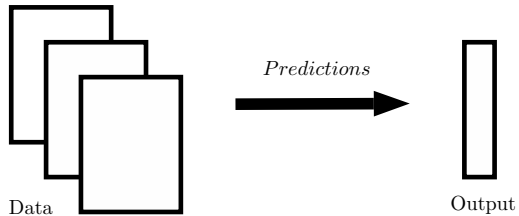
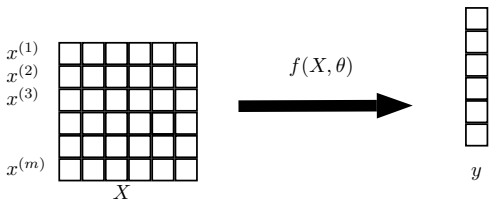
POLIMI GRADUATE SCHOOL OF MANAGEMENT

MACHINE LEARNING IN PYTHON - LAB SESSION

Andrea Mor - andrea.mor@polimi.it - https://github.com/polimi-door-group/BABD2024

SUPERVISED LEARNING





THE PROBLEM: BANK TELEMARKETING[1]

Attribute		Type	Description/Values
Personal	age	num	Age of the potential client
	job	cat	admin., blue- collar, entrepreneur, housemaid, ... , unknown
	marital_status	cat	divorced, married, single, unknown
	education	cat	basic.4y, basic.6y, basic.9y, high.school, ... , unknown
Bank	default	cat	The client has credit in default: no,yes,unknown
	housing	cat	The client has a housing loan contract: no,yes,unknown
	loan	cat	The client has a personal loan: no,yes,unknown
Campain	contact	cat	Communication type: cellular,telephone
	month	cat	Last month contacted: jan, feb ,..., dec
	day_of_week	cat	Last contact day : mon, tue,..., fri
	duration	num	Last contact duration (in seconds)
	campain	num	Number of contacts performed during this campaign
	pdays	num	Number of days that passed by after last contact
	previous	num	Number of contacts performed before this campaign
	poutcome	cat	Outcome of the prev. marketing campaign: failure,nonexistent,success
Economical	emp.var.rate	num	Employment variation rate in the last quarter
	cons.price.idx	num	Consumer price index in the last month
	cons.conf.idx	num	Monthly consumer confidence index
	euribor3m	num	Dayly Euro Interbank Offered Rate
	nr.employed	num	Number of employed citizens in the last quarter (thousands)
Target	**success**	**target**	**0: no, 1: yes**

[1] A data-driven approach to predict the success of bank telemarketing. S. Moroa, P. Cortez, P. Rita. Decision Support Systems, 62:22-31, 2014.

CLASSIFICATION PROBLEM

attributes

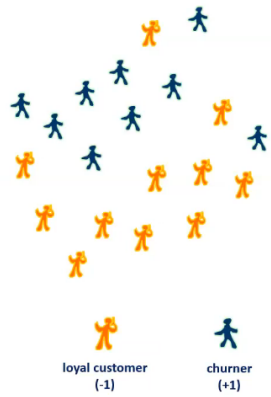
customers

Area	Pothers	Pmob	...	NumSMS	Class
2	0.14	0.59	...	18	1
3	0.26	0.35	...	9	-1
1	0.37	0.23	...	1	1
⋮	⋮	⋮	⋮	⋮	⋮
4	0.41	0.27	...	64	-1

past data

Area	Pothers	Pmob	...	NumSMS	Class
1	0.27	0.67	...	36	?
4	0.44	0.22	...	50	?
4	0.31	0.47	...	14	?
⋮	⋮	⋮	⋮	⋮	⋮
2	0.31	0.14	...	49	?

future data

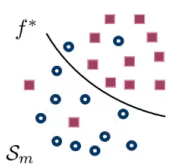


CLASSIFICATION FORMULATION

$\mathcal{S}_m = \{(\mathbf{x}_i, y_i),\ i \in \mathcal{M}\}$: **training set, where** $\mathbf{x}_i \in \Re^n$ **and** $y_i \in \mathcal{D}$

$\mathcal{H}$ **denotes a set of functions** $f(\mathbf{x}) : \Re^n \mapsto \mathcal{D}$

Classification problem: define a hypotheses space $\mathcal{H}$ **and a function** $f^* \in \mathcal{H}$ **which optimally describes the relationship between** $\mathbf{x}_i$ **and** y_i



$\Rightarrow$ $f^* \in \mathcal{H}$ $\Rightarrow$ $\overline{\mathbf{x}}$ $\Rightarrow$ $f(\overline{\mathbf{x}}) = -1$

learning prediction

new patient

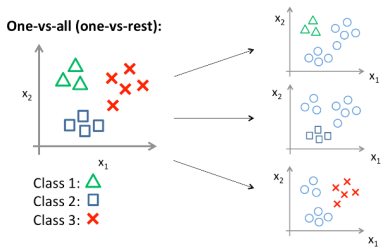
healthy !

healthy patient (-1)

ill patient (+1)

MULTI-CLASS CLASSIFICATION

1. **One-vs-Rest** We perform $|H|$ different binary classifications: one for every class.



