# Machine Learning Basics

**Claudio Gentile**
Google Research, NY
`cla.gentile@gmail.com, cgentile@google.com`

Pula

Sept 17th, 2019

Deep Learning @ INAF

## Goal and content

Goal: Give quick introduction to basic ideas and methods in Machine Learning

Content:

## Introduction/1

### Machine Learning:

- Extraction of knowledge (or regularity patters) from sources of data (images, text, web server logs, signals, etc.)
- Knowledge extraction $\Longrightarrow$ (inductive) Inference
- Delivers both methodology and practical tools to solve tasks which are not easy to formalize and/or are intrinsically complex

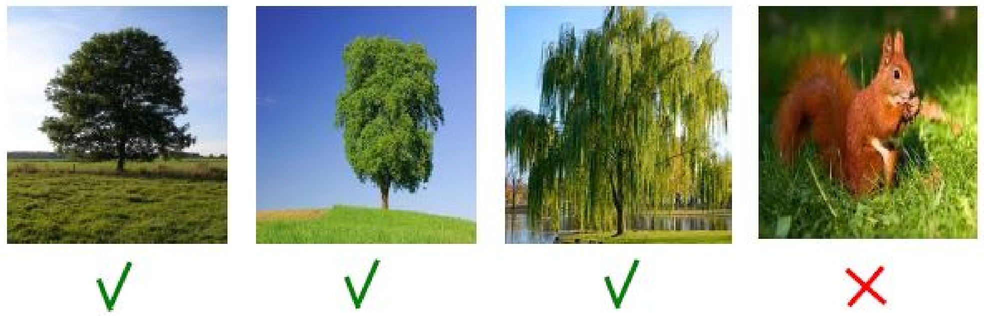
(Supervised) Machine Learning algorithm:
Machine/automated tool that acquires knowledge/desired behavior not through direct instruction but through examples of such behavior (supervision)

- Widespread interest in both Science and Industry
- Requires (fast) computing tools

## Introduction/2

Basic assumption: Underlying regularity in the source of data

For instance:
Want to learn the concept of ”tree” out of database of images of trees (positive examples) and non-trees (negative examples)

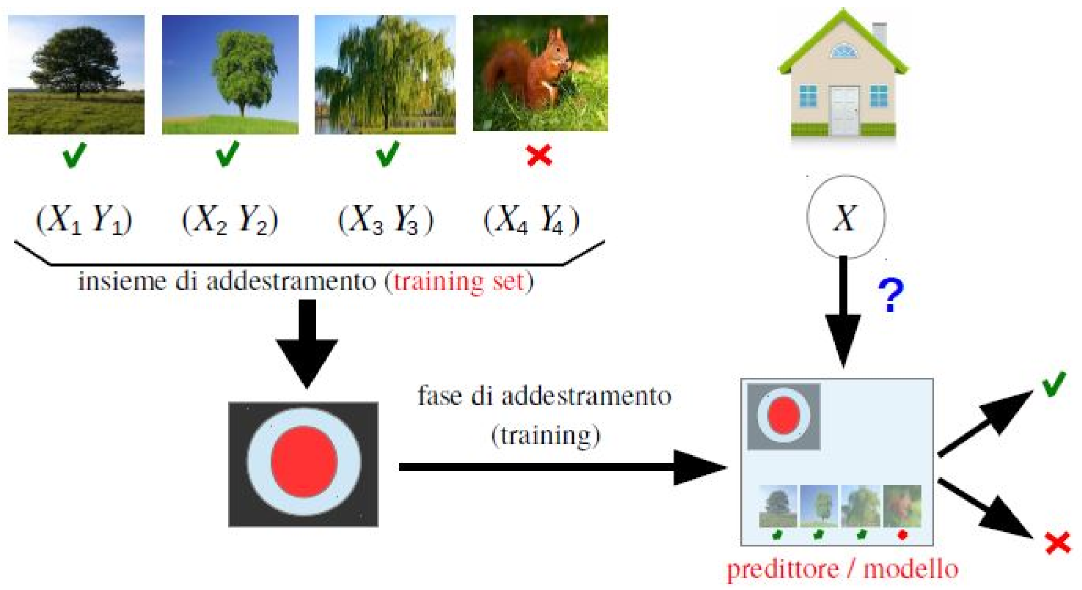


Trees have something statistically significant in common that need to be learnt based on examples

Basic task:
Write a computer program (any language you like) that, given in input a bitmap image, outputs ”1” if image contains a tree, and ”0” otherwise

## Introduction/3

(Supervised) machine learning algorithm:


✓
✓
✓
✗
$(X_1\ Y_1)$ $(X_2\ Y_2)$ $(X_3\ Y_3)$ $(X_4\ Y_4)$
insieme di addestramento (training set)
$X$
?
fase di addestramento
(training)
predittore / modello
✓
✗

- **Input:** Training set (e.g. set of images)
- **Output:** Predictor (or model, hypothesis) ...
- ... which is able to predict well unseen patterns (for given performance measure)

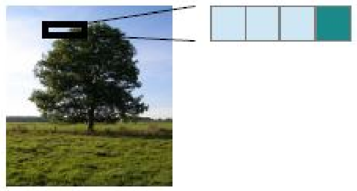
## Introduction/4

Some relevant questions:

- Exploiting prior knowledge (contextual, etc.) HERE I AM
- Data Preprocessing
- Choice of knowledge representation (or geometry):
  Simple and natural for task at hand, but complex enough to be able to represent underlying phenomenon (inductive bias)

  $\Longrightarrow$ More recently: Learning representations (e.g. Deep Learning)
- Design of learning algorithms
- Performance evaluation (conflicting needs):
  - error trends (loss function)
  - training and test time
  - flexible and easy to use
  - easy to interpret (symbolic vs. numeric)
  - easy to implement in hardware (GPU, etc.)
  - fairness, privacy, . . .

## An example: Naive Bayes/1



- Natural probabilistic viewpoint
- $A_1, \ldots, A_d$ are attributes or features
- $Y \in \{0, 1\}$ is category or class
- Given input test pattern
$$\boldsymbol{x} = (x_1, \ldots, x_d)$$
classify $\boldsymbol{x}$ by assigning category
- Classification based on class $y \in \{0, 1\}$ maximizing posterior probability
$$\Pr(Y = y \,|\, A_1 = x_1 \ldots A_d = x_d) = \frac{\Pr(A_1 = x_1 \ldots A_d = x_d \,|\, Y = y) \times \Pr(Y = y)}{\Pr(A_1 = x_1 \ldots A_d = x_d)}$$
- $\Pr(A_1 = x_1 \ldots A_d = x_d \,|\, Y = y)$ is class-conditioned probability
- $\Pr(Y = y)$ is prior probability of class $y$
- Denominator is irrelevant (independent of $y$)

## An example: Naive Bayes/2

- Naive Bayes assumption
- value of attributes $A_1, \ldots, A_d$ independent given class $Y$

$$\Pr(A_1 = x_1 \ldots A_d = x_d \,|\, Y = y) = \prod_{i=1}^{d} \Pr(A_i = x_i \,|\, Y = y)$$

- Naive Bayes classifier:
  Classify $\boldsymbol{x}$ with class $y \in \{0, 1\}$ maximizing

$$\prod_{i=1}^{d} \Pr(A_i = x_i \,|\, Y = y) \times \Pr(Y = y)$$

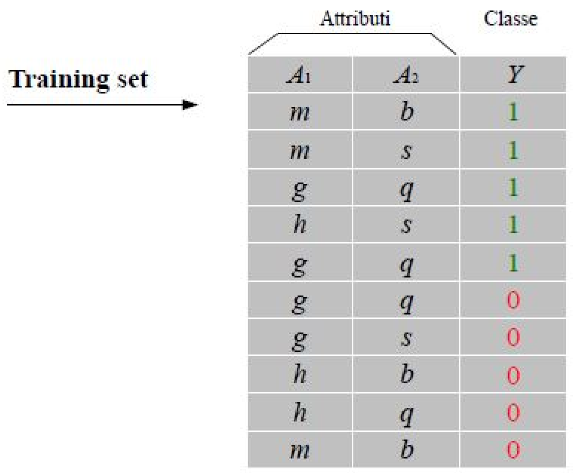
- Training phase: estimate (frequency counts) on training set

$$\Pr(Y = y) \qquad y = 0, 1$$

and

$$\Pr(A_i = x_i \,|\, Y = y) \qquad y = 0, 1, \quad i = 1 \ldots d$$

## An example: Naive Bayes/3



Training set

| Attributi | | Classe |
|---|---|---|
| $A_1$ | $A_2$ | $Y$ |
| $m$ | $b$ | 1 |
| $m$ | $s$ | 1 |
| $g$ | $q$ | 1 |
| $h$ | $s$ | 1 |
| $g$ | $q$ | 1 |
| $g$ | $q$ | 0 |
| $g$ | $s$ | 0 |
| $h$ | $b$ | 0 |
| $h$ | $q$ | 0 |
| $m$ | $b$ | 0 |

$\Pr(Y = 1) = 1/2$ $\quad$ $\Pr(Y = 0) = 1/2$

$\Pr(A_1 = m \,|\, 1) = 2/5$ $\quad$ $\Pr(A_1 = g \,|\, 1) = 2/5$ $\quad$ $\Pr(A_1 = h \,|\, 1) = 1/5$
$\Pr(A_1 = m \,|\, 0) = 1/5$ $\quad$ $\Pr(A_1 = g \,|\, 0) = 2/5$ $\quad$ $\Pr(A_1 = h \,|\, 0) = 2/5$

$\Pr(A_2 = b \,|\, 1) = 1/5$ $\quad$ $\Pr(A_2 = s \,|\, 1) = 2/5$ $\quad$ $\Pr(A_2 = q \,|\, 1) = 2/5$
$\Pr(A_2 = b \,|\, 0) = 2/5$ $\quad$ $\Pr(A_2 = s \,|\, 0) = 1/5$ $\quad$ $\Pr(A_2 = q \,|\, 0) = 2/5$

How to classify test pattern $(A_1 = m, A_2 = q)$ ?

