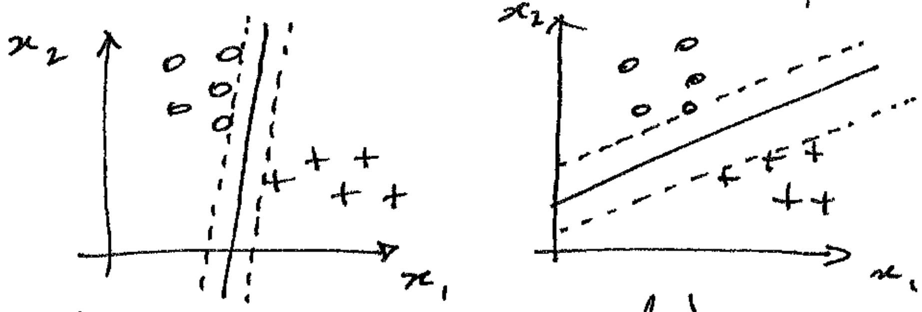
Support vector machines

Intuition: we need a "fat" margin



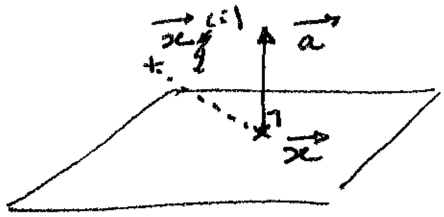
thin margin fat margin.

The hyperplane defining the boundary is given by $\underline{\theta}^T \underline{x} = 0$

Change of notations / habits: no $x_0 = 1$ ⚠

$\theta_0 \longrightarrow b$

$\underline{\theta}^T \underline{x} = 0 \longrightarrow \underline{a}^T \underline{x} + b = 0$



$\vec{x} \in$ plane : $\underline{a}^T \underline{x} + b = 0$

what is the distance between $\underline{x}^{(i)}$ and the plane: $d^{(i)} = \dfrac{a^T}{\|\underline{a}\|} (\underline{x}^{(i)} - \underline{x})$ ← same for any $\underline{x}$

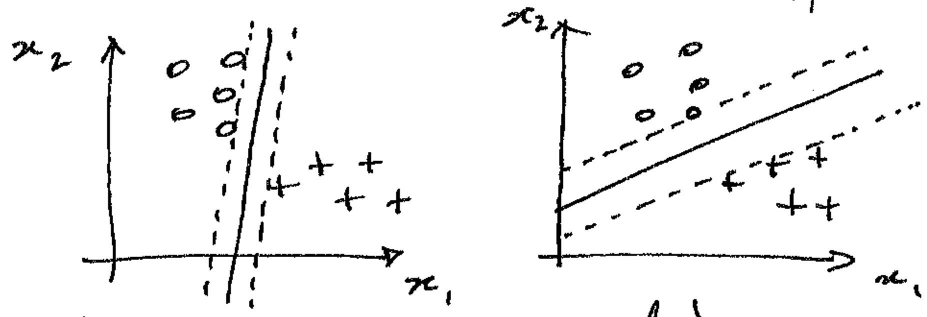
$$d^{(i)} = \frac{1}{\|\underline{a}\|} \left(\underline{a}^T \underline{x}^{(i)} + b - \underline{a}^T \underline{x} - b \right)$$

$$d^{(i)} = \frac{1}{\|\underline{a}\|} \left(\underline{a}^T \underline{x}^{(i)} + b \right)$$

Here, we can constrain $|\underline{a}^T \underline{x}^{(i)} + b| \geqslant 1$ with $\underline{a}^T \underline{x}^{(i)} + b = 1$ for the closest point(s) (we can do that because $\underline{a}$ and b are determined up to a multiplic. constant).

Support vector machines

Intuition: we need a "fat" margin



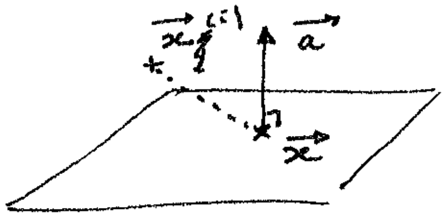
thin margin fat margin.

The hyperplane defining the boundary is given by $\underline{\theta}^T \underline{x} = 0$

Change of notations / habits: no $x_0 = 1$ ⚠

$\theta_0 \longrightarrow b$

$\underline{\theta}^T \underline{x} = 0 \longrightarrow \underline{a}^T \underline{x} + b = 0$



$\vec{x} \in$ plane : $\underline{a}^T \underline{x} + b = 0$

what is the distance between $\underline{x}^{(i)}$ and the plane : $d^{(i)} = \frac{\underline{a}^T}{\|\underline{a}\|} (\underline{x}^{(i)} - \underline{x})$ ← same for any $\underline{x}$

$$d^{(i)} = \frac{1}{\|\underline{a}\|}\left(\underline{a}^T \underline{x}^{(i)} + b - \underline{a}^T \underline{x} - b\right)$$

$$d^{(i)} = \frac{1}{\|\underline{a}\|}\left(\underline{a}^T \underline{x}^{(i)} + b\right)$$

Here, we can constrain $|\underline{a}^T \underline{x}^{(i)} + b| \geq 1$ with $\underline{a}^T \underline{x}^{(i)} + b = 1$ for the closest point(s) (we can do that because $\underline{a}$ and b are determined up to a multiplic. constant).