[1]

   We decide based on a majority vote.
2. **One-vs-One** We perform $|H|(|H-1|)/2$ binary classifications: one for every pair of classes. We decide based on a majority vote.

[1]Image via www.cc.gatech.edu

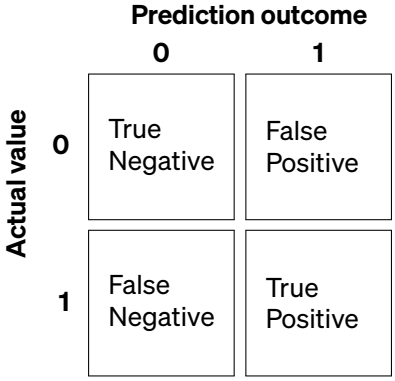
CLASSIFICATION MODELS

- Heuristics Methods
  - Nearest Neighbours
  - Classification Trees
- Probabilistic Methods
  - Bayesian Methods
- Regression Methods
  - Logistic regression
- Separation Methods
  - Support vector machine
  - Perceptron
  - Neural Networks

EVALUATION DIMENSIONS

- Prediction accuracy
- Speed
- Robustness
- Scalability
- Intepretability
- Rules effectiveness

QUALITY MEASURES - CONFUSION MATRIX



- **Accuracy** = $\frac{TP+TN}{TP+FP+TN+FN}$
  When: balanced problems, equal costs of FN and FP.
  When NOT: imbalanced problems.

- **Recall** (True Positive rate)=$\frac{TP}{FN+TP}$
  "proportion of true positives among actual positive"
  When: you want to be sure to catch all the 1s, at the cost of having false alerts.

- **Precision** (Positive Predictive value) = $\frac{TP}{TP+FP}$
  "proportion of true positives among positive predictions"
  When: you want to be sure that all that you predict as 1, are indeed 1, e.g., false alerts are costly.

- Precision and Recall are typically evaluated together.

QUALITY MEASURES - CONFUSION MATRIX

Actual value \ Prediction outcome	0	1
0	True Negative	False Positive
1	False Negative	True Positive

- **Geom. mean**=$\sqrt{\text{Precision} \times \text{Recall}}$
- **F-score**=$\frac{(1+\beta^2)}{\beta^2} \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = (1+\beta^2) \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) \cdot \text{Recall}}$
- **F1**: harmonic mean of Prec. and Recall
  $= \frac{2TP}{2TP+FP+FN}$
  Frequently the to-go metric.
- **F2**: 2x emphasis on Recall
  When: False Negatives are more costly.

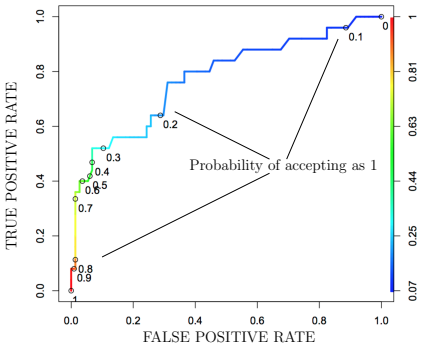
QUALITY MEASURES - CONFUSION MATRIX (AUXILLIARY)

Actual value \ Prediction outcome	0	1
0	True Negative	False Positive
1	False Negative	True Positive

- **False Positive rate** = $\frac{FP}{FP+TN}$
  "proportion of false positives among actual negatives"
  When: cost of dealing with positive prediction is high (e.g., false alarms).

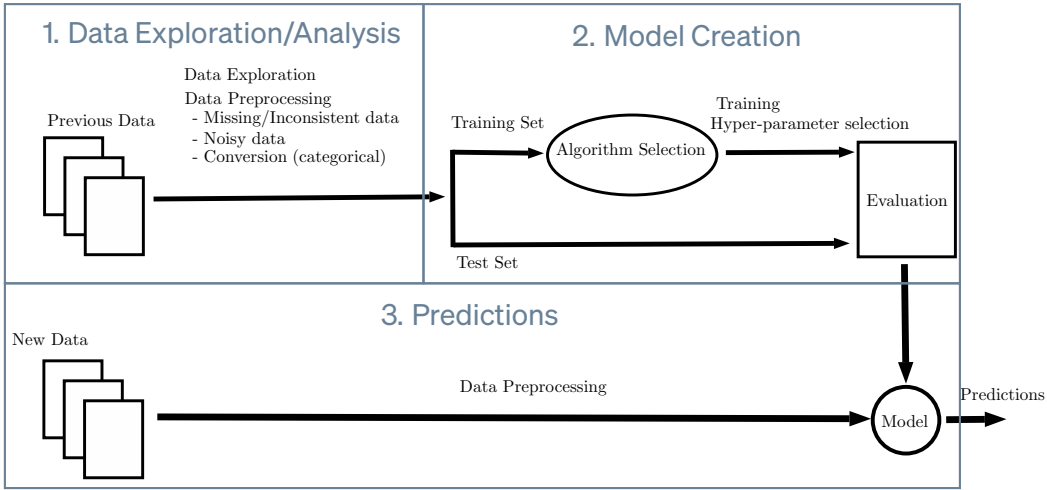
- **True Negative rate** (Specificity) = $\frac{TN}{TN+FP}$
  When: e.g., you want to be sure when you say something is safe for health.