$$\Pr(Y = 1) \cdot \Pr(A_1 = m \,|\, Y = 1) \cdot \Pr(A_2 = q \,|\, Y = 1) = \frac{1}{2} \cdot \frac{2}{5} \cdot \frac{2}{5} = \frac{2}{25}$$
$$\Pr(Y = 0) \cdot \Pr(A_1 = m \,|\, Y = 0) \cdot \Pr(A_2 = q \,|\, Y = 0) = \frac{1}{2} \cdot \frac{1}{5} \cdot \frac{2}{5} = \frac{1}{25}$$

$\Longrightarrow$ On this training set Naive Bayes classifies $(A_1 = m, A_2 = q)$ as 1

## General Bayesian approach/1

More general ingredients for Bayes approach:

- Loss function $\ell : \underbrace{\mathcal{Y}}_{class\ \ set} \times \underbrace{\widehat{\mathcal{Y}}}_{decision\ \ set} \rightarrow \mathbf{R}^{+}$

- $\boldsymbol{x}$-Conditional risk for decision $\widehat{y}$ :

$$R(\widehat{y}\,|\,\boldsymbol{x}) = \int \ell(y, \widehat{y})\, \overbrace{\mathsf{Pr}(y|\boldsymbol{x})}^{posterior}\, dy$$

where

$$\mathsf{Pr}(y|\boldsymbol{x}) = \frac{\overbrace{\mathsf{Pr}(\boldsymbol{x}|y)}^{class-conditioned} \quad \cdot \quad \overbrace{\mathsf{Pr}(y)}^{prior}}{\int \mathsf{Pr}(\boldsymbol{x}|y)\,\mathsf{Pr}(y)dy} \quad \propto \quad \mathsf{Pr}(\boldsymbol{x}|y)\,\mathsf{Pr}(y)$$

## General Bayesian approach/2

- From training set, get estimate $\widehat{\Pr}(\boldsymbol{x}|y)$ of class-conditioned prob. Often by parametric/simplifying assumptions, e.g.,
  - Gaussian mean $\mu$, covariance $\Sigma$
  - Naive Bayes

$$\Pr(\boldsymbol{x} \,|\, y) = \prod_{i=1}^{d} \underbrace{\Pr(x_i \,|\, y)}_{1-dim.\,parametric\,family}$$

- Replace $\Pr(\boldsymbol{x}|y)$ by $\widehat{\Pr}(\boldsymbol{x}|y)$ and classify input pattern $\boldsymbol{x}$:

$$Bayes(\boldsymbol{x}) = \text{argmin}_{y \in \hat{\mathcal{Y}}} R(\hat{y} \,|\, \boldsymbol{x})$$

- Maximum likelihood estimator when prior is uniform

- E.g. when $\mathcal{Y} = \hat{\mathcal{Y}} = \{1, 2, \ldots, c\}$ and $\ell(y, \hat{y}) = \{\hat{y} \neq y\}$ (0-1 loss)

$$R(\hat{y} \,|\, \boldsymbol{x}) = 1 - \Pr(\hat{y} \,|\, \boldsymbol{x})$$

$\Longrightarrow$

$$Bayes(\boldsymbol{x}) = \text{argmax}_{\hat{y} \in \hat{\mathcal{Y}}} \Pr(\hat{y} \,|\, \boldsymbol{x})$$

## Decision Trees/1

Non-metric method

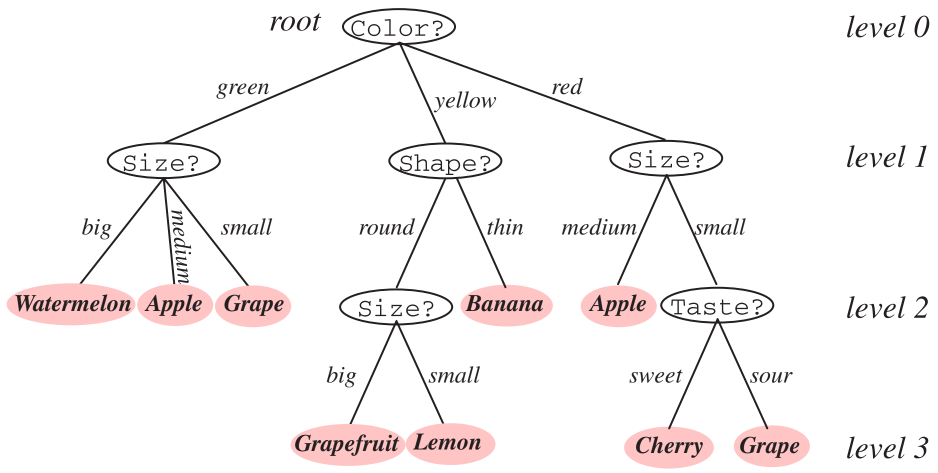


**FIGURE 8.1.** Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, `Size?`, appears in different places in the tree and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

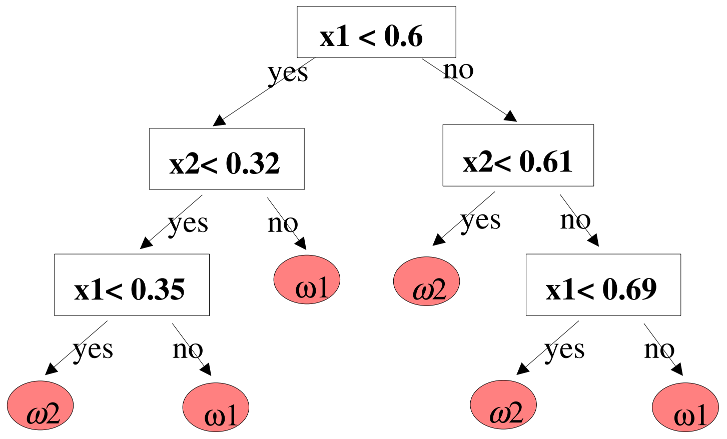
Encodes a function

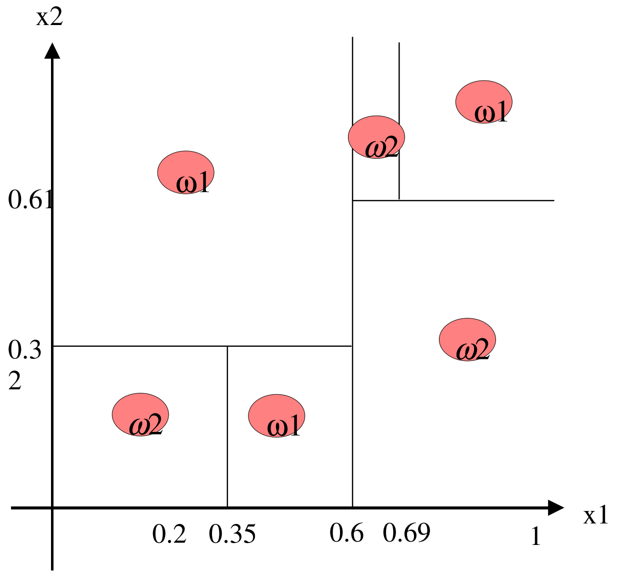
$$\{color,\ shape,\ taste,\ size\} \to \{fruit\}$$

Very interpretable representation !

## Decision Trees/2

Further example (features need not be discrete)





It's a function

$\mathbf{R}^2 \rightarrow \{\omega_1, \omega_2\}$

## Decision Trees/3

Useful when:

- Features are discrete or discretized (finite no. of values)
  – like Naive Bayes !
- Interpretable representation needed
- Training Set contains missing values (robustness)

## Decision Trees/4: Construction/1

ID3, C4.5 (Quinlan): top-down construction

Q: “Which feature at the root?”
A: “The one that best discriminates Training Set”

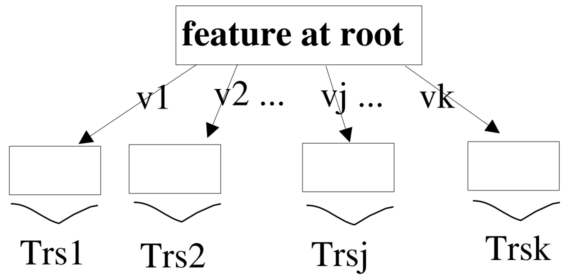
Tag each child node with sub-Training Set

Q: “Which feature in each such child node ? ”
A: “The one that best discriminates corresponding sub-Training Set”

etc.

## Decision Trees/4: Construction/2

Q: "Which feature at the root?"
A: "The one that (alone) best discriminates Training Set"



Q: Which feature at child node $j$ ?
A: The one that (alone) best discriminates TrS$_j$

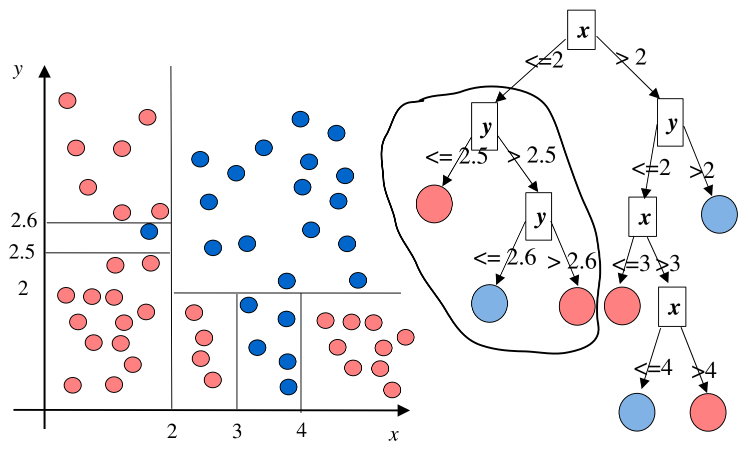
- Best discrimination based on entropy reduction on sub-Training Set

$$\sum_{i=1}^{c} p_i \log(1/p_i)$$

  $p_i$ = fraction of examples in sub-Training Set of class $\omega_i$

- Entropy reduction is Mutual Information between class label and feature value
- Stop when sub-Training Set zero entropy and assign surviving class

