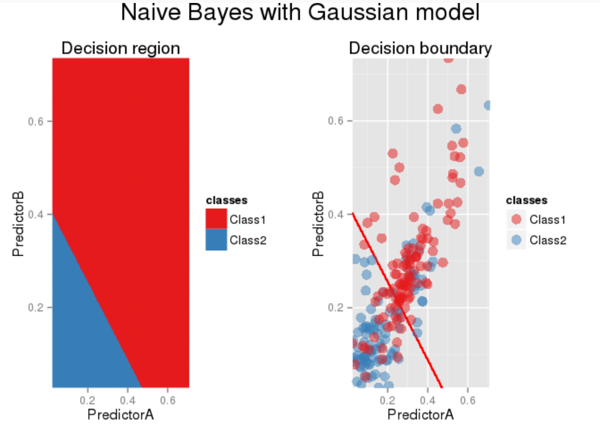
$$\mathbb{P}\left\{\mathbf{X}|Y\right\} = \prod_{i=1}^{d} \mathbb{P}\left\{X^{(i)}\middle|Y\right\} .$$

- Simple featurewise model: binomial if binary, multinomial if finite and Gaussian if continuous.

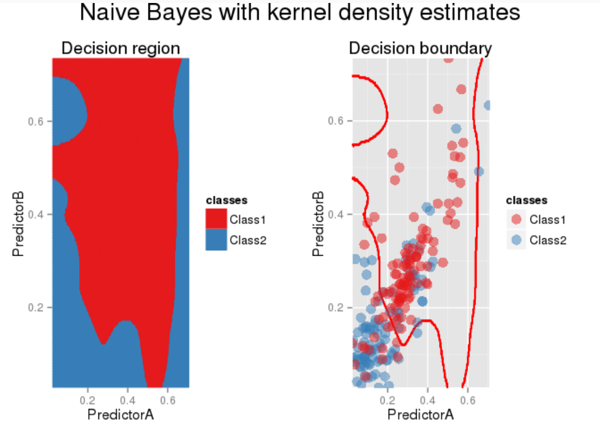
If all features are continuous, similar to the previous Gaussian but with a **diagonal covariance matrix**!

Very simple learning even in very high dimension!

Example: Naive Bayes



Example: Naive Bayes



Discriminant Analysis (Gaussian model)

The densities are modeled as multivariate normal, i.e., for all class k, conditionnally on $\{Y = k\}$,

$$\mathbf{X} \sim \mathcal{N}(\mu_k, \Sigma_k)\,.$$

Discriminant functions:

$$g_k(\mathbf{X}) = \ln(\mathbb{P}\{\mathbf{X}|Y = k\}) + \ln(\mathbb{P}\left\{Y = k\right\})\,.$$

In a two-classes problem, the optimal classifier is

$$f^* : x \mapsto 2\mathbb{1}\{g_1(x) > g_{-1}(x)\} - 1\,.$$

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 practice, we will need to estimate μ_k, Σ_k and $\pi_k := \mathbb{P}\{Y = k\}$.

Estimated proportions $\widehat{\pi}_k = \frac{n_k}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{Y_i=k\}}$.

Maximum likelihood estimate of $\widehat{\mu_k}$ and $\widehat{\Sigma_k}$ (explicit formulas).

The DA classifier then becomes

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

If $\Sigma_{-1} = \Sigma_1 = \Sigma$ then the **decision boundary is an affine hyperplane**.