QUALITY MEASURES - ROC CURVE & AUC

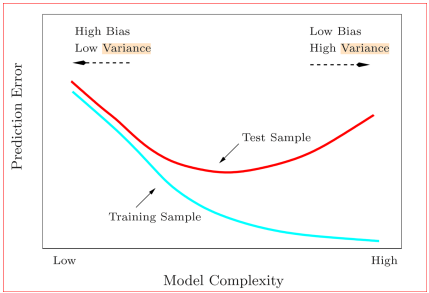


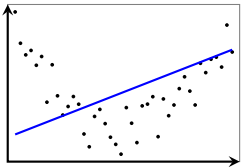
- If we accepting even with small probability then $TPR = FPR = 1$
- If we accepting just with high probability then $TPR = FPR = 0$
- The perfect classificator is the the point $(0, 1)$
- $AUC \in [0.5, 1]$ area under the curve is a quality measure of our algorithm.

WORKFLOW

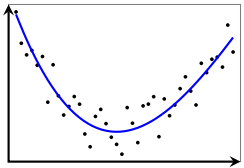

1. Data Exploration/Analysis
Data Exploration
Data Preprocessing
- Missing/Inconsistent data
- Noisy data
- Conversion (categorical)
Previous Data
2. Model Creation
Training Set
Algorithm Selection
Training
Hyper-parameter selection
Evaluation
Test Set
3. Predictions
New Data
Data Preprocessing
Model
Predictions