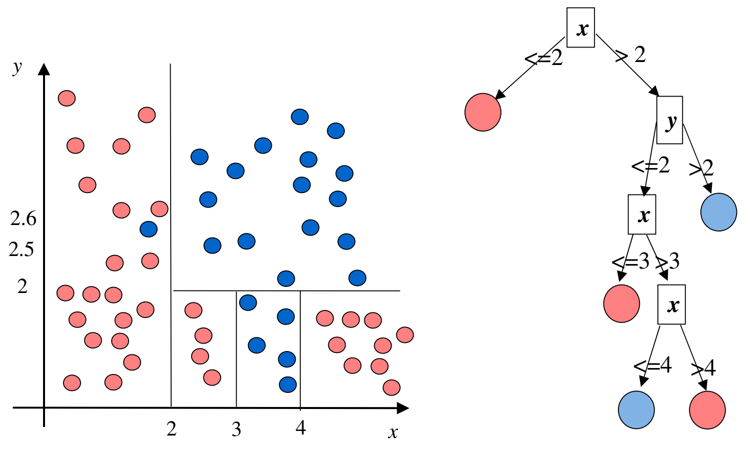
## Decision Trees/5: Post-Pruning



R1: $x \leq 2, y > 2.5, y > 2.6$
$\rightarrow o$

R2: $x \leq 2, y > 2.5, y \leq 2.6$
$\rightarrow o$

R3: $x \leq 2, y \leq 2.5$
$\rightarrow o$



Pruning:

R1: $x \leq 2, y > 2.6$
$\rightarrow o$

Further pruning:

R1: $x \leq 2 \rightarrow o$

Avoids overfitting: Usually performed on a hold-out set

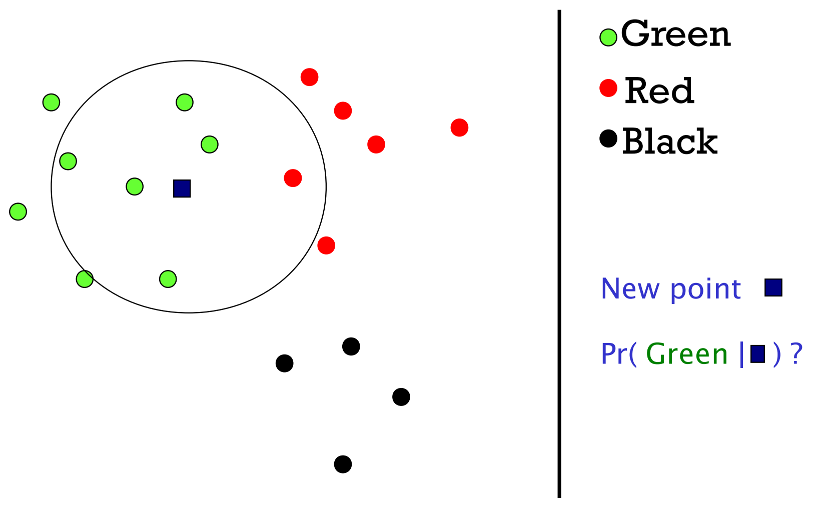
## $k$-Nearest Neighbor/1

- Unlike other (''eager'') methods $k$-NN does not build a model from Training Set (''lazy'')
- To classify pattern $\boldsymbol{x}$, determines $k$ closest points to $\boldsymbol{x}$ in Training Set
- Count no. $n_y$ of items in Training Set of class $y$ and estimate

  $$\Pr(y|\boldsymbol{x}) \simeq \frac{n_y}{k}$$

  (or weighted variants thereof)
- To classify go with the (weighted) majority
- Training not needed
- Classification time is linear in Training Set size if naively implemented and sublinear if specific data-structures are used (e.g. kd-trees on Voronoi tassellations)

## $k$-Nearest Neighbor/2



- distance function of paramount importance (and it heavily depends on scale factors), not easy to identify, especially for categorical features
- $k$ is hyper-parameter usually selected by cross-validation:
  - $k$ too big: aggregating also points far from $\boldsymbol{x}$
  - $k$ too small: too few aggregated points close to $\boldsymbol{x}$ to give rise to reliable majority

## $k$-Nearest Neighbor/3

- $k$-NN is simple non-parametric classifier (lacking other solutions . . . )
- Unlike Naive Bayes and Decision trees, $k-$NN is metric method (closeness in distance $\simeq$ semantic closeness)
- Being lazy, slow at test time
- Strongly dependent on distance function
- In the case of Euclidean distance

$$dist(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||^2 = (\boldsymbol{x} \cdot \boldsymbol{x}) + (\boldsymbol{x}' \cdot \boldsymbol{x}') - 2(\boldsymbol{x} \cdot \boldsymbol{x}')$$

classification depends on data only via scalar products

$\Longrightarrow$ can turn scalar products into more general products in (separable) Hilbert Spaces (more general representations to incorporate invariants, etc.)

## Aggregation methods/1

Improved accuracy by combining classifiers trained on same Training Set

$TrS = (\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_m, y_m) \in \mathbf{R}^d \times \{1 \dots c\}$

$h_1(\boldsymbol{x}) = h_{1,TrS}(\boldsymbol{x}) \in \{1 \dots c\}$
$h_2(\boldsymbol{x}) = h_{2,TrS}(\boldsymbol{x}) \in \{1 \dots c\}$
$\vdots$
$h_T(\boldsymbol{x}) = h_{T,TrS}(\boldsymbol{x}) \in \{1 \dots c\}$

$$h(\boldsymbol{x}) = \text{combination}(h_1, h_2, ..., h_T) \in \{1 \dots c\}$$

Requires diversity across classifiers

Diversity through:

- Different subsets of TrS for each $h_t$
- Different subset of features for each $h_t$
- Decorrelation of $h_t$ during training

## Aggregation methods/2: Bagging/1

| | | | | | | | | | Feature set |
|---|---|---|---|---|---|---|---|---|---|
| Original Tr.Set | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | a b c d e |
| Training Set 1 | 2 | 7 | 8 | 3 | 7 | 6 | 3 | 1 | b c e |
| Training Set 2 | 7 | 8 | 5 | 6 | 4 | 2 | 7 | 1 | a d e |
| Training Set 3 | 3 | 6 | 2 | 7 | 5 | 6 | 2 | 2 | a b |
| Training Set 4 | 4 | 5 | 1 | 4 | 6 | 4 | 3 | 8 | c d |

Bagging = Boostrap Aggregating: generates diversity via both different subsets of TrS and different subsets of features

Boostrap sample:
Given TrS, generates TrS' by sampling uniformly with replacement from TrS

```
**Input:** TrS, $T \in N$
**for** $t = 1$ **to** $T$ **do**
    • TrS$_t$ = boostrap sample of TrS
    • $F_t$ = random subset of features
    • $h_t$ = Learning-Algorithm(TrS$_t$, $F_t$)
**Output:** Flat Majority over $h_1, h_2, \ldots, h_T$
```

## Aggregation methods/2: Bagging/2

Useful when learner is unstable (small variations in TrS yield big changes in learned classifier):

- True e.g. for decision trees (random forests)
- Not true e.g. for Naive Bayes

Experimentally, bagging substantially increase performance of unstable methods, but can degrate performance of stable ones

Hyperparameters (usually by cross-validation):

- Size of boostrap samples $TrS_t$
- Size of subset of features $F_t$

## Aggregation methods/3: Boosting/1

Main intuition: adaptive combination of pool of classifiers forced to be different during training

- Training:
  - produces sequence of classifiers coming from simple base hypothesis set
  - Base hypothesis set: set of simple classifiers (e.g. linear or based on few features only)
  - each new classifier in sequence depends on previous one by focusing on mistakes of old one
  - training examples incorrectly classified is given higher weight
- Output:Weighted combination of classifiers so produced

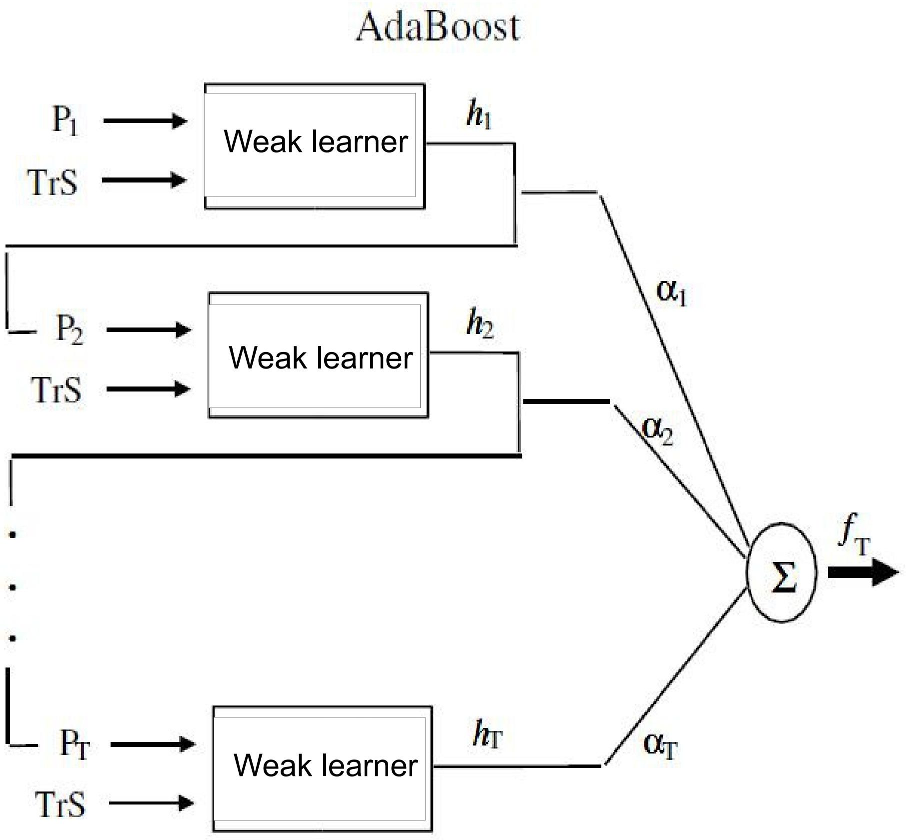
## Aggregation methods/4: Boosting/2

### Adaboost

Given distribution $P$ on TrS
$h_t \in$ base hypothesis set is
weak learner if accuracy
slightly better than random

