Machine Learning Summary

Supervised

Types of models :

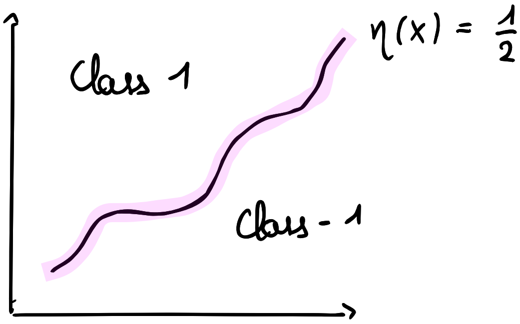
- discriminant : estimate decision frontier directly
  $\mathbb{P}(Y=1 \mid X=x)$
- generative : estimate $\mathbb{P}(X=x \mid Y=1)$ and $\mathbb{P}(X=x \mid Y=-1)$

If we know the decision frontier $\eta(x) = \mathbb{P}(Y=1 \mid X=x)$, there is no need for simulations. This is the Bayes classifier:

$$g(x) = \begin{cases} +1 & \text{if } \eta(x) \geq \frac{1}{2} \\ -1 & \text{if } \eta(x) < \frac{1}{2} \end{cases}$$

$$\Leftrightarrow \boxed{g(x) = 2\,\mathbb{1}\{\eta(x) \geq \tfrac{1}{2}\} - 1}$$

However, this classifier can never be reached.



Usually, we try to approach the Bayes classifier using the so-called plug-in estimator:

↳ · Estimate $\eta(x)$

· Plug-it in $\hat{g}(x) = 2\,\mathbb{1}\{\hat{\eta}(x) \geq \frac{1}{2}\} - 1$

I. Generative models

1. Linear discriminant analysis (LDA)

Let : $\cdot\ G = \mathcal{L}(X \mid Y=1)$
$\quad\ \ H = \mathcal{L}(X \mid Y=-1)$ } distribution of the 2 classes

Define a likelihood ratio $\Phi(X) = \frac{dG}{dH}(x) = \frac{\mathbb{P}(X=x \mid Y=+1)}{\mathbb{P}(X=x \mid Y=-1)}$

$$= \frac{\mathbb{P}(Y=1 \mid X=x) \cdot \cancel{\mathbb{P}(X=x)}/\mathbb{P}(Y=1)}{\mathbb{P}(Y=-1 \mid X=x) \cdot \cancel{\mathbb{P}(X=x)}/\mathbb{P}(Y=-1)} = \frac{1-p}{p} \cdot \frac{\eta(x)}{1-\eta(x)}$$

$$\hookrightarrow \frac{p\,\Phi(x)}{(1-p)+p\,\Phi(x)} = \eta(x)$$

Gaussian underlying hypothesis :

$$\left.\begin{aligned} G &= N_A(\mu_+, \Gamma) \\ H &= N_B(\mu_-, \Gamma) \end{aligned}\right\} \quad N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

If $\eta(x) \geq \frac{1}{2} \Rightarrow \Phi(x) \geq \frac{1-p}{p}$

$$\hookrightarrow {}^tx\Gamma^{-1}(\mu_+ - \mu_-) + \frac{1}{2}\left({}^t\mu_- \Gamma^{-1}\mu_- - {}^t\mu_+\Gamma^{-1}\mu_+\right) \geq \log\frac{p}{1-p}$$

$$\Leftrightarrow \alpha + {}^t\beta x \geq 0$$

where : $\cdot\ \beta = \Gamma^{-1}(\mu_+ - \mu_-)$

$\cdot\ \alpha = \frac{1}{2}\left({}^t\mu_-\Gamma^{-1}\mu_- - {}^t\mu_+\Gamma^{-1}\mu_+\right) - \log\left(\frac{p}{1-p}\right)$