UNDER/OVER-FITTING

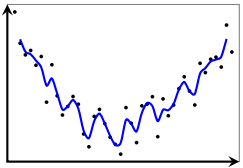




Underfitting

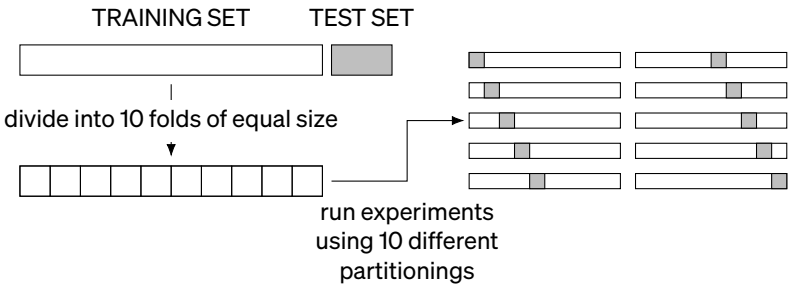


Balance



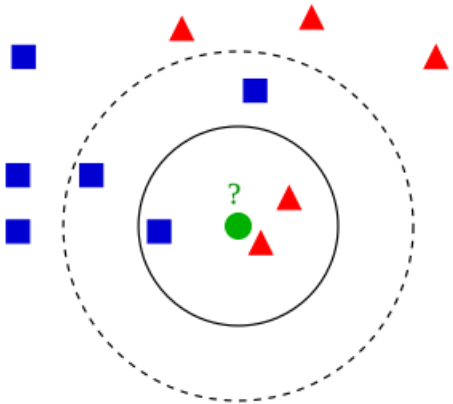
Overfitting

CROSS VALIDATION



THE ALGORITHMS

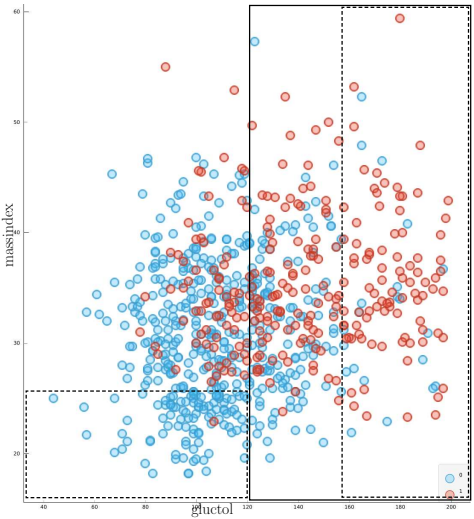
K-NEAREST NEIGHBORS

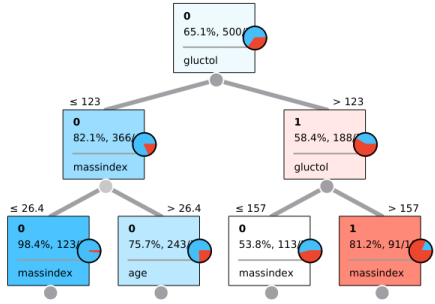


Main Parameters

- k : number of neighbors
- neighbor weights
- distances

CLASSIFICATION TREE

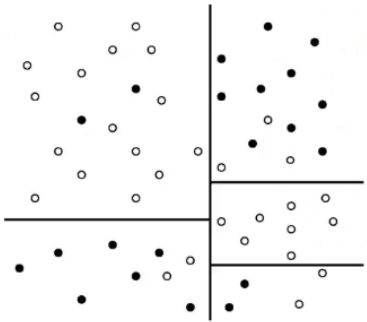




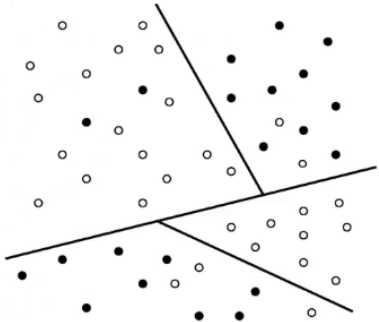
Tree types

- Binary tree (zero/two descendants)
- General trees
- Uni-variate tree ($X_j < b$)
- Multi-variate tree ($\sum_{j=1}^{n} w_j x_j < b$)

CLASSIFICATION TREE

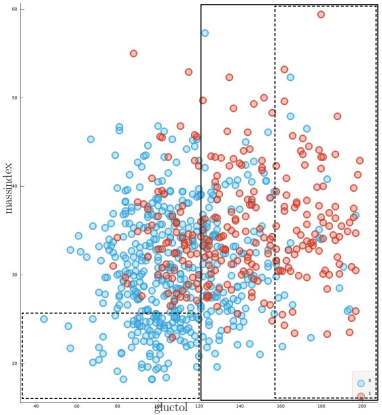


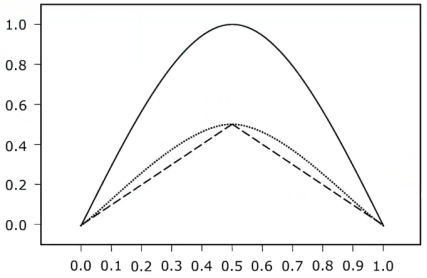
classification by an
axis parallel tree



classification by an
oblique tree

CLASSIFICATION TREE





- **Entropy index:**
- **Gini index:**
- **Miss-classification index:**

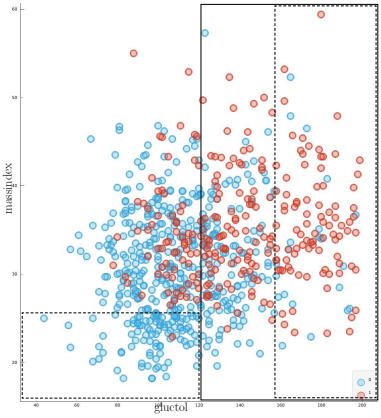
Split criteria

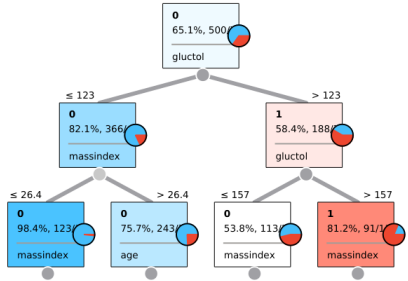
$$-\sum_{h=1}^{H} f_h \log_2 f_h$$

$$1 - \sum_{h=1}^{H} f_h^2$$

$$1 - \max_h f_h$$

CLASSIFICATION TREE





Main Parameters

- impurity measure: “gini”, “entropy”
- max_depth
- min_samples_split: minimum number of samples to split an internal node
- min_sample_leaf: minimum number of samples required to be at a leaf node

NAIVE BAYESIAN CLASSIFIER

- Bayes Theorem

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)\,P(y)}{\sum_{l=1}^{H} P(\mathbf{x}|y)\,P(y)}$$

- Maximum a posteriori hypothesis

$$y_{MAP} = \arg\max_{y \in \mathcal{H}} P(y|\mathbf{x}) = \arg\max \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- Independence (Naive)

$$P(\mathbf{x}|y) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y) = \prod_{j=1}^{n} P(x_j|y)$$