The loglikelihood of the observations is given by

$$\log \mathbb{P}_\theta\left(X_{1:n}, Y_{1:n}\right) = \sum_{i=1}^{n} \log \mathbb{P}_\theta\left(X_i, Y_i\right) \,,$$

$$= -\frac{nd}{2}\log(2\pi) - \frac{n}{2}\log\det(\Sigma) + \left(\sum_{i=1}^{n} \mathbb{1}_{Y_i=1}\right)\log \pi_1 + \left(\sum_{i=1}^{n} \mathbb{1}_{Y_i=-1}\right)\log(1-\pi_1)$$

$$-\frac{1}{2}\sum_{i=1}^{n} \mathbb{1}_{Y_i=1}\left(X_i-\mu_1\right)'\Sigma^{-1}\left(X_i-\mu_1\right) - \frac{1}{2}\sum_{i=1}^{n} \mathbb{1}_{Y_i=-1}\left(X_i-\mu_{-1}\right)'\Sigma^{-1}\left(X_i-\mu_{-1}\right) \,.$$

This yields, for $k \in \{-1, 1\}$,

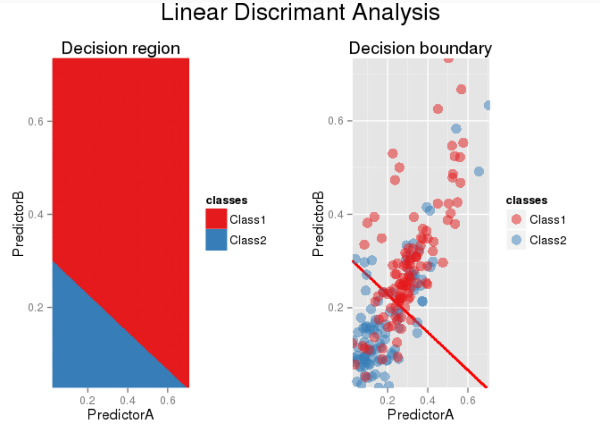
$$\widehat{\pi}_{\mathbf{k}}^{\mathbf{n}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{Y_i=k} \,,$$

$$\widehat{\mu}_{\mathbf{k}}^{\mathbf{n}} = \frac{1}{\sum_{i=1}^{n} \mathbb{1}_{Y_i=k}} \sum_{i=1}^{n} \mathbb{1}_{Y_i=k} X_i \,,$$

$$\widehat{\Sigma}^{n} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \widehat{\mu}_{Y_i}^{n}\right)\left(X_i - \widehat{\mu}_{Y_i}^{n}\right)' \,.$$

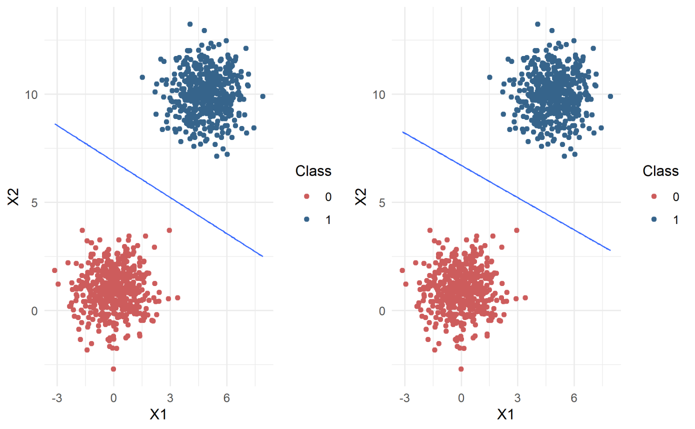
Remains to plug these estimates in the classification boundary.

Example: LDA

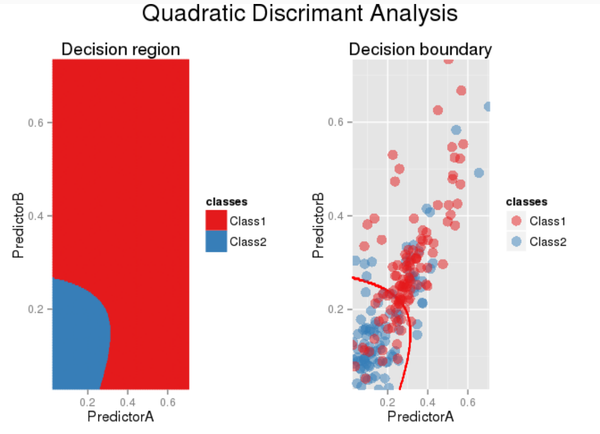


Linear Discriminant Analysis
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: LDA