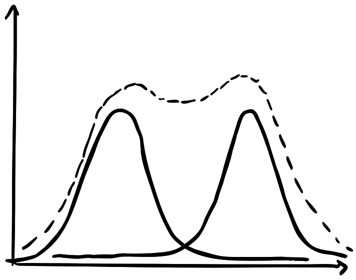
$$\hookrightarrow \hat{p} = \frac{n_+}{n}, \quad \hat{\mu}_+ = \frac{1}{n_+}\sum_{\substack{i=1 \\ \{Y_i=+1\}}}^{n} X_i, \quad \hat{\mu}_- = \frac{1}{n_-}\sum_{\substack{i=1 \\ \{Y_i=-1\}}}^{n} X_i$$

2. Quadratic discriminant analysis (QDA).

Relax the Gaussian hypothesis $\Rightarrow$ Mixture of gaussians.

$\gamma F_1 + (1-\gamma) F_2$

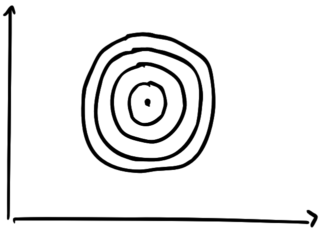
G and H can be bi-modal.



3. Naive Bayes classifier

We now suppose independence among our variables:

$\Gamma = \text{diag}_{1 \leq i \leq d} \; \sigma_i^2$

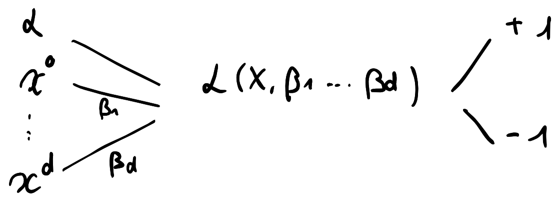


II. Discriminant models

1. Perceptron

$g(x) = \text{sign}(\alpha + {}^t\beta x)$

$\min_{\alpha, \beta} \hat{L}_n(g)$, $\hat{L}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{-(\alpha + {}^t\beta x) y > 0\}$



Due to lack of derivability of "sign" function :

↳ Apply stochastic gradient descent → Stops evolving at some point.

Limitation : • Data need to be linearly separable

• Does not care about margin of the hyperplane.

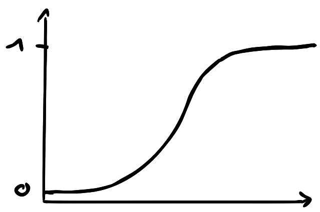
2. Logistic regression

• Regression : $\eta(x) = \mathbb{E}(Y \mid X) = \alpha + {}^t\beta x$

• Classification : $\eta(x) = \dfrac{e^{\alpha + {}^t\beta x}}{1 + e^{\alpha + {}^t\beta x}} = \dfrac{1}{1 + e^{-(\alpha + {}^t\beta x)}}$

Find $\hat{\alpha}$ and $\hat{\beta}$ by conditional maximum log-likelihood, and plug-in:

$$(\hat{\alpha}, \hat{\beta}) \Rightarrow \hat{\eta}_{\hat{\alpha},\hat{\beta}}(x) \Rightarrow \hat{g}(x) = 2\,\mathbb{1}\{\hat{\eta}_{\hat{\alpha},\hat{\beta}}(x) \geq \tfrac{1}{2}\} - 1$$



$$= 2\,\mathbb{1}\left\{ \frac{e^{\hat{\alpha} + {}^t\hat{\beta} x}}{1 + e^{\hat{\alpha} + {}^t\hat{\beta} x}} \geq \frac{1}{2} \right\} - 1$$

$$= 2\,\mathbb{1}\{\hat{\alpha} + {}^t\hat{\beta} \geq 0\} - 1$$

3. K-Nearest Neighbors

Let D be a distance metric (e.g Euclidean) and σ_x classifies points.