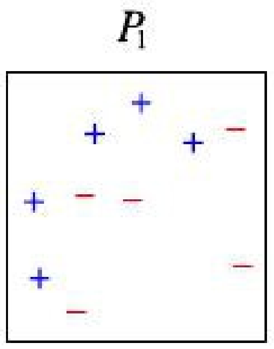
Error $\epsilon_t$ of $h_t$ on TrS w.r.t. $P_t$:
total $P_t$-weight of
misclassified examples $(\boldsymbol{x}_i, y_i)$

Weight $\alpha_t$ high
if error $\epsilon_t$ small



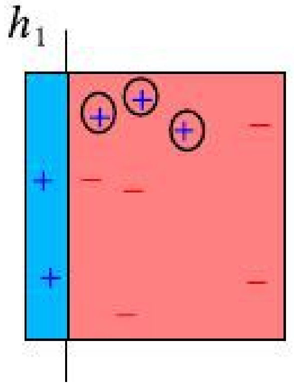
## Aggregation methods/4: Adaboost in action (binary)/1

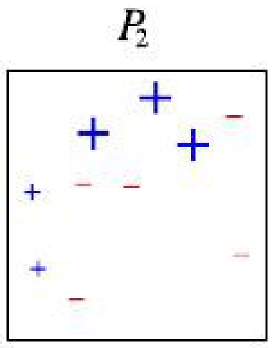
Init



Weak learners = horizontal or vertical half-planes

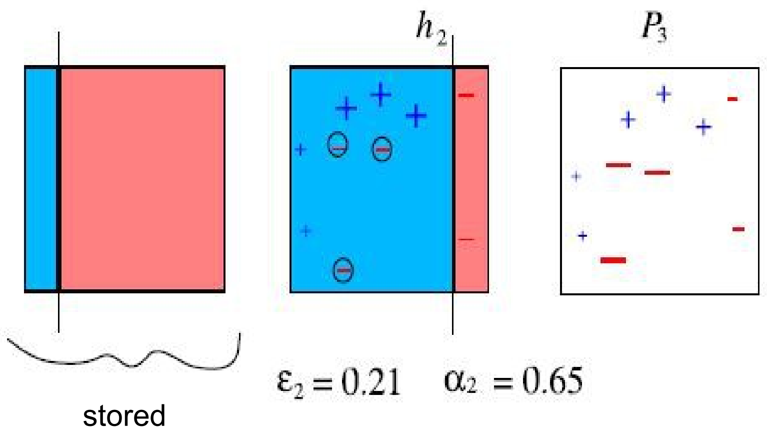
Iteration 1



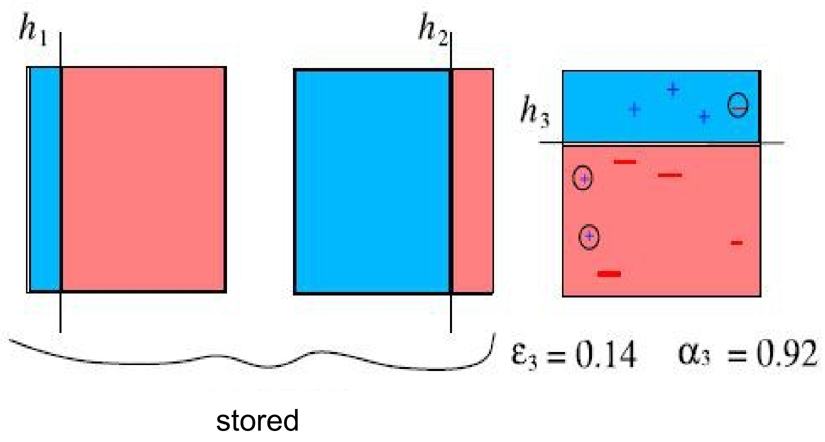


## Aggregation methods/4: Adaboost in action (binary)/2

Iteration 2

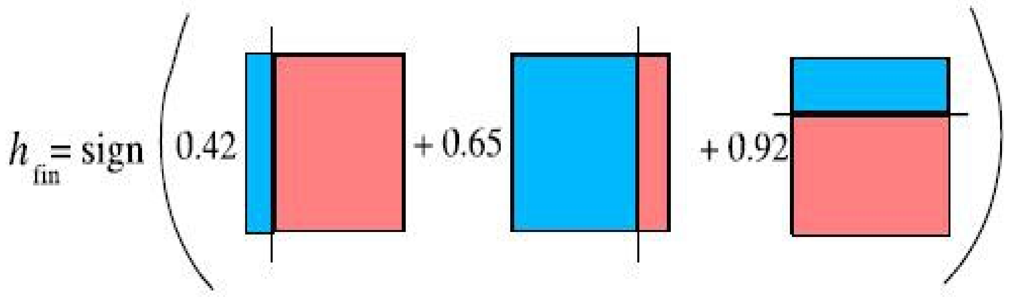


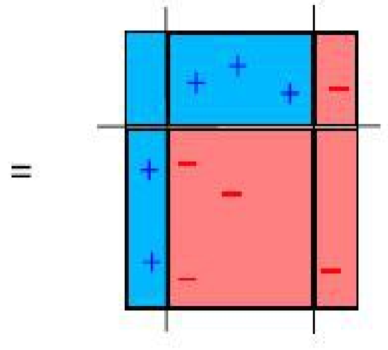
Iteration 3



## Aggregation methods/4: Adaboost in action (binary)/3

Final hypothesis (trained model):





## Aggregation methods/5: Adaboost

Adaboost pros:

- General and flexible meta-algorithm, works in practice with any reasonable weak learner
- Tends to outperform bagging in practice
- Only one hyperparameter ($T$ = no. of iterations)
- Easy to implement
- Theoretical underpinning

Adaboost cons:

- Unclear how to incorporate prior knowledge in effective manner
- Caution needed on noisy data (strong focus on mistake can degrage performance)
- Unclear how to select best base hypothesis set, needs unstable weak learners (like bagging)
- Not easy to parallelize (unlike bagging)

## Performance evaluation and workflow/1

- A dataset $D$ at our disposal
- Training set TrS (to learn a model $M = M(\text{TrS})$)
- Test set TeS (to test learned model $M$), sometimes called holdout set

$$\text{TrS} \cap \text{TeS} = \emptyset \qquad \text{TrS} \cup \text{TeS} = D$$

- Typical proportions 1/2 − 1/2 or 2/3 − 1/3
- Partitioning of $D$:
  - random sampling
  - determined by nature of data (e.g. TrS = past, TeS = future)
- (*) A modeling assumption: $M$ from class of models, data have specific representation
- Evaluation:

$$Accuracy = 1 - Error = \frac{\text{no. of correct classifications on TeS}}{|\text{TeS}|}$$

- If "unhappy" $\Longrightarrow$ iterate back to (*)

## Performance evaluation and workflow/2

However:

- Evaluation is multifaceted (conflicting needs: training time, test time, robustness to noise or missing values, . . . )
- Better solutions than holdout set (cross-validation)
- Accuracy need not be reasonable statistical criterion
- More sublte: cannot iterate too many times . . .
- Benchmarking is very important

## Performance evaluation and workflow/3

$n$-fold cross-validation (CV):

- $D$ partitioned into $n$ disjoint subsets Trs$_1$, . . . ,TrS$_n$ of same size
- Use every subset as TeS and remaining $n-1$ subsets as TrS
- Get $n$ models $M(TrS_1), \ldots, M(TrS_n)$ and $n$ accuracy values
- Average accuracy over model is estimated accuracy of $M$(TrS)
- 5-fold CV and 10-fold CV used in practice

More effective than hold out set since accuracy refers to $M(D)$, but also more demanding computationally

Leave-one out cross-validation: $n$-fold CV with $n = |D|$

CV typically used to estimate hyperparameters

## Performance evaluation and workflow/4

Accuracy inadequate when dealing with many classes or rare events

- Two unbalanced classes 1% and 99%: trivially achieve 99% accuracy by never predicting rare class
- One-vs-rest: class of interest is positive class, rest is negative class

## Performance evaluation and workflow/5

### Confusion matrix

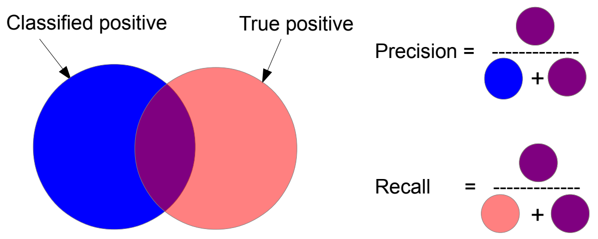
| | classified positive | classified negative |
|---|---|---|
| true positive | TP | FN |
| true negative | FP | TN |

- TP: no. of correct classifications among positive examples
- FN: no. incorrect classification among positive examples
- FP: no. incorrect classification among negative examples
- TN: no. of correct classifications among negative examples

$$\text{Precision } (P) = \frac{TP}{TP + FP} \qquad \text{Recall } (R) = \frac{TP}{TP + FN}$$

- Precision: fraction of correctly classified positive among the classified positive

- Recall: fraction of correctly classified positive among the true positive



## Performance evaluation and workflow/6

For instance

| | classified positive | classified negative |
|---|---|---|
| true positive | 1 | 99 |
| true negative | 0 | 1000 |

$$P = 100\% \qquad R = 1\%$$

- Precision and Recall only measure classification on positive class at hand
- Meaningless if taken in isolation:
  high $P$ achievable at low $R$ and vice versa

F-measure: harmonic mean of $P$ and $R$

$$F = \frac{1}{\frac{1}{2P} + \frac{1}{2R}} = \frac{2PR}{P+R}$$

Tends to be close to the smaller of the two: both $P$ and $R$ have to be large to ensure large $F$