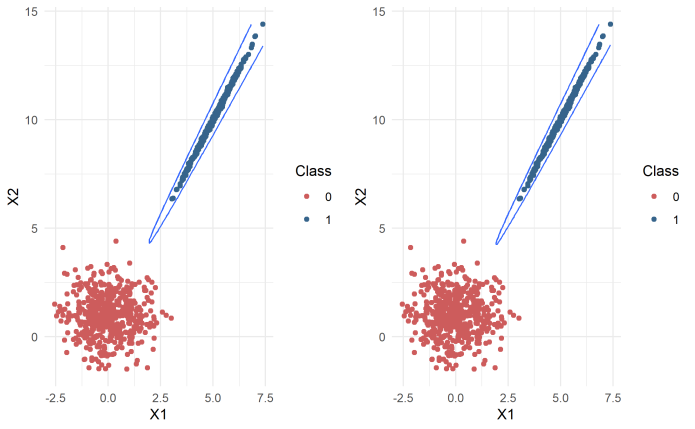


Example: QDA



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

Example: QDA


15
10
5
0
X2
-2.5
0.0
2.5
5.0
7.5
X1
Class
0
1
15
10
5
0
X2
-2.5
0.0
2.5
5.0
7.5
X1
Class
0
1

Packages

Function `svm` in package `e1071`.

Function `lda` and `qda` in package `MASS`.

Function `naive_bayes` in package `naivebayes`.

Plan

Setting

You have past/historical data from your platform, containing data about individuals $i = 1, \ldots, n$.

You have a **features** vector $\mathbf{x}_i \in \mathbb{R}^d$ for each individual i.

For each i, you know if he/she clicked ($y_i = 1$) or not ($y_i = -1$).

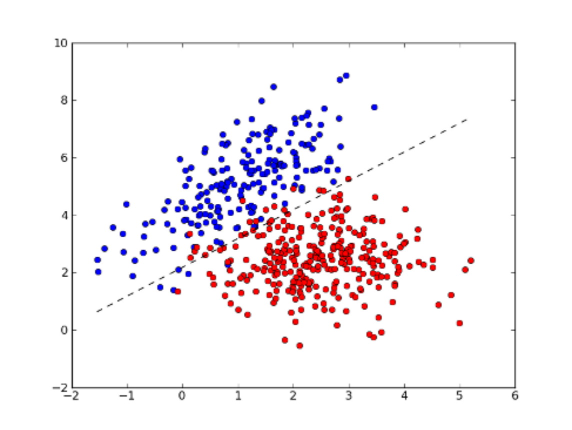
We call $y_i \in \{-1, 1\}$ the **label** of i.

Aim

Given a features vector $\mathbf{x}$ (with no corresponding label), predict a label $\hat{y} \in \{-1, 1\}$.

Use data $\mathcal{D}_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ to construct a **classifier**.

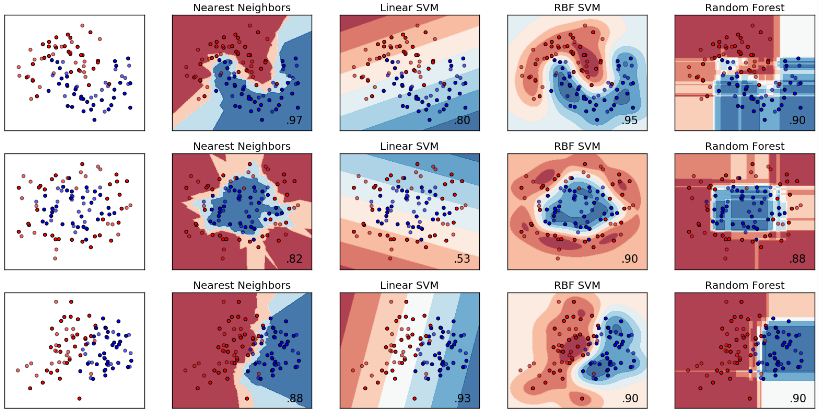
Geometrically



Learn a way to separate two "groups" of points

Classification

But, many ways to separate points!