$D(x, x_{\sigma_x(1)}) \leq D(x, x_{\sigma_x(2)}) \leq \dots$

↳ Only keep K first neighbors.

$$g_{KNN}(x) = g_{KNN}(x, Y_{\sigma_x(1)}, \dots, Y_{\sigma_x(k)})$$

$$= \begin{cases} +1 & \text{if } \sum_{i=1}^{k} \mathbb{1}\{Y_{\sigma_x(i)}\} > \frac{k}{2} \\ -1 & \text{if } \sum_{i=1}^{k} \mathbb{1}\{Y_{\sigma_x(i)}\} < \frac{k}{2} \end{cases}$$

$$= 2 \cdot \mathbb{1}\{\textstyle\sum_{i=1}^{k} Y_{\sigma_x(i)} \geq 0\} - 1$$

Limitations : • Sensitive to metric D
• Sensitive to choice of K
• Computationally expensive
• Does not scale

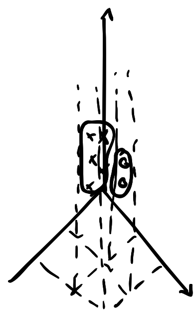
Extreme cases : • $k = 1 \rightarrow$ Training error is 0
• $k = n \rightarrow$ Training error is $\min \{ \frac{n_+}{n}, \frac{n_-}{n} \}$

4. Local averaging

• Divide feature space in regions $C_1 \cup \dots \cup C_K$
• $\hat{\eta}(x) = \sum_{k=1}^{K} \mathbb{1}\{x \in C_k\} \dfrac{\sum_{i=1}^{n} \mathbb{1}\{Y_i = 1, X_i \in C_k\}}{\sum_{i=1}^{n} \mathbb{1}\{X_i \in C_k\}}$ Majority rule.

↳ Nadaraya-Watson

• $\hat{g}(x) = 2\, \mathbb{1}\{\hat{\eta}(x) \geq \frac{1}{2}\} - 1$



Limitations : Depends highly on the regions we consider

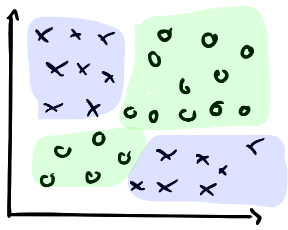
5. Decision tree

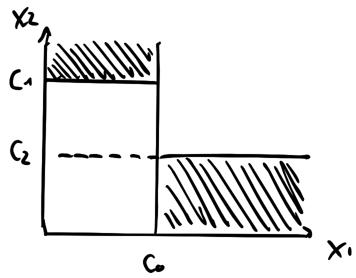
Build $g(x) = \sum_{\ell=1}^{m} \mathbb{1}\{x \in C_\ell\} \left(\operatorname{argmax}_c \left[\sum_{i, x \in C_\ell} \mathbb{1}\{Y_i = c\}\right]\right)$

where $C_1 \dots C_m$ automatic partitions

• Partition rule should minimize the local criteria
1) Crossed entropy : $H(S) = - \sum_{\ell=1}^{c} p_\ell(S) \log p_\ell(S)$
2) Gini index : $H(S) = \sum_{\ell=1}^{c} p_\ell(S)(1 - p_\ell(S))$
3) Classification error : $H(S) = 1 - p_c(S)$

where c # classes and p proportion of well classified data for each point : $p_c(S) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{Y_i = c\}$

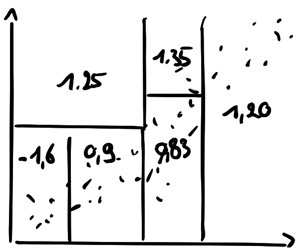




Advantage: • Interpretable, consistent, multiclass
• No pre-processing

Limitations: • Large variance, unstable
• No global optimization.