A more refined one: Area under the (ROC) curve [not covered]

## Empirical risk minimization/1

Classification/regression tasks:

- Prediction models $h$ mapping feature space $\mathcal{X}$ to label space $\mathcal{Y}$
- $\mathcal{Y} = \{1, \ldots, c\}$ [classification], $\mathcal{Y} = \mathbf{R}$ [regression]
- Training set TrS $= (X_1, Y_1), \ldots, (X_T, Y_T)$
- Learning algorithm $A$ maps training set TrS to prediction model

$$A_{\mathsf{TrS}} : \mathcal{X} \to \mathcal{Y}$$

Goal: Evaluate risk of trained model $A = A_{\mathsf{TrS}}$ w.r.t. given loss function

$$\ell : \mathcal{Y} \times \hat{\mathcal{Y}} \to \mathbf{R}^+$$

viewing TrS as statistical sample

Statistical risk:

$$Risk_A = \mathbb{E}_{(X,Y)}\Big[\ell\Big(\underbrace{A_{\mathsf{TrS}}(X)}_{\text{predicted by trained model}}, \underbrace{Y}_{\text{value to be predicted}}\Big)\Big]$$

Training set TrS $= (X_1, Y_1), \ldots, (X_T, Y_T)$ and test sample $(X, Y)$ drawn i.i.d. from the same fixed but unknown distribution over $\mathcal{X} \times \mathcal{Y}$

## Empirical risk minimization/2: A landmark method

Compare to best in comparison class:

- Given class of models (or hypothesis class)

$$\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$$

  (e.g. class of linear functions)

- Want:

$$Risk_A \leq \inf_{h \in \mathcal{H}} \mathbb{E}_{(X,Y)}\Big[\ell\Big(h(X), Y\Big)\Big] + \dots$$

  with high probability (over TrS)

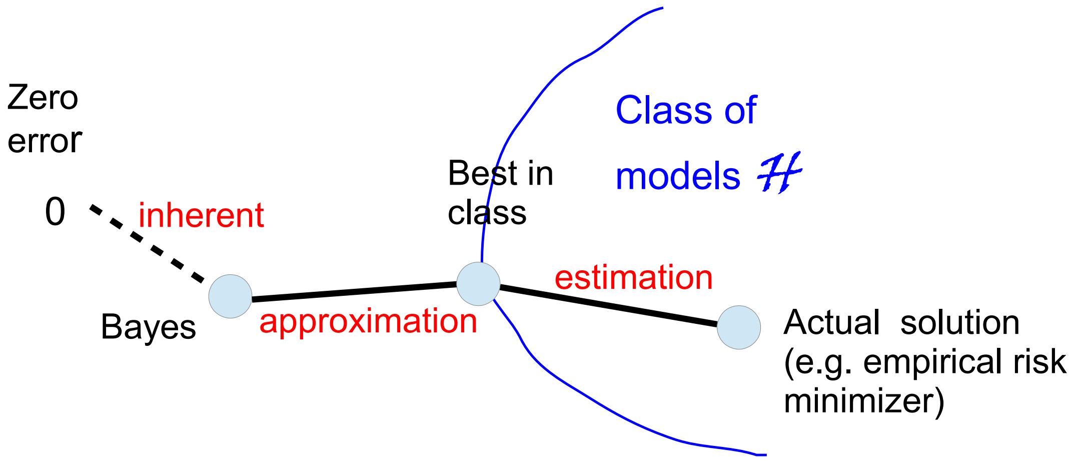
Empirical risk minimizer:

$$A_{\mathsf{TrS}} = \mathrm{argmin}_{h \in \mathcal{H}} \overbrace{\tfrac{1}{T} \sum_{t=1}^{T} \ell(h(X_t), Y_t)}^{\text{empirical risk}}$$

Regularized empirical risk minimizer:

$$A_{\mathsf{TrS}} = \mathrm{argmin}_{h \in \mathcal{H}} \overbrace{\tfrac{1}{T} \sum_{t=1}^{T} \ell(h(X_t), Y_t)}^{\text{empirical risk}} + \eta \overbrace{\text{Complexity}(h)}^{\text{regularizer}}$$

## Empirical risk minimization/3: Model selection


Zero error
0
inherent
Bayes
approximation
Best in class
Class of models $\mathcal{H}$
estimation
Actual solution (e.g. empirical risk minimizer)

- Inherent error: Inner indistinguishability in process generating data
- Approximation error: "Distance" of Bayes to best in class $\mathcal{H}$
- Estimation error: "Distance" of best in class to computed solution

Model selection: For given TrS, enlarging $\mathcal{H}$ gets reduced approximation error, but increased estimation error, and vice versa

## Empirical risk minimization/4: Computation

Empirical risk minimization (ERM):

- Generally hard to compute (local minima, NP hardness, . . .)
  Hardness heavily depends on model class $\mathcal{H}$ and loss function $\ell(\cdot, \cdot)$
- Many learning algs can be seen as approximate solutions to ERM
- Many of them are (stochastic) optimization algs (gradient descent-like)

## Empirical risk minimization/5: Examples/1

- $\mathcal{H}$ is a class of linear functions

$$\mathcal{H} = \{\boldsymbol{w} \in \mathbf{R}^d \; : \; \boldsymbol{x} \to \boldsymbol{w} \cdot \boldsymbol{x}\}$$

- Regularized ERM on TrS $= (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T) \in \mathbf{R}^d \times \mathbf{R}$:

$$\text{argmin}_{\boldsymbol{w} \in \mathcal{H}} \frac{1}{T} \sum_{t=1}^{T} \ell(y_t, \boldsymbol{w} \cdot \boldsymbol{x}_t) + \eta \, ||\boldsymbol{w}||^2$$

- $\ell(y, \widehat{y}) = \ell(y, \boldsymbol{w} \cdot \boldsymbol{x})$ (loss function in estimating $y$ by $\boldsymbol{w} \cdot \boldsymbol{x}$)

- $\ell(y, \widehat{y})$ convex in $\widehat{y}$

Stochastic Gradient Descent (SGD):

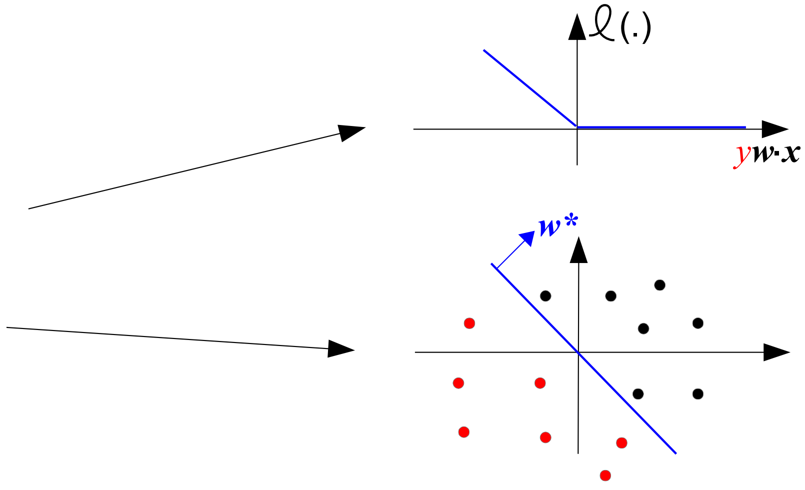
- Init: $\boldsymbol{w}_0$

- For $t = 1, 2, \ldots$
  - . Pick index $I_t \in \{1, \ldots, T\}$ at random
  - . $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} \ell(y_{I_t}, \boldsymbol{w} \cdot \boldsymbol{x}_{I_t})_{|\boldsymbol{w}=\boldsymbol{w}_t}$

## Empirical risk minimization/5: Examples/2

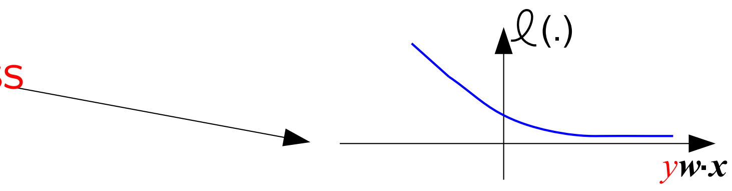
Perceptron algorithm:

- Labels $y$ are binary $\pm 1$
- $\ell(y, \boldsymbol{w} \cdot \boldsymbol{x}) = \max\{0, -y\boldsymbol{w} \cdot \boldsymbol{x}\}$ is hinge loss
- Data are linearly separable
  $(\exists \boldsymbol{w}^* \text{ s.t. } y_t \boldsymbol{w}^* \cdot \boldsymbol{x}_t > 0 \; \forall t)$



Logistic regression:

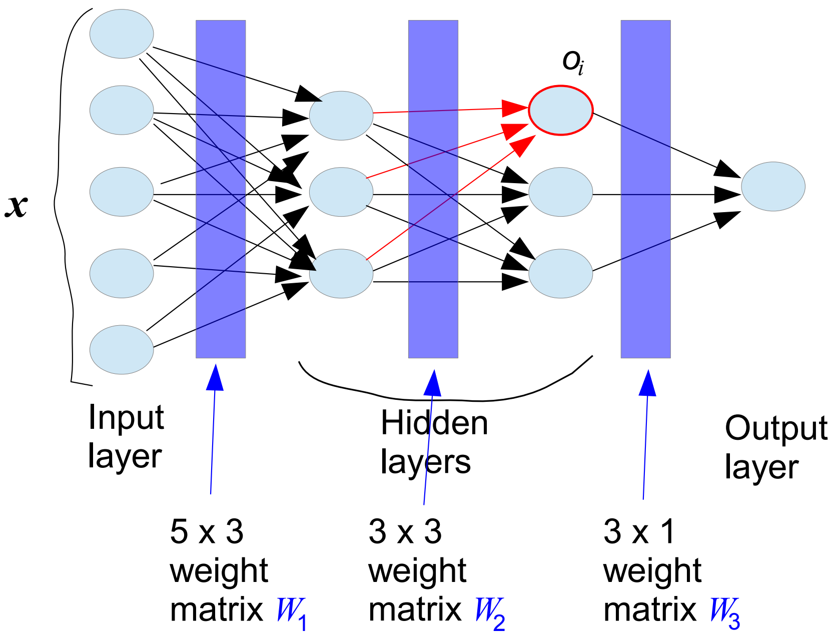
- Labels $y$ are binary $\pm 1$
- $\ell(y, \boldsymbol{w} \cdot \boldsymbol{x}) = \log(1 + \exp(y\boldsymbol{w} \cdot \boldsymbol{x}))$ is logistic loss
- $\Pr(y = 1 \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}^* \cdot \boldsymbol{x})}{\exp(\boldsymbol{w}^* \cdot \boldsymbol{x}) + 1}$
  logistic model: ERM = maximum likelihood estimator