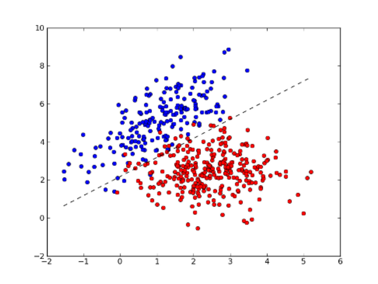


Simple to interpret and to implement.

On very large datasets (n is large, say $n \geq 10^6$), no other choice (training complexity).

Big data paradigm: lots of data $\Rightarrow$ simple methods are enough.

A linear classifier



Learn $\hat{\mathbf{w}} \in \mathbb{R}^d$ and $\hat{b}$ such that

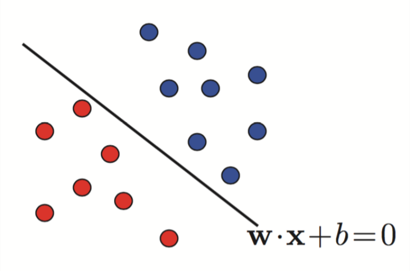
$$\hat{y} = \mathrm{sign}(\langle \mathbf{x}, \hat{\mathbf{w}} \rangle + \hat{b})$$

is a good classifier

Linearly separable data

A dataset is **linearly separable** if we can find an hyperplane H (linear classification rule) that puts

- Points $\mathbf{x}_i \in \mathbb{R}^d$ such that $y_i = 1$ on one side of the hyperplane
- Points $\mathbf{x}_i \in \mathbb{R}^d$ such that $y_i = -1$ on the other
- H do not pass through a point $\mathbf{x}_i$



Some geometry

A **hyperplane**

$$H_{\mathbf{w},\mathbf{x}} = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$$

is a translation of a set of vectors orthogonal to $\mathbf{w}$.

- $\mathbf{w} \in \mathbb{R}^d$ is a **non-zero vector normal** to the hyperplane.
- $b \in \mathbb{R}$ is a scalar.

Following for instance the results obtained for linear discriminant analysis and logistic regression, a **hyperplane $H_{\mathbf{w},b}$ may be used as a classifier** by defining

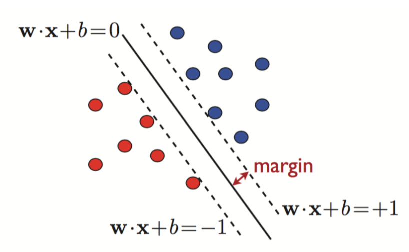
$$h_{\mathbf{w},b} : \mathbf{x} \mapsto \begin{cases} 1 & \text{if } \langle \mathbf{w}\,;\, \mathbf{x} \rangle + b > 0\,, \\ -1 & \text{otherwise}\,. \end{cases}$$

Some geometry

Definition of H is invariant by multiplication of $\mathbf{w}$ and b by a non-zero scalar

If H does not pass through any sample point $\mathbf{x}_i$, we can scale $\mathbf{w}$ and b so that

$$\min_{(\mathbf{x},y)\in\mathcal{D}_n} |\langle \mathbf{w}, \mathbf{x}\rangle + b| = 1$$