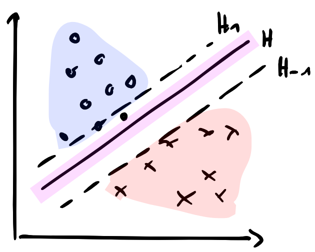
Can also be applied for regression trees:



$$L(t_j, \tau, S) = \frac{n_d}{n} \, Var_{emp}(D(j,\tau,s)) + \frac{n_g}{n} \, Var_{emp}(G(j,\tau,s))$$

where $Var_{emp}(s) = \frac{1}{|S|} \sum_{x, x_i \in S} (y_i - \bar{y})^2$

6. Support Vector Machine



a. Linear SVM

• **Separable case**

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad s.t \quad 1 - y_i(w^T x_i + b) \leq 0 \quad , i = 1 \ldots n$$

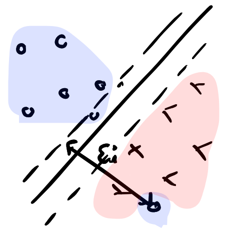
$$\min_{w,b,\alpha} \Rightarrow \mathcal{L}(w,b,\alpha) = \frac{1}{2} \|w\|^2 + \sum_i \alpha_i (1 - y_i(w^T x_i + b)) \quad , \alpha_i \geq 0$$

Find w, b and α by quadratic solver

↳ $g(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i x_i^T x + b\right)$

- Non-separable case

Define slack variables as miss-classified : ε_i



↳ soft-margin

$$\min_{w,b,\varepsilon} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \varepsilon_i$$

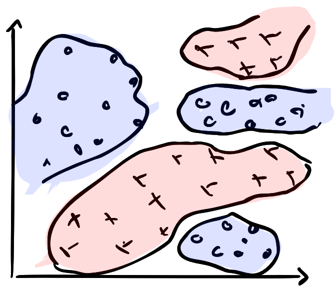
$$\text{s.t } y_i (w^T x_i + b) \geq 1 - \varepsilon_i \ , \ i = 1 \dots n$$

$$\mathcal{L}(w, b, \varepsilon, \alpha, \mu) = \frac{1}{2} w^T w + C \sum \varepsilon_i + \sum_{i=1}^{n} \alpha_i (1 - \varepsilon_i - y_i (w^T x_i + b)) - \sum \mu_i \varepsilon_i$$

C regulates the importance of misclassified observations.

If :
- $\alpha_i^* = 0$ → No error, does not influence hyperplane
- $0 < \alpha_i^* < C$ → On the frontier
- $\alpha_i^* = C$ → x_i^* is misclassified and is support

b. Non-linear SVM



Using Kernel trick :

$$g(x) = \text{sign} \left(\sum_i \alpha_i y_i \underbrace{k(x_i, x)}_{\text{instead of } x_i^T x} + b \right)$$

Idea : Transform data through non-linear function ϕ

Thanks to Moore-Aronszajn, guarantee that :
- $\exists$ Hilbert space : feature space
- $\phi : \mathcal{X} \to \mathcal{F}$: feature map

s.t $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = k(x, x')$

$$\min_{w,b,\varepsilon} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \varepsilon_i \quad \text{s.t. } y_i (w^T \phi(x_i) + b) \geq 1 - \varepsilon_i \quad i = 1 \dots n$$
$$\varepsilon_i \geq 0$$

↳ $g(x) = \text{sign} (\sum_i \alpha_i y_i \phi(x_i^T) \phi(x_i') + b)$

$= \text{sign} (\sum_i \alpha_i y_i k(x_i, x_i') + b)$

What kernels can we use?

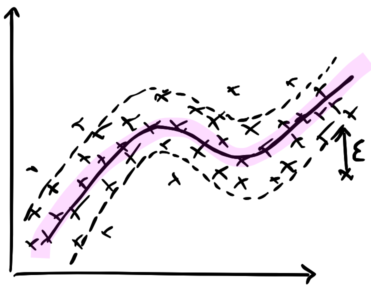
- Linear : $k(x, x') = x^T x'$
- polynomial : $k(x, x') = (x^T x' + c)^d$
- gaussian : $k(x, x') = \exp(-\gamma \| x - x' \|^2)$ (RBF kernels)

Why do we use kernels?