NAIVE BAYESIAN CLASSIFIER

- Categorical/discrete attributes

$$P(x_j|y) = P(x_j = r_{jk}|y = v_h)$$

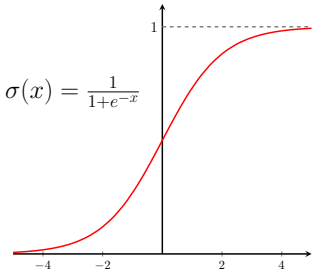
$\rightarrow$ empirical frequency of the observed value on the class v_h

- Numerical attribute

$$P(x_j|y) \sim N(\mu_{jh}, \sigma_{jh})$$

$\rightarrow$ assuming Gaussian density with empirical parameters

LOGISTIC REGRESSION



The model postulates that

$$\log\left[\frac{P(y=1|x)}{P(y=0|x)}\right] = w^T x$$

then if $p = P(y=1|x)$:

$$\frac{p}{1-p} = e^{w^T x}$$

$$\Rightarrow p = P(y=1|x) = \frac{e^{w^T x}}{1+e^{w^T x}} = \frac{1}{1+e^{-w^T x}},$$

$$P(y=0|x) = 1-p = \frac{1}{1+e^{w^T x}} = \frac{e^{-w^T x}}{1+e^{-w^T x}}$$

LOGISTIC REGRESSION

Therefore if we maximizes the likelihood

$$L(w) := P(Y_1 = y_1, \ldots, Y_m = y_m | w, x_1, \ldots, x_m) = \Pi_{i=1}^{n} P(Y_i = y_i | w, x_1, \ldots, x_m).$$

Assuming independence:

$$L(w) := \prod_{i|y_i=1} p(x_i) \cdot \prod_{i|y_i=0} (1 - p(x_i))$$

$$L(w) := \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

is equivalent to maximize the log likelihood

$$l(w) = \sum_{i=1}^{n} (y_i log(p(x_i)) + (1 - y_i) log(1 - p(x_i)))$$

$$= \sum_{i=1}^{n} (y_i log(\frac{p(x_i)}{1 - p(x_i)}) + log(1 - p(x_i)))$$

Given the quantities defined above

$$l(w) = \sum_{i=1}^{n} (y_i w^T x_i) - log(1 + e^{-w^T x_i}))$$

Therefore we aim to resolve the following optimization problem

$$\max_w l(w)$$

LOGISTIC REGRESSION

Regulation term(s) can be considered (recall that max $l(w)$ is equivalent to min $-l(w)$):

$$\min_w \frac{1}{2}||w||^2 - C \cdot l(w)$$

Main Parameters

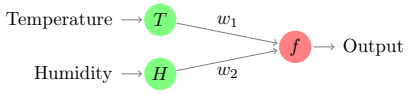
- C: Inverse of regularization strength
- Resolution algorithm parameters:
  - solver: lbfgs, newton-cg, liblinear, sag, saga.
  - tol: Tolerance for stopping criteria.
  - max_iter: max. number of iterations
  - n_jobs: Number of CPU cores

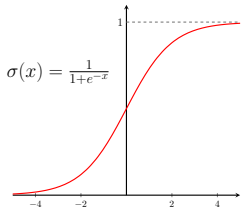
MULTI-LAYER PERCEPTRON - SMALL EXAMPLE

Temp. [C]	20	31	15	18	21
Humidity [%]	40	36	23	45	30
Prob. Rain	0.70	0.52	0.55	0.73	0.60

MULTI-LAYER PERCEPTRON - SMALL EXAMPLE

Temp. [C]	20	31	15	18	21
Humidity [%]	40	36	23	45	30
Prob. Rain	0.70	0.52	0.55	0.73	0.60





$$f(T, H, w_1, w_2) = \underbrace{\sigma}_{\text{activation function}}(w_1 \cdot T + w_2 \cdot H)$$

$$\min_{w_1, w_2} \sum_{i=1}^{5} [P_i(\text{rain}) - f(T_i, H_i, w_1, w_2)]^2$$

For a classification problem we can use the Likelihood as cost function.

MULTI-LAYER PERCEPTRON - SMALL EXAMPLE

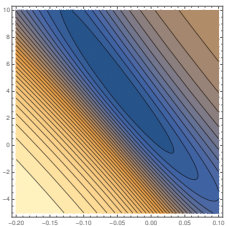
$$\min_{w_1,w_2} \sum_{i=1}^{5} [P_i(\text{rain}) - f(T_i, H_i, w_1, w_2)]^2$$

$$= \min \left[0.7 - 1/(1 + e^{-(w_1 \cdot 20 + w_2 \cdot 0.4)})\right]^2 + \left[0.52 - 1/(1 + e^{-(31 \cdot w_1 + w_2 \cdot 0.36)})\right]^2 + \cdots$$