For such $\mathbf{w}$ and b, we call H the *canonical* hyperplane

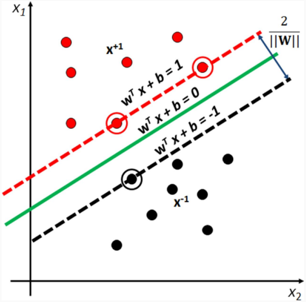
Some geometry

The distance of any point $\mathbf{x}' \in \mathbb{R}^d$ to H is given by

$$\frac{|\langle \mathbf{w}, \mathbf{x}' \rangle + b|}{\|\mathbf{w}\|}$$

So, if H is a canonical hyperplane, its margin is given by

$$\min_{(\mathbf{x},y)\in\mathcal{D}_n} \frac{|\langle \mathbf{w}, \mathbf{x} \rangle + b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}.$$



Summary

If $\mathcal{D}_n$ is strictly linearly separable, we can find a canonical separating hyperplane

$$H = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}.$$

that satisfies

$$|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \geq 1 \quad \text{for any} \quad i = 1, \ldots, n,$$

which entails that a point $\mathbf{x}_i$ is correctly classified if

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1.$$

The margin of H is equal to $1/\|\mathbf{w}\|$.

Linear SVM: separable case

Hard Support Vector Machines is a classification procedure which aims at building a linear classifier with the largest possible margin, i.e. the largest minimal distance between a point in the training set and the hyperplane.

The hyperplane which **correctly separates all training data sets with the largest margin** is obtained with:

$$(\widehat{w}_n, \widehat{b}_n) \in \underset{\substack{(w,b)\in\mathbb{R}^d\times\mathbb{R}^d\,;\,\|w\|=1,\\ \forall i\in\{1,\ldots,n\},\, Y_i(\langle w\,;\,X_i\rangle+b)>0}}{\mathrm{argmax}} \left\{\min_{1\leqslant i\leqslant n} |\langle w\,;\,X_i\rangle + b|\right\}.$$

Linear SVM: separable case

The **hard Support Vector Machines** procedure is equivalent to solving the following optimization problem:

$$(\widehat{w}_n, \widehat{b}_n) \in \underset{(w,b)\in\mathbb{R}^d\times\mathbb{R}\,;\,\|w\|=1}{\operatorname{argmax}} \left\{ \min_{1\leqslant i\leqslant n} Y_i\left(\langle w\,;\,X_i\rangle + b\right)\right\},$$

A solution to the hard Support Vector Machines optimization problem is obtained by setting $(\widehat{w}_n, \widehat{b}_n) = (w_\star/\|w_\star\|, b_\star/\|w_\star\|)$ where

$$(w_\star, b_\star) \in \underset{\substack{(w,b)\in\mathbb{R}^d\times\mathbb{R}\\ \forall i\in\{1,\ldots,n\},\, Y_i(\langle w\,;\,X_i\rangle+b)\geqslant 1}}{\operatorname{argmin}} \|w\|^2.$$

Linear SVM: separable case

A way of classifying $\mathcal{D}_n$ with maximum margin is to solve the following problem:

$$(w_\star, b_\star) \in \underset{\substack{(w,b)\in\mathbb{R}^d\times\mathbb{R} \\ \forall i\in\{1,\dots,n\},\, Y_i(\langle w\,;X_i\rangle+b)\geqslant 1}}{\operatorname{argmin}} \|w\|^2 .$$

This problem admits a **unique** solution

It is a **quadratic programming** problem, which is easy to solve numerically.

Dedicated optimization algorithms can solve this on a large scale very efficiently.

Linear SVM: separable case

The optimization problem is solved using Karush-Kuhn-Tucker's theorem).

There are $\alpha_i \geq 0$, $i = 1, \ldots, n$, called dual variables, such that the solution $(\mathbf{w}, b)$ of this problem satisfies:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \quad \text{and} \quad \alpha_i((y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) = 0 \;\; \text{for} \;\; i = 1, \ldots, n.$$

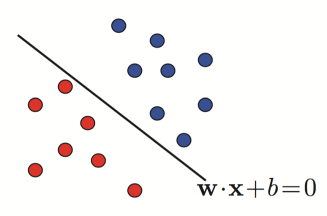
$\alpha_i \neq 0$ iff $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b = 1$, meaning that $\mathbf{x}_i$ is on the marginal hyperplane.