- work in spaces of infinite dimensions
- gaussian kernels allow approximation of most problems
- there is an algebra for kernels

Limitations :
- Find parameter C and γ for kernel
- Multiclass does not really work → several binary SVMs instead.

C. Support Vector Regressor



Define an ε-tube.

$$\min_{w,b,\xi} \frac{1}{2} \| w \|^2 + C \sum_i (\xi_i + \xi_i^*)$$

$$\text{s.t } \; y_i - f(x_i) \leq \epsilon + \xi_i$$
$$f(x_i) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i^* \geq 0$$

where in the general case : $f(x) = w^T \phi(x) + b$

Limitations :
- Quadratic programm computationally expensive
- Does not really scale

7. Bagging

Comes from Bootstrap Aggregating.

- Draw T training independent samples $\{S_1 \dots S_T\}$
- Learn a model $f_t \in \mathcal{F}$ from each sample S_t
- Compute the average model $f_{ens}(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x)$

Why use bagging ?

- Bias remains the same : $\mathbb{E}_{S_1 \dots S_T}[f_{ens}(x)] = \mathbb{E}_S[f_S(x)]$
- Variance divided by T : $\mathbb{E}_{S_1 \dots S_T}[(f_{ens}(x) - \mathbb{E}_{S_1 \dots S_T}[f_{ens}(x)])^2]$

$$= \frac{1}{T} \mathbb{E}_S[(f_S(x) - \mathbb{E}_S[f_S(x)])^2]$$

- Useful if unstable learning algorithm such as trees.

Limitations :
- Obtained model is complex
- Outperformed by random forest and gradient boosting.

8. Random Forest

We have a train set S_{train} and p features.

- Create T samples (bootstrap) $\{S_1 \dots S_T\}$
- For $t = 1 \dots T$:
  - Select f features s.t. $f \ll p$ randomly without replacement.
  - $h_{tree}^{(t)}$ decision tree learned from S_t with best split of features.
- $H^T = \frac{1}{T} \sum_t h_{tree}^{(t)}$

9. Extra Trees

We won't use bootstrap here, but S_{train} each time.

- For $t = 1 \dots T$:
  - Always use S_{train}
  - Select f features randomly without replacement ⇒ Split (S, i)
    - Let a^i_{max} and a^i_{min} denote maximal value of x_i in S
    - Draw uniformely a cut-point a_c in $[a^i_{max}, a^i_{min}]$

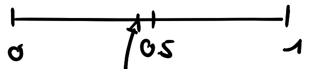
    ↳ Choose the best split
  - $h_{tree}^{(t)}$ randomized tree learned from S_{train}.
- $H^T = \frac{1}{T} h_{tree}^{(t)}$.

Advantage : Fast, parallelizable, easy to tune

Limitations :
- Overfitting if large tree
- Lost interpretability

10. Boosting

Let h be a binary weak classifier $\Rightarrow [-1, +1]$

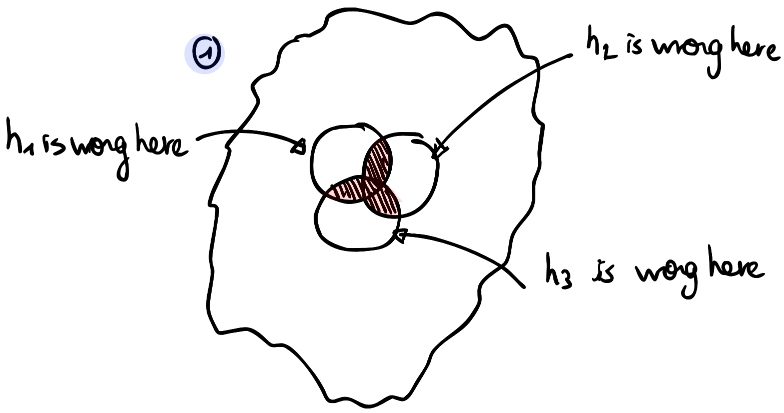


weak classifier.