## Empirical risk minimization/5: Examples/3

$\mathcal{H}$ is class of multilayer feedforward networks

$$\mathcal{H} = \{\boldsymbol{x} \in \mathbf{R}^d \rightarrow N(\boldsymbol{x}, W_1, W_2, \ldots, W_k)\}$$



$o_i$
$\boldsymbol{x}$
Input layer
Hidden layers
Output layer
5 x 3 weight matrix $W_1$
3 x 3 weight matrix $W_2$
3 x 1 weight matrix $W_3$

5-3-3-1 feed-forward architecture:

connections are to next layer only

$o_i = f(\sum_{j:j\rightarrow i} w_{j,i} o_j)$

$f$ is (non-linear) activation function

E.g.: $f$ sigmoid or ReLU

Want to learn parameters $W_1, W_2, \ldots, W_k$ via SGD-like alg.

## Empirical risk minimization/5: Examples/4

Training error measured by quadratic criterion (ERM with square loss):

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - N(\boldsymbol{x}_t, W_1, W_2, \ldots, W_k))^2$$

Minimize via SGD over set of weights $W_1, W_2, \ldots, W_k$

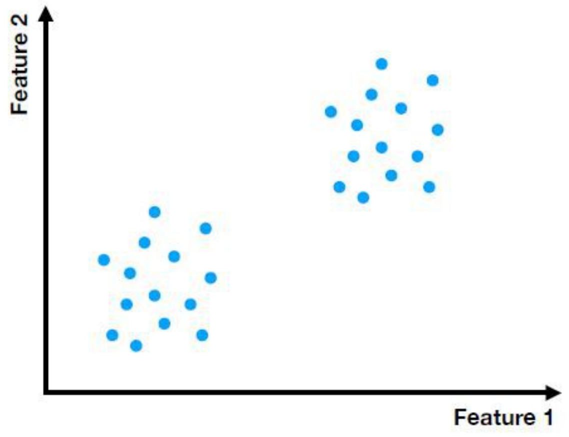
Issue: local minima, but lot of recent literature on ReLU showing these minima are not "bad" (properly tuned SGD can escape them)

Back-propagation:

- Standard algorithm performing (stochastic) (sub-)gradient descent
- Uses chain rule of differentials to propagate updates forward and then backward
- Main reason for its success is fast implementation

## Clustering/1

Grouping objects: inferring similarities, compression, etc.



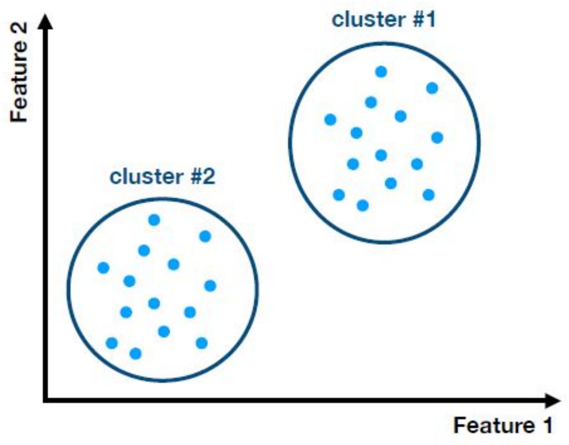
From: D. Baron, “clustering algorithms”, Nov. 2018

Different clustering tasks:

- Flat clustering vs. hierarchical clustering
- Unsupervised clustering vs. supervised clustering

## Clustering/1

Grouping objects: inferring similarities, compression, outlier detection, density estimation, etc.



Different clustering tasks:

- Flat clustering vs. hierarchical clustering
- Unsupervised clustering vs. supervised clustering

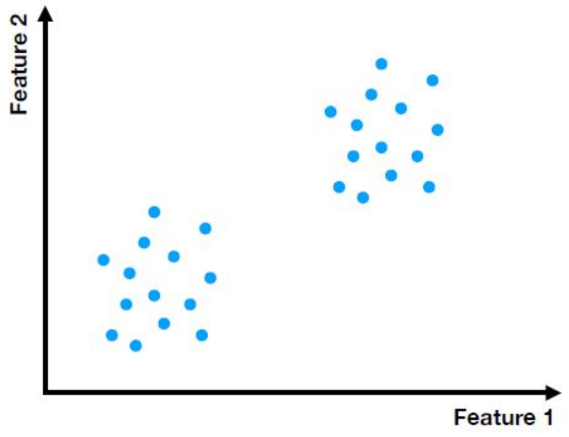
## Clustering/2: $k$ Means/1

Flat unsupervised clustering

Given

- Items $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \in \mathbf{R}^d$
- desired no. of clusters $k$

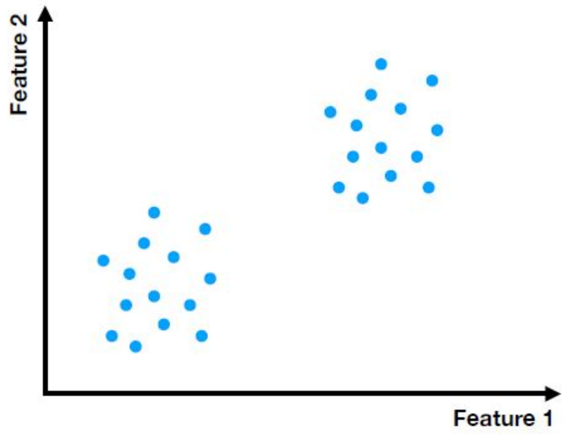
Cluster the $T$ items into $k$ clusters (w.r.t. Euclidean distance)



## Clustering/2: $k$ Means/2

$k$ Means alg. [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

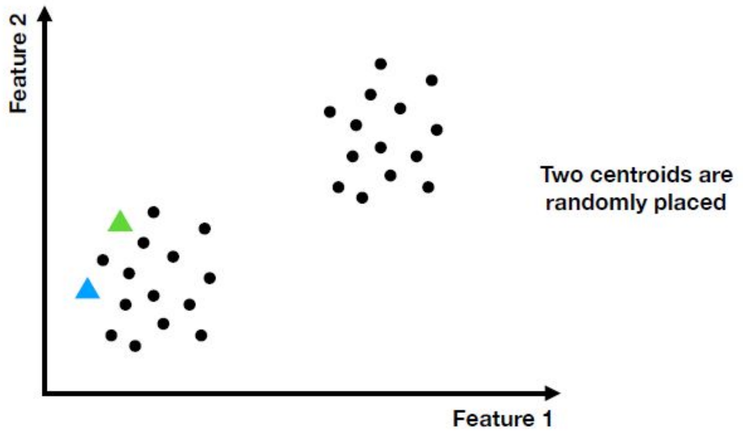
- Init: Selects randomly $k$ points representing the cluster centroids
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it



## Clustering/2: $k$ Means/2

$k$ Means alg: [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

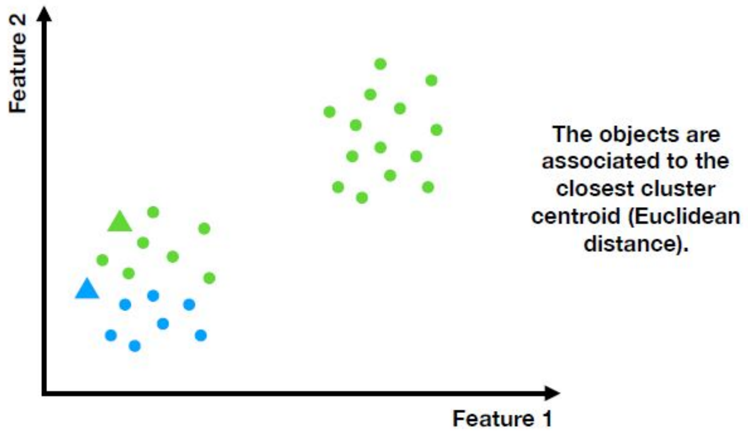
- Init: Selects randomly $k$ points representing the cluster centroids
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it



## Clustering/2: $k$ Means/2

$k$ Means alg: [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

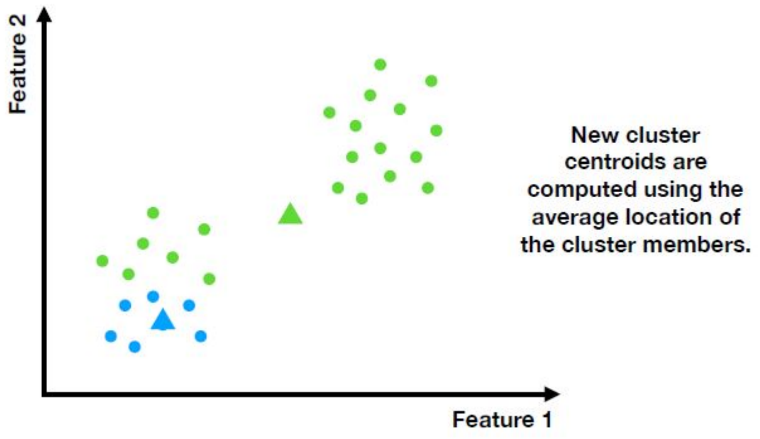
- Init: Selects randomly $k$ points representing the cluster centroids
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it



## Clustering/2: $k$ Means/2

$k$ Means alg: [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

- Init: Selects randomly $k$ points representing the cluster centroids
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it

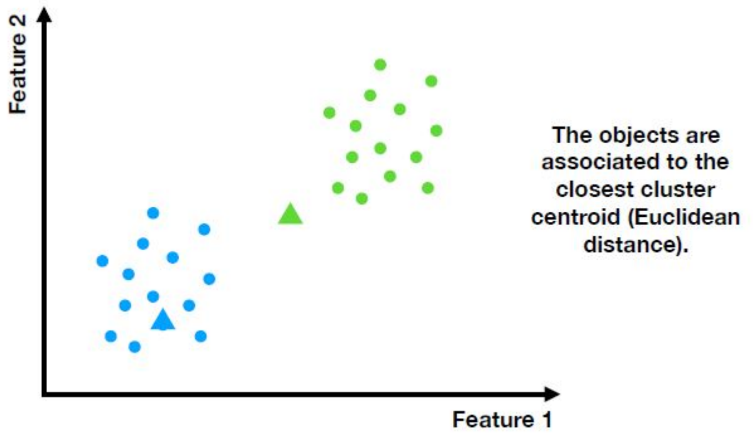

Feature 2
Feature 1
New cluster centroids are computed using the average location of the cluster members.