Optimization problem

Find $\max \frac{1}{\|\underline{a}\|}$ subject to $\min_{i=1..N} |\underline{a}^T \underline{x}^{(i)} + b| = 1$

Or find $\min \|\underline{a}\|^2$ subject to ———

Be aware that $\underline{a}^T \underline{x}^{(i)} + b = \underline{\Theta}^T \underline{x}^{(i)} = h_\Theta(x^{(i)})$

and we want $\left| \begin{array}{ll} h_\Theta(x^{(i)}) \geq 1 & \text{for } y^{(i)} = 1 \\ h_\Theta(x^{(i)}) \leq -1 & \text{for } y^{(i)} = -1 \end{array} \right.$

$$|\underline{a}^T \underline{x}^{(i)} + b| = y^{(i)} (\underline{a}^T \underline{x}^{(i)} + b)$$

Find $\min \frac{\|\underline{a}\|^2}{2}$ subject to $y^{(i)}(\underline{a}^T \underline{x}^{(i)} + b) \geq 1$ for all i's

Quadratic programming

Lagrange formulation

$$\mathcal{L}(\underline{a}, b, \underline{\alpha}) = \frac{1}{2}\|\underline{a}\|^2 - \sum_{i=1}^{N} \underbrace{\alpha_i}_{\geq 0} \underbrace{\left(y^{(i)}(\underline{a}^T \underline{x}^{(i)} + b) - 1\right)}_{\geq 0}$$

We want to find the min $\mathcal{L}$ w.r.t. $\underline{a}$ and b

and maximize ~~each~~ w.r.t. $\alpha_i \geq 0$ (KKT ≠ Lagrange multiplier)

$$\left.\begin{array}{l} \text{grad}_{\vec{a}} \mathcal{L} = \underline{a} - \sum_{i=1}^{N} \alpha_i y^{(i)} \underline{x}^{(i)} = \underline{0} \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \end{array}\right\} \begin{array}{l}\text{min} \\ \text{w.r.t} \\ \underline{a}, b\end{array}$$

Then $\underline{a} = \sum_{i=1}^{N} \alpha_i y^{(i)} \underline{x}^{(i)}$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

Substituting in $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}(\alpha) = \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j \underline{x}^{(i)T} \underline{x}^{(j)}}_{\|\underline{a}\|^2}$$

$$- \sum_{i=1}^{N} \alpha_i \left(-1 + b \cancel{y^{(i)}} + y_i \underline{x}^{(i)T} \sum_{j=1}^{N} \alpha_j y^{(j)} \underline{x}^{(j)}\right)$$

$$\mathcal{L} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N}\sum_{j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j \underline{x}^{(i)T} \underline{x}^{(j)}$$

we want to find $\max(\mathcal{L})$ w.r.t α

with $\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$ and $\alpha_i \geq 0$

This is a quadratic optimization problem for which there is plenty of methods...

One of them is a penalty method (iterative): at each step, we maximize:

$$\mathcal{L} - \tau_k \sum_i \min(\alpha_i, 0) - \tau_k \left(\sum_i \alpha_i y^{(i)}\right)^2$$

with τ_k increasing at each step (10-fold, Brinkman)

<u>Soft margin</u>

If data is not linearly separable, we use a "hinge loss" function

$$\max(0, 1 - y^{(i)}(\underline{a}^T \underline{x}^{(i)} + b))$$

this function is 0 if the output $y^{(i)} = \pm 1$ is correctly predicted, otherwise it is prop. to the distance to the margin.

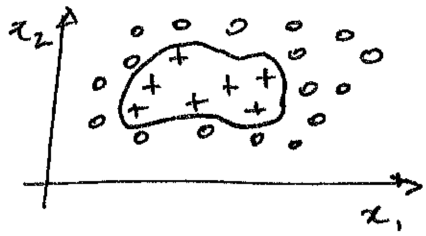
We want to minimize:

$$\lambda \|\underline{a}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max\left(0,\, 1 - y^{(i)}\left(\underline{a}^T \underline{x}^{(i)} + b\right)\right)$$

λ is the trade-off between the margin size and ensuring that $\underline{x}^{(i)}$'s lie on the correct side.

In the limit $\lambda \longrightarrow 0$, it converges towards a hard-margin SVM.