• Adaboost.

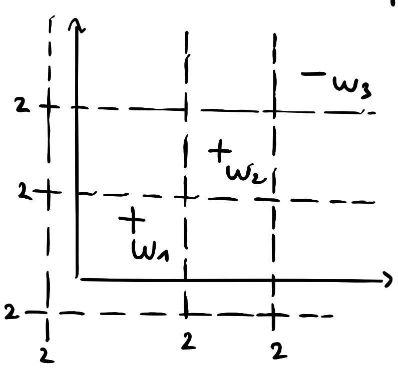
Boosting is a way to recursively combine classifiers that are weak learners.
It is based on several key concepts:

①



- Data $\longrightarrow$ h^1
- Exagerated data on h^1 errors $\longrightarrow$ h^2
- Exagerated data on $h^1 \neq h^2$ $\longrightarrow$ h^3

② Decision tree stumps



A stump defines what we can do with 1 test.
Each possible test is a classifier

↳ 12 possible tests here

The error rate is:

$$\epsilon = \sum_{y_i \neq h(x_i)} w_i \quad \text{s.t.} \sum w_i = 1$$

The weights account for the exageration on some data.

↳ $H(x) = \text{sign}\left(\alpha^1 h^1(x) + \alpha^2 h^2(x) + \alpha^3 h^3(x) + \dots \right)$

General idea of Adaboost:

1) Let $w_i^1 = \frac{1}{N}$, $N = \#$ samples

2) Pick h^t that minimizes ϵ^t → pick α^t

3) Calculate w^{t+1} s.t:

$$w_i^{t+1} = \frac{w_i^t}{Z} e^{-\alpha^t h^t(x) y(x)}$$

where:
- Z normalizing constant ⇒ $\sum w_i = 1$
- $h^t(x)$ predicted classifier value
- $y(x)$ label of training observation.

↳ If y_i and $h^t(x_i)$ have same sign, exponential is negative and weight on observation is small.

We want to choose α^t such that it minimizes the error:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$ is an error bound.

$$\Rightarrow w_i^{t+1} = \frac{w_i^t}{Z} \times \begin{cases} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} & \text{if well classified} \\ \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} & \text{if misclassified} \end{cases}$$

What is the value of the constant Z?

$$\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \cdot \underbrace{\sum_{y_i = h(x_i)} w_i^t}_{1-\epsilon^t} + \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \underbrace{\sum_{y_i \neq h(x_i)} w_i^t}_{\epsilon^t} = Z = 2\sqrt{\epsilon^t(1-\epsilon^t)}$$

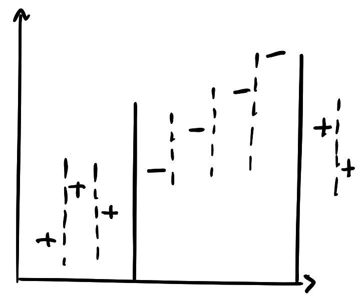
We can replace: $w_i^{t+1} = \begin{cases} \frac{w^t}{2} \cdot \frac{1}{1-\epsilon} & \text{if correct} \\ \frac{w^t}{2} \cdot \frac{1}{\epsilon} & \text{if wrong} \end{cases}$

$\Rightarrow \sum_{correct} w^{t+1} = \frac{1}{2}$ and $\sum_{wrong} w^{t+1} = \frac{1}{2}$ whereas number of misclassified observations decreases. (still larger weights).

Used for computations:

- no Z
- no α
- no exponential.

To reduce computation time, also notice that a classifier between 2 well classified points will never be optimal $\Rightarrow$ No need to compute it.



Limitations: • Adaboost does overfit at some point
↳ early stop or L1-reg.