## Clustering/2: $k$ Means/2

$k$ Means alg: [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

- Init: Selects randomly $k$ points representing the cluster centroids

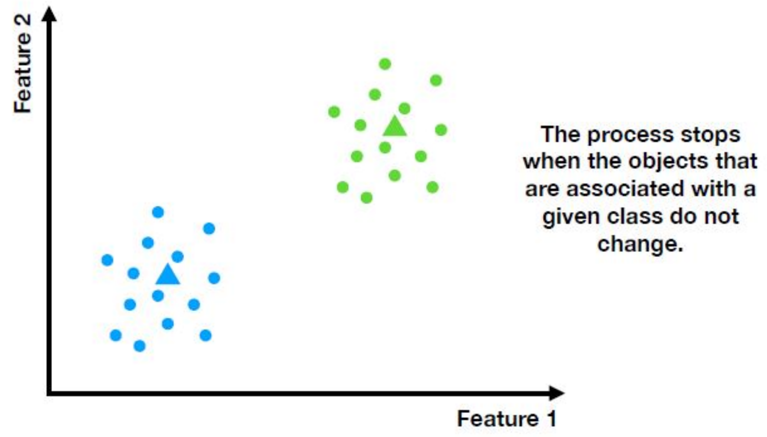
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it


Feature 2
Feature 1
The objects are associated to the closest cluster centroid (Euclidean distance).

## Clustering/2: $k$ Means/2

$k$ Means alg: [ slides from D. Baron, clustering algorithms, Nov. 2018 ]

- Init: Selects randomly $k$ points representing the cluster centroids
- Iterate:
  - Alg. associates each item with single cluster based on distance from cluster centroid
  - Alg. re-computes cluster centroids based on items currently associated with it


Feature 2
The process stops when the objects that are associated with a given class do not change.
Feature 1

## Clustering/2: $k$ Means/3

$k$ Means criterion:
Find partition $C_1, \ldots, C_k$ of $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ s.t.

$$f_k = \sum_{i=1}^{k} \sum_{j \in C_i} || \underbrace{\boldsymbol{x}_j}_{j\text{-th item}} - \underbrace{\mu_i}_{\text{centroid of } C_i} ||^2$$

is mimimized

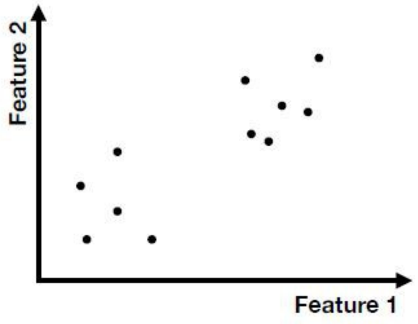
- NP-hard for given $k$ (but trivial when $k = T$)
- Hyperparamenter $k$ has to be selected based on exogenous information
  Common pratice: identify elbow of $f_k$ as a function of $k$
- Approximation algorithms (e.g., $k$ Means++) exist based on smarter init step
- Other methods exist (e.g, Gaussian Mixture + EM)

## Clustering/3: Hierarchical clustering

Given

- Items $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \in \mathbf{R}^d$
- distance function $dist(\boldsymbol{x}_1, \boldsymbol{x}_2)$ over items
- distance function over sets of items (linkage function)

Build binary tree (dendrogram) whose leaves are the items, each cut of the tree being a flat clustering of the $T$ items

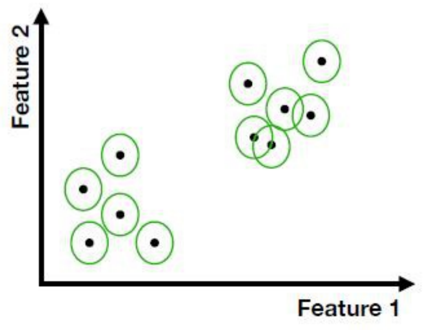


From: D. Baron, “clustering algorithms”, Nov. 2018

## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

Init: one cluster per item (bottom up or agglomerative)

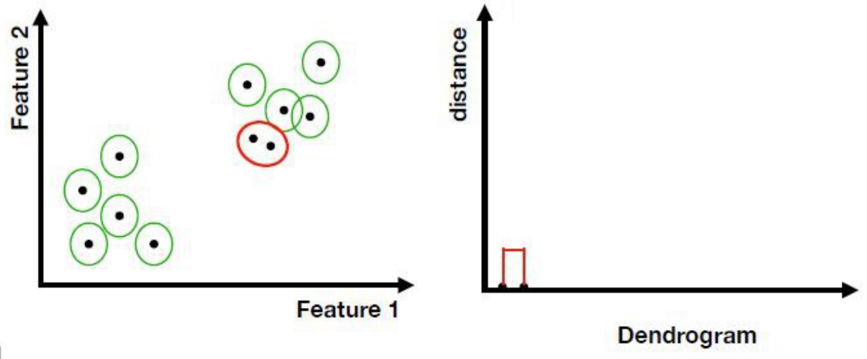


From: D. Baron, “clustering algorithms”, Nov. 2018

## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

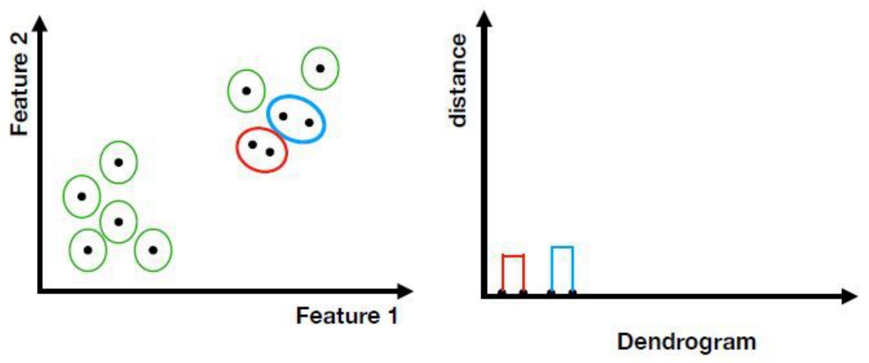
Merge the two closest points and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

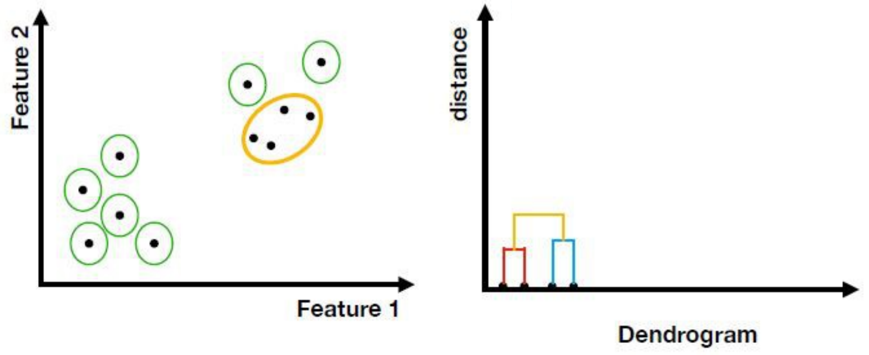
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

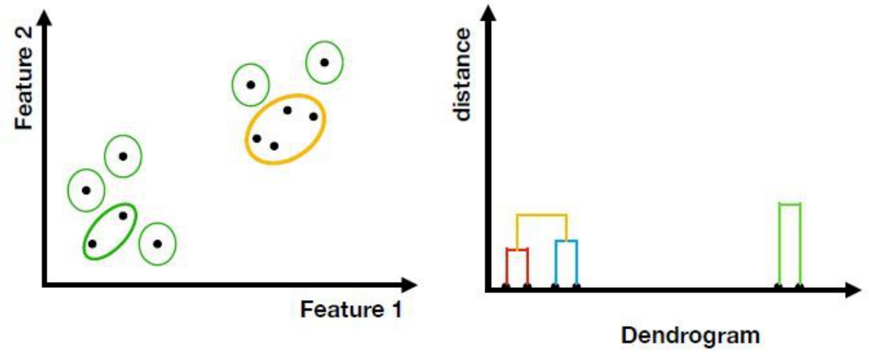
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

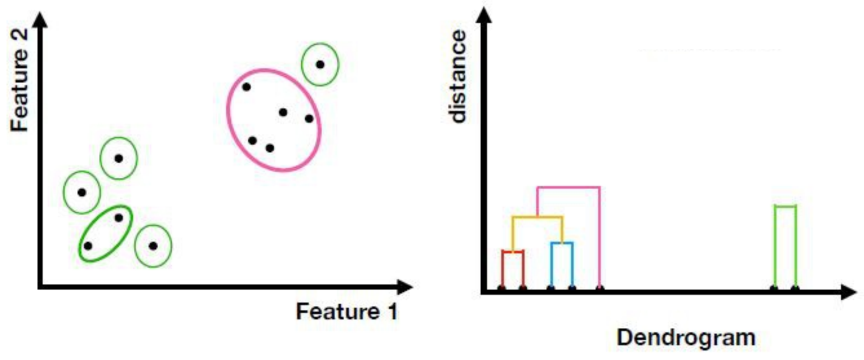
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