MULTI-LAYER PERCEPTRON - SMALL EXAMPLE

$$\min_{w_1,w_2} \sum_{i=1}^{5} [P_i(\text{rain}) - f(T_i, H_i, w_1, w_2)]^2$$

$$= \min \left[0.7 - 1/(1 + e^{-(w_1 \cdot 20 + w_2 \cdot 0.4)})\right]^2 + \left[0.52 - 1/(1 + e^{-(31 \cdot w_1 + w_2 \cdot 0.36)})\right]^2 + \cdots$$

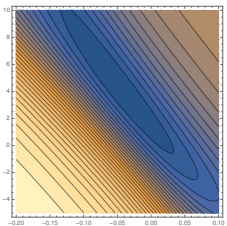


$$(w_1^*, w_2^*) = (-0.044, 4.147)$$

MULTI-LAYER PERCEPTRON - SMALL EXAMPLE

$$\min_{w_1,w_2} \sum_{i=1}^{5} [P_i(\text{rain}) - f(T_i, H_i, w_1, w_2)]^2$$

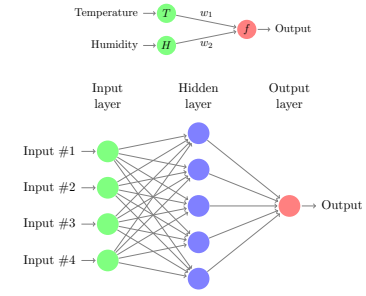
$$= \min \left[0.7 - 1/(1 + e^{-(w_1 \cdot 20 + w_2 \cdot 0.4)})\right]^2 + \left[0.52 - 1/(1 + e^{-(31 \cdot w_1 + w_2 \cdot 0.36)})\right]^2 + \cdots$$

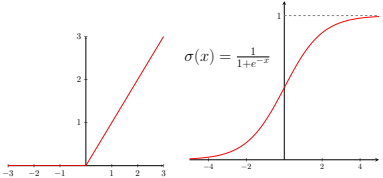


$$(w_1^*, w_2^*) = (-0.044, 4.147)$$

Temp. [C]	20	31	15	18	21
Humidity [%]	40	36	23	45	30
Prob. Rain	0.70	0.52	0.55	0.73	0.60
Predicted	0.70	0.56	0.58	0.75	0.60
Error	**0.0**	**-0.04**	**-0.03**	**-0.02**	**0.0**

MULTI-LAYER PERCEPTRON



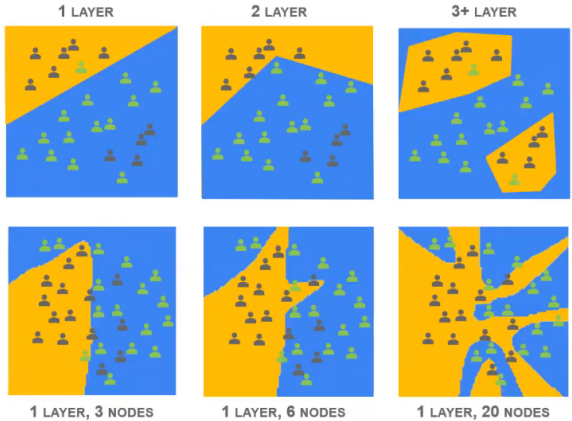


ReLU Sigmoid

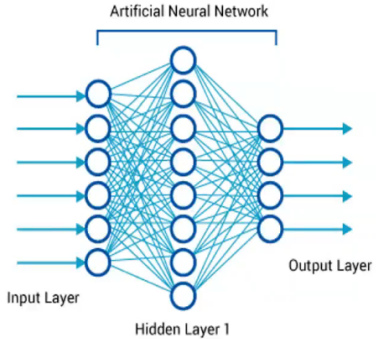
Main Parameters

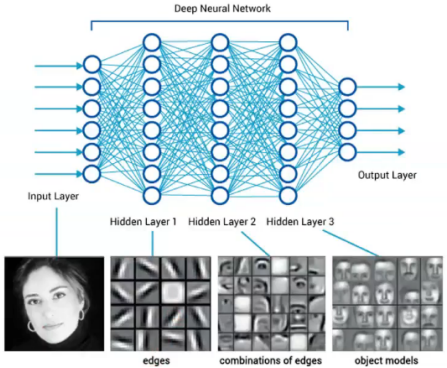
- hidden_layer_sizes: $(n_1, n_2, \ldots, n_L)$
- activation: identity, logistic, tanh, relu
- alpha regularization term parameter
- Resolution algorithm parameters: solver, tol, batch_size, learning_rate, max_iter.

NEURAL NETWORK



DEEP LEARNING





CLASSIFICATION AND STATISTICAL LEARNING THEORY

Given an hypothesis space $\mathcal{F}$, a predictive function $f \in \mathcal{F}$, and a loss function $V(y, f(\mathbf{x}))$, we define

- the **empirical risk** on a train set $\mathcal{T}$ as

$$R_{emp}(f) = \frac{1}{m} \sum_{i=1}^{m} V(y_i, f(\mathbf{x_i}))$$

- and the **expected risk** as

$$R(f) = \frac{1}{2} \int V(y, f(\mathbf{x}))\, dP(\mathbf{x}, y)$$

Overfitting arises when the difference between these errors is large.