• Gradient boosting

Idea: At each boosting step, we would like to find how to correct h^{t-1} by gradient descent.

$$-g_t(x_i) = \frac{\partial l(y_i, H_{t-1}(x_i))}{\partial H_{t-1}(x_i)}$$

But gradient only defined for x_i's values, not all x. $\Rightarrow$ Take best approximation.

$$(h_t, \alpha_t) = \underset{h,\alpha}{\arg\min} \sum_{i=1}^{n} \left(l(y_i, H_{t-1}(x_i) + \alpha h) \right) = L(Y_i, H_{t-1}(X_i) + \alpha h(X_i))$$

$$\simeq l(y_i, H_{t-1}(x_i)) + \alpha \langle \nabla l(h_{t-1}), h \rangle$$

Steps : • Replace minimization step by a gradient descent type step
• Choose h_t as the best possible descent direction.
• Choose α_t that minimizes $L(y, H + \alpha h_t)$

$\Rightarrow$ Gradient boosting and Adaboost can be shown to be equivalent.

• Anyboost / Forward Stagewise additive model

1) Set $t=0$, $H_0 = 0$
2) For $t = 1, \dots T$:

• $h_t, \alpha_t = \underset{\alpha, h}{\text{argmin}} \sum_{i=1}^{n} \ell(x_i, H_{t-1}(x_i) + \alpha h(x_i))$

• $H_t = H_{t-1} + \alpha_t h_t$

3) Output $H_T = \sum_{t=1}^{T} \alpha_t h_t$.

Losses might be :

• Adaboost : $\ell(y, h) = e^{-yh}$
• Logit boost : $\ell(y, h) = \log(1 + e^{-yh})$
• L_2 - boost : $\ell(y, h) = (y-h)^2$
• L_1 - boost : $\ell(y, h) = |y-h|$
• Huberboost : $\ell(y, h) = |y-h|^2 \mathbb{1}_{|y-h| \leq \varepsilon} + (2\varepsilon|y-h| - \varepsilon^2) \mathbb{1}_{|y-h| \geq \varepsilon}$

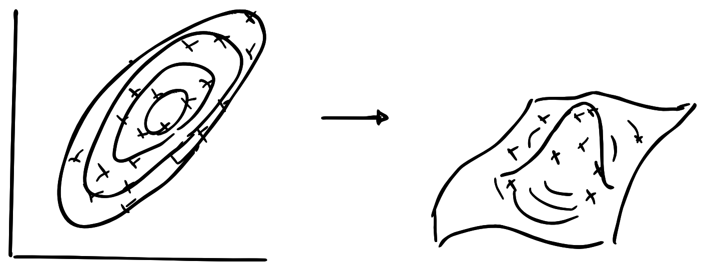
VII. Depth and depth-based classification

1. Depth metrics

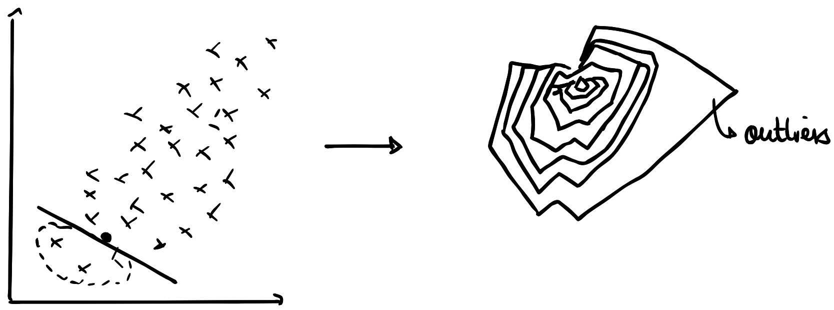
Pick a centroid X and compute depth of point x to X. Adds a dimension to plot the data in 3D.