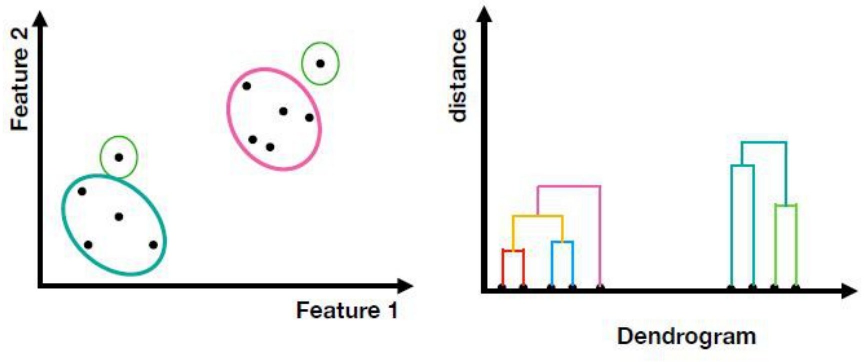
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

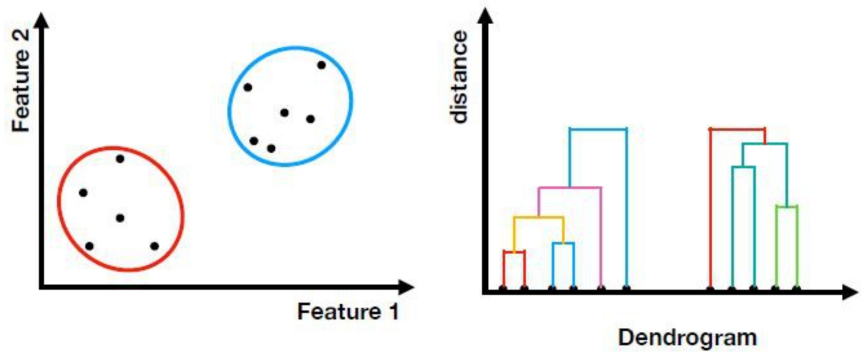
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

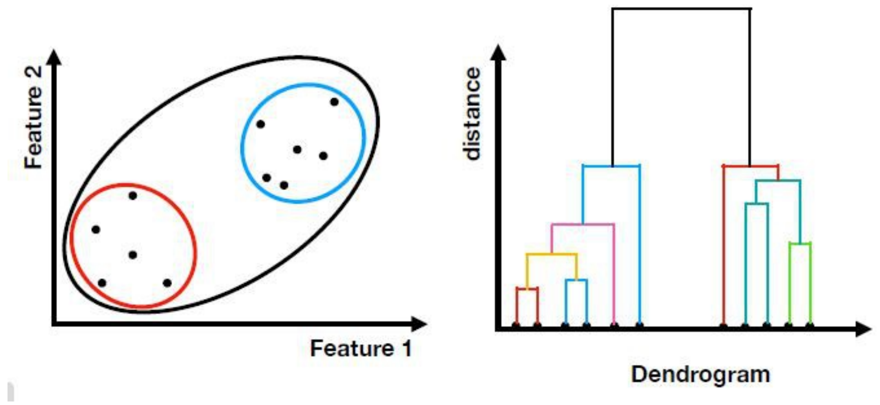
Merge the two closest points/clusters and iterate



## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

Merge the two closest points/clusters and iterate

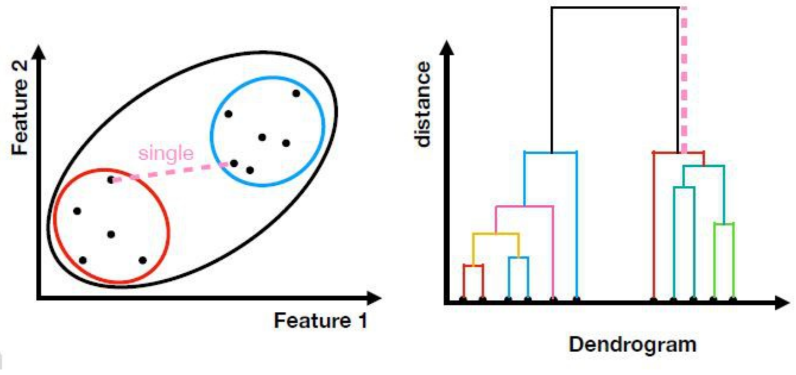


## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

Single linkage:

$$dist(C_1, C_2) = \min_{\boldsymbol{x}_1 \in C_1, \boldsymbol{x}_2 \in C_2} dist(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

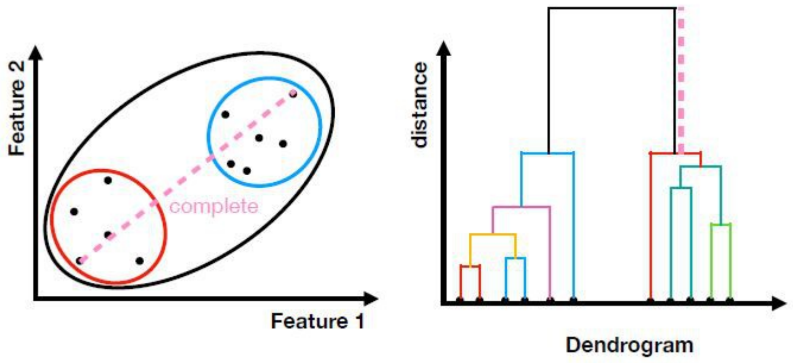


## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

Complete linkage:

$$dist(C_1, C_2) = \max_{\boldsymbol{x}_1 \in C_1, \boldsymbol{x}_2 \in C_2} dist(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

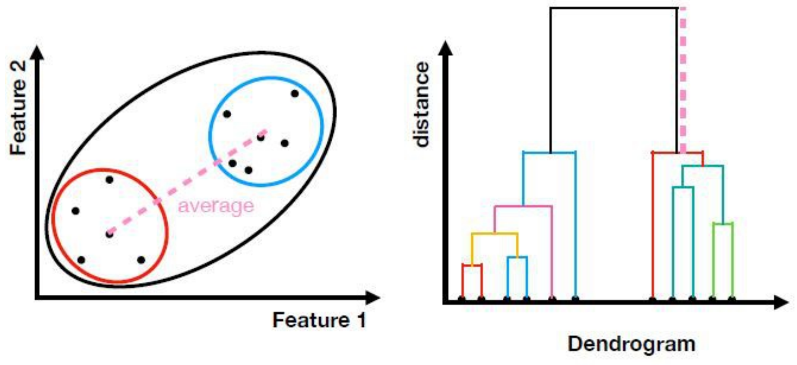


## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ][-10mm]

Average linkage:

$$dist(C_1, C_2) = dist(\mu_1, \mu_2)$$

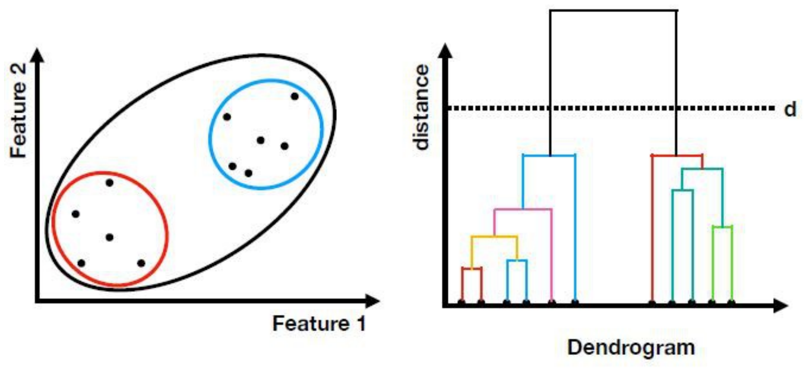


A few more: median linkage, quantile linkage (more robust to outliers)

## Clustering/3: Hierarchical clustering

[ slides from D. Baron, clustering algorithms, Nov. 2018 ]

Hyperparameter: Flat clustering determined below given distance threshold or given target no. $k$ of clusters



## Clustering/4: Supervised clustering

Clustering with extra source of information:

must link and/or cannot link constraints

Many ad hoc algorithms exist:

- variants of $k$ Means
- or not . . .

## Conclusions/1

- Machine Learning has very complex landscape of
  - Problems: classification, regression, ranking, clustering, dimensionality reduction, Sequential and Reinforcement Learning, . . .
  - Methods: $k$-NN, Linear Discriminant Analysis, Decision Trees, Stochastic Gradient Descent, Support Vector Machines, Boosting, $k$-means, Variational Autoencoders, Deep Networks, Generative Adversarial Networks, Gaussian Process methods, . . .
  - Tools: Weka, Tensorflow, Pytorch, Keras, Caffe, . . .
- Wide interaction with other fields (Statistical Inference, Control Theory, Signal processing, Complex Networks, . . .)

## Conclusions/2

Ask yourself: "What is the goal of my investigation ?"

- Just prediction accuracy vs. higher-level knowledge
- Validating a hypothesis ?
- Is scalability an issue ?

Inspecting data first is of paramount importance (don't be lazy . . .)

- How to incorporate prior knowledge (e.g. invariances)
- Preprocess data: normalize, re-center, augment, cluster data
- Pay attention to imbalanced, incomplete, noisy data
- Visualization techniques at this stage often important
- Carry out correct benchmarking: pay attention to bias in both data and experimental practice

Claim: Better you know what you are using out of a software library . . .

## Conclusions/3

Some textbooks:

- C. Bishop, Pattern Recognition and Machine Learning. Cambridge, UK: Cambridge University Press, 2007.
- Duda, Hart, Stork, Pattern Classification, 2nd Ed., 2001.
- David J.C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
- S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.
- I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Cambridge MA: MIT Press, 2016.