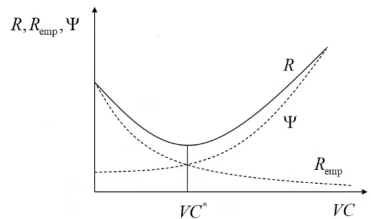
VC-DIMENSION

The Vapnik–Chervonenkis (VC) dimension is the maximum number of points that can be correctly classified by classifiers in $\mathcal{F}$. We can prove that

$$R(f) - R_{emp}(f) \leq \sqrt{\frac{1}{t}(\gamma \log(2t/\gamma) - \log(\eta/4) + 1)} = \Psi(t, \gamma, \eta)$$

with probability $1 - \eta$, where

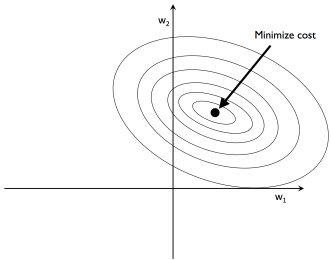
- t is the number of training points
- γ is the VC-dimension

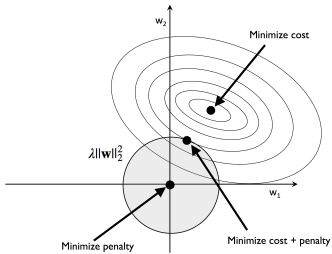


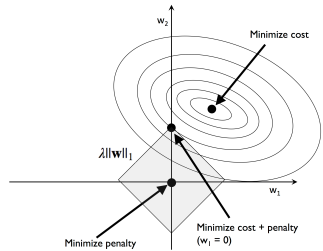
REGULARIZATION

Structural risk minimization (SRM)

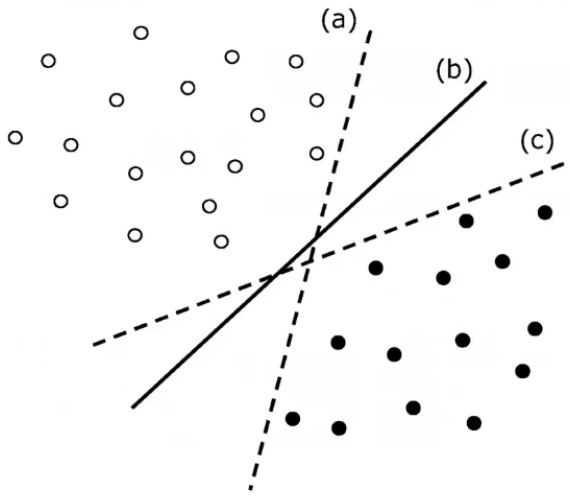
$$\hat{R}(f) = \frac{1}{t}\sum_{i=1}^{t} V(y_i, f(\mathbf{x_i})) + \lambda \|f\|_K^2$$



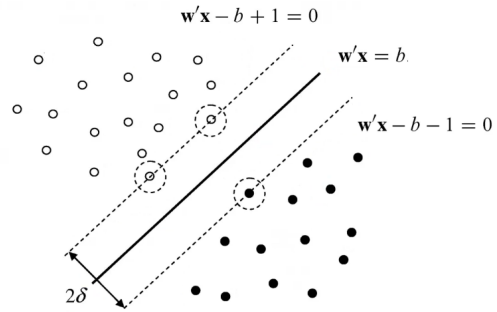




SUPPORT VECTOR MACHINE - LINEARLY SEPARABLE



SVM - LINEARLY SEPARABLE



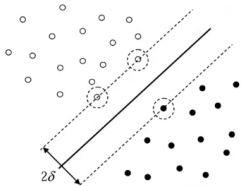
$$\delta = \frac{2}{\|w\|}, \qquad \|w\| = \sqrt{\sum_{j \in \mathcal{N}} w_j^2}$$

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$

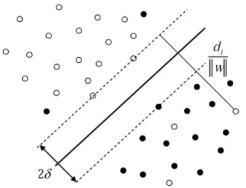
$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad i \in \mathcal{M}$$