- Mahalanobis depth $D^{Mah(n)}(x|X) = \dfrac{1}{1 + (x-\mu_x)^T \Sigma_x^{-1} (x-\mu_x)}$

  ↳ Contours are however always elipses, and not robust to outliers

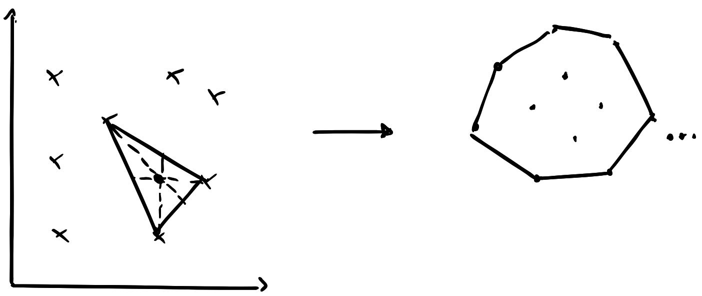


- Tukey depth: $D^{Tuk(n)}(x|X) = \dfrac{1}{n} \min_{u \in S^{d-1}} \# \{ i : u' X_i \geq u' x \}$ number of closed halfspace linked to a point a

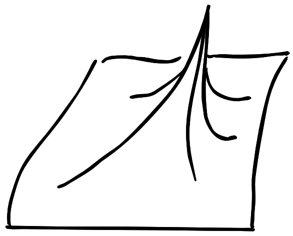


Robust to outliers, and non-parametric.

- Zonoid depth: Number of centers that can be built from e.g 3 observations



- Projection depth: $D^{Prj}(x \mid X) = \dfrac{1}{1 + \sup_{u \in S^{d-1}} \dfrac{|x^T u - \text{med}(X^T u)|}{MAD(X^T u)}}$

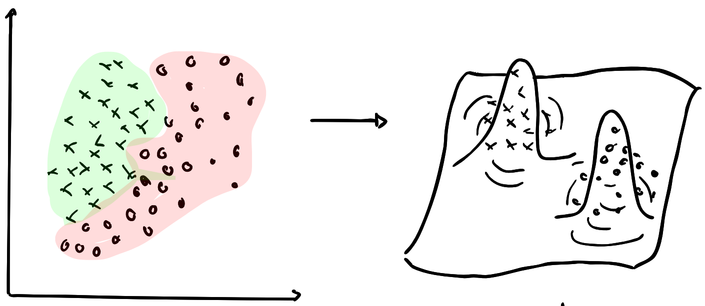


2. Depth-based classification

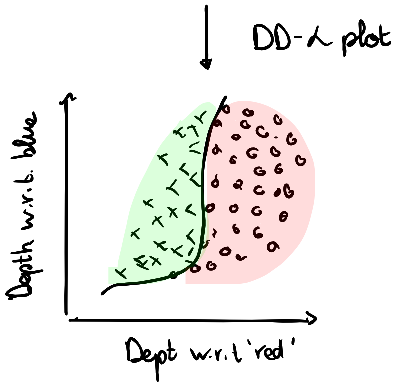
Suppose Q conditional r.v of Q classes, $X_1 \dots X_q$ corresponding training samples containing $n_1 \dots n_q$ observations. Maximal depth classifier.

$$Cl_n^{MD}(x) = \arg\max_{q \in \{1 \dots q\}} D^{(n)}(x \mid X_q)$$

↳ Assign to cluster (with centroid) giving largest depth.



An outsider has zero depth in all classes and is at the origin of the DD-plot.



3. Depth KNN

Define depth-based neighborhood $R_x^{\beta(n)}$ as the smallest depth region $D_\alpha^{(n)}(Y)$ containing k observations from $X_0 \cup X_1$ where $\beta(n) = \dfrac{k}{n_1 + n_2}$

$$Cl_n^{Dknn}(x) = \begin{cases} 0 & \text{if } k_{X_0} \geq k_{X_1} \\ 1 & \text{if } k_{X_0} < k_{X_1} \end{cases} \quad \text{where } k_{X_i} = \#\{z : z \in X_i \cap R_x^{\beta(n)}\}$$