Weights vector $\mathbf{w}$ is a linear combination of the vectors $\mathbf{x}_i$ that belong to a marginal hyperplane.

Such points $\mathbf{x}_i$ are called **support vectors**.

Linear SVM: non-separable case

Have you ever seen a dataset that looks that this?



Restricting the problem to linearly separable training data sets is a somehow strong assumption.

Inequality constraints in the quadratic optimization problem can be relaxed.

Introduction of nonnegative variables $(\xi_i)_{1 \leqslant i \leqslant n}$ which quantify the nonfeasability of the constraint $y_i(\langle \mathbf{w} \, ; \, \mathbf{x}_i \rangle + b) \geqslant 1$.

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant 1 - \xi_i \, .$$

Linear SVM: non-separable case

Replace the constraint

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, n,$$

which is too strong by the **relaxed** one

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - s_i \quad \text{for all} \quad i = 1, \ldots, n,$$

for slack variables $s_1, \ldots, s_n \geq 0$.

Linear SVM: non-separable case

The original problem is then replaced by

$$(w_\star, b_\star, \xi_\star) \in \underset{\substack{(w,b,\xi)\in\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}_+^d \\ \forall i\in\{1,\ldots,n\},\, Y_i(\langle w\,;X_i\rangle+b)\geqslant 1-\xi_i}}{\operatorname{argmin}} \left\{\lambda\|w\|^2 + \frac{1}{n}\sum_{i=1}^n \xi_i\right\},$$

where $\lambda > 0$.

The **soft Support Vector Machines** algorithm simultaneously the margin of the linear classifier and the average value of these slack variables

Note that, if $(w_\star, b_\star)$ is solution to

$$(w_\star, b_\star) \in \underset{(w,b)\in\mathbb{R}^d\times\mathbb{R}}{\operatorname{argmin}} \left\{\lambda\|w\|^2 + \frac{1}{n}\sum_{i=1}^n (1 - Y_i\left(\langle w\,;\,X_i\rangle + b\right))_+\right\},$$

then $(w_\star/\|w_\star\|, b_\star, \xi_\star/\|w_\star\|)$ is solution to the soft SVM problem.

Linear SVM: non-separable case

The **soft SVM problem boils down** to computing:

$$(w_\star, b_\star, \xi_\star) \in \underset{\substack{(w,b,\xi)\in\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}_+^d \\ \forall i\in\{1,\dots,n\},\, Y_i(\langle w\,;X_i\rangle+b)\geqslant 1-\xi_i}}{\operatorname{argmin}} \left\{ \lambda\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \right\},$$

where $\lambda > 0$.

This problem admits a **unique** solution.

It is a **quadratic programming** problem, which is easy to solve numerically.

Dedicated optimization algorithms can solve this on a large scale very efficiently.

The hinge loss

This problem can be reformulated as follows

$$\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \max\Big(0, 1 - y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b)\Big),$$

or even better, by introducing the hinge loss

$$\ell(y, y') = \max(0, 1 - yy') = (1 - yy')_+,$$

the problem can be written as

$$\operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \ell(y_i, \langle \mathbf{x}_i, \mathbf{w}\rangle + b).$$

We'll see later more loss functions ℓ and explain the general setting hidden behind this.

The joint law of (X, Y) is **not assumed to belong to any parametric or semiparametric** family of models.

The classification risk **cannot be computed nor minimized**, it is instead estimated by the empirical classification risk defined as

$$\widehat{L}^n_{\mathrm{miss}}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{Y_i \neq f(\mathbf{X}_i)}\,,$$

where $(X_i, Y_i)_{1\leqslant i\leqslant n}$ are independent with the same distribution as (X, Y).