SVM - GENERAL CASE

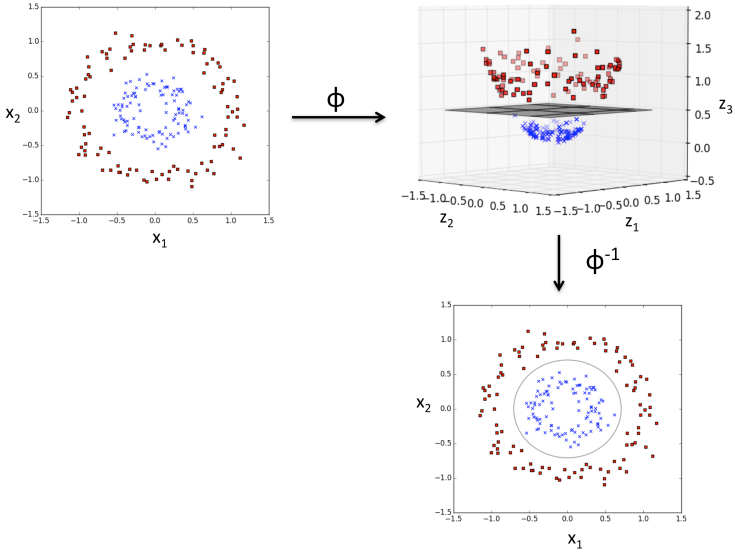
$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$
$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad i \in \mathcal{M}$$



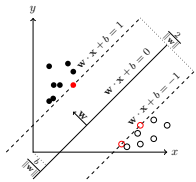
$$\min_{\mathbf{w},b,d} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + \lambda \sum_{i=1}^{m} d_i$$
$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - d_i \quad i \in \mathcal{M}$$
$$d_i \geq 0 \quad i \in \mathcal{M}$$

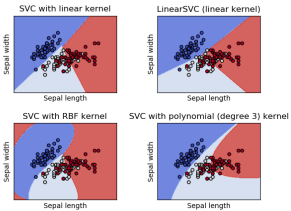


SVM - KERNELS



SVM - GENERAL CASE





$$\min_{w,b,d} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} d_i$$

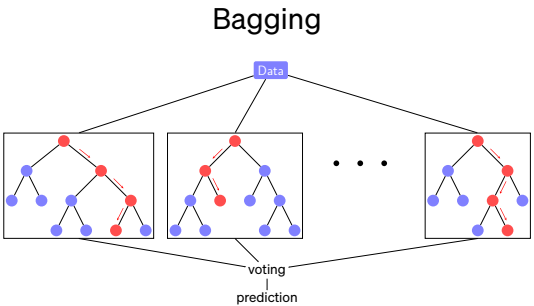
$$\text{subject to } y_i(w^T \underbrace{\phi(x_i)}_{\text{kernel}} - b) \geq 1 - d_i,$$

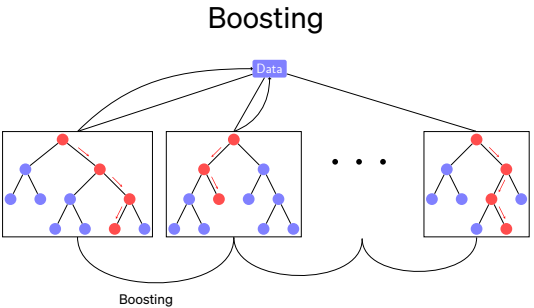
$$d_i \geq 0$$

Main Parameters

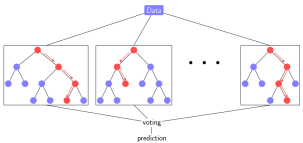
- C: Inverse of regularization strength
- kernel:
  - linear: $x'x$
  - poly: $(\gamma x'x + r)^d$
  - rbf: $exp(-\gamma||x - x'||^2)$
  - sigmoid: $\tanh(\gamma x'x + r)$
- degree(d), gamma(γ), coef0(r)
- Resolution algorithm parameters

ENSEMBLE METHODS





RANDOM FOREST

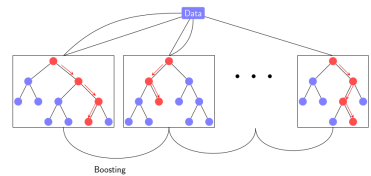


1. Create different (simple) tree models (stumps)
2. Each model is created with a subset of observation/features ($\sim$2m/3)
3. We combine the prediction of all trees

Main Parameters

- n_estimators: Number of trees
- max_features: Number of features selected for the split
- bootstrap=False: Use all samples
- Tree parameters

ADABOOST



1. Assign equal weights to observations $w_i^{(0)} = 1/m$
2. For $k = 1, \ldots, K$
   - Select a sample of observations based on the weights.
   - Create the k-th weak learner and compute predictions $x^{(k)}$
   - Compute the model **weighted** error and assign its coefficient:

$$\alpha^{(k)} = \lambda \times \log((1 - error)/error)$$

   - Update sample weights:

$$w_i^{(k+1)} \propto w_i^{(k)} \times \exp(-\alpha^{(k)} y_i \hat{x}_i^{(k)})$$

3. Final weighted prediction

Main Parameters

- n_estimators: Number of estimators (K)
- base_estimator: Weak estimator type
- learning_rate: weights of estimator in final decision (λ)

THANK YOU