The classification problem then boils down to solving

$$\widehat{f}^n \in \operatorname*{argmin}_{f\in\mathcal{F}} \widehat{L}^n_{\mathrm{miss}}(f)\,,$$

for a chosen class $\mathcal{F}$ of classifiers.

Nonparametric classification based on the empirical risk minimization may seem appealing

It cannot be used to derive efficient practical classifiers due to **the computational cost of the optimization problem**.

The target loss function $\widehat{L}^n_{\mathrm{miss}}$ is replaced by **a convex surrogate and its minimization is constrained to a convex set of classifiers**.

For any convex function $\phi : \mathcal{X} \to \mathbb{R}$, it is possible to build a classifier f given by $f_\phi = \mathrm{sign}(\phi)$. The associated empirical classification is then

$$\widehat{L}^n_{\mathrm{miss}}(\phi) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{Y_i \neq f_\phi(X_i)} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{Y_i \phi(X_i) < 0}\,.$$

Replacing the indicator function by **any convex loss funtion ℓ yields a convex surrogate**:

$$\widehat{L}_{\mathrm{miss}}^{n,\mathrm{conv}}(\phi) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i\phi(X_i))\,.$$

Penalizing the smoothness of the function ϕ is penalized, $\widehat{L}_{\mathrm{miss}}^{n,\mathrm{conv}}$ may be replaced by

$$\widehat{L}_{\mathrm{miss}}^{n,\mathrm{conv}}(\phi) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i\phi(X_i)) + \lambda\|\phi\|^2\,,$$

where $\lambda > 0$ and $\|\cdot\|$ is a norm on the space $\mathcal{H}$.

The soft Support Vector Machines algorithm fits this framework with the affine base function $\phi : \mathbf{x} \mapsto \langle \mathbf{w}\,;\,\mathbf{x}\rangle + b$ and ℓ chosen as the hinge loss $\ell : x \mapsto (1-x)_+$ when the target function is penalized by its margin $\|w\|^2$.

Kernel trick

A useful case in practice consists in choosing $\mathcal{H}$ as a **Reproducing Kernel Hilbert Space** with positive definite kernel k on $\mathcal{X} \times \mathcal{X}$.

A function k on $\mathcal{X} \times \mathcal{X}$ is said to be **a positive definite kernel** if and only if it is symmetric and if for all $n \geqslant 1$, $(x_1, \ldots, x_n) \in \mathcal{X}^n$ and all $(a_1, \ldots, a_n) \in \mathbb{R}^n$,

$$\sum_{1 \leqslant i,j \leqslant n} a_i a_j k(x_i, x_j) \geqslant 0 \,.$$

The **Reproducing Kernel Hilbert Space** with kernel k is the only Hilbert space $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ such that for all $x \in \mathcal{X}$, $k(x, \cdot) \in \mathcal{H}$ and for all $x \in \mathcal{X}$ and all $f \in \mathcal{H}$,

$$f(x) = \langle f \,;\, k(x, \cdot) \rangle_{\mathcal{H}} \,.$$

Kernel trick

$k : \mathcal{X} \times \mathcal{X} :\to \mathbb{R}$ a positive definite kernel and $\mathcal{H}$ the RKHS with kernel k.

$$\widehat{\phi}^n_{\mathcal{H}} \in \underset{\phi \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i \phi(X_i)) + \lambda \|\phi\|^2_{\mathcal{H}} ,$$

where $\|\phi\|^2_{\mathcal{H}} = \langle \phi\, ;\, \phi \rangle$, is given by

$$\widehat{\phi}^n_{\mathcal{H}} : x \mapsto \sum_{i=1}^{n} \widehat{\alpha}_i k(X_i, x) ,$$

with

$$\widehat{\alpha} \in \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{j=1}^{n} \alpha_j Y_i k(X_j, X_i) \right) + \lambda \sum_{1 \leqslant i,j \leqslant n} \alpha_i \alpha_j k(X_i, X_j) \right\} .$$