SVM with kernels



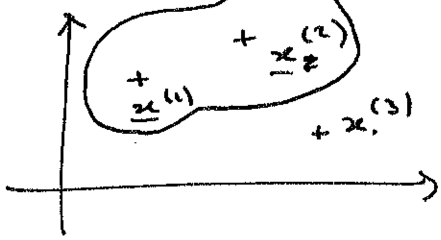
non-linear decision boundary

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \dots \geq 0$$

$$f_1 = x_1,\; f_2 = x_2,\; f_3 = x_1^2 \quad : \text{features}$$

Is there | a better choice of features?
| an alternative

Gaussian kernel: $f_1 = \exp\left(-\frac{\|\underline{x} - \underline{x}^{(1)}\|^2}{2\sigma^2}\right)$



Intuition if

$\theta_0 = -0,5 \qquad \theta_1 = \theta_2 = 1$

$\theta_3 = 0$

Q: how many landmarks?
how to choose them?

$\longrightarrow$ all $\underline{x}^{(i)}$'s

$$\underline{x}^{(i)} \longrightarrow \begin{bmatrix} f_1 = \mathrm{sim}(\underline{x}^{(i)}, \underline{x}^{(1)}) \\ \vdots \\ f_i = 1 \\ \vdots \\ f_N \end{bmatrix} \underline{f}^{(i)} \qquad \begin{aligned} \underline{x}^{(i)} &\in \mathbb{R}^{n(+1)} \\ \underline{f}^{(i)} &\in \mathbb{R}^{N(+1)} \end{aligned}$$

Logistic vs. SVMs

n : number of features

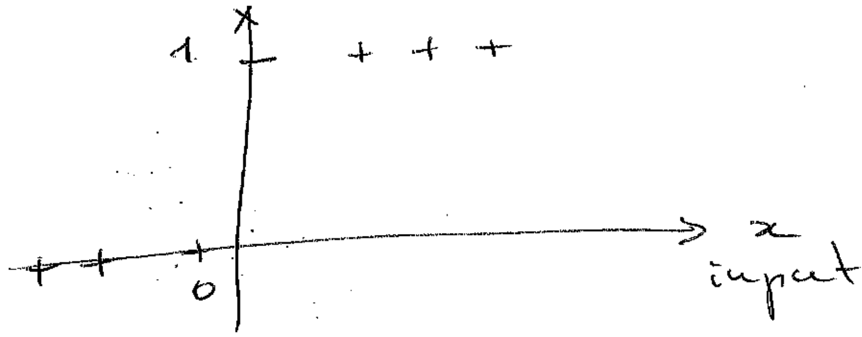
N : ——— training examples

- $n \geq N$ ⟶ logistic or SVM without kernel
- n small
  - N intermediate ($\in [10\ 10000]$) ⟶ Gaussian kernel
  - N large (> 50000) ⟶ create more features and then logistic or SVM without kernel

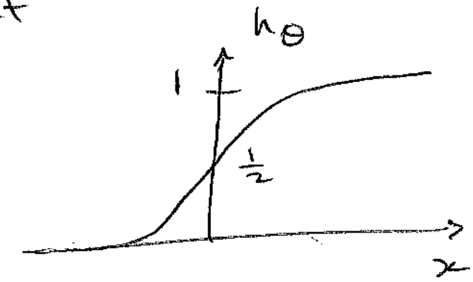
⟶ neural network is an alternative in all cases.

Logistic regression

Useful for classification



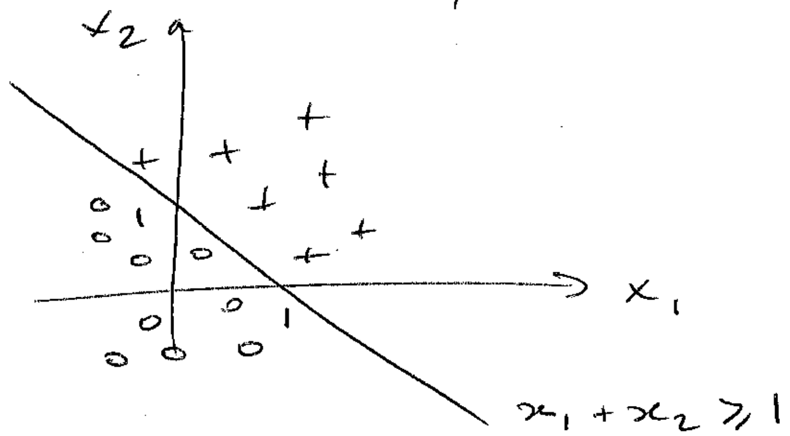
hypothesis : $h_\theta = \dfrac{1}{1+e^{-\underline{\theta}^T \underline{x}}}$



(logistic function)

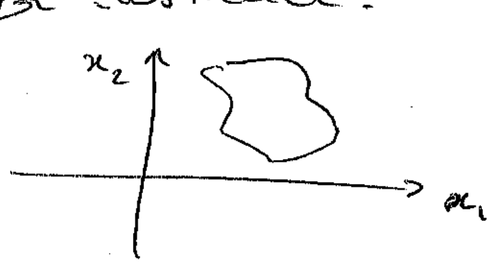
Interpretation : $h_\theta = p(y=1 \mid \underline{x} ; \theta)$

Decision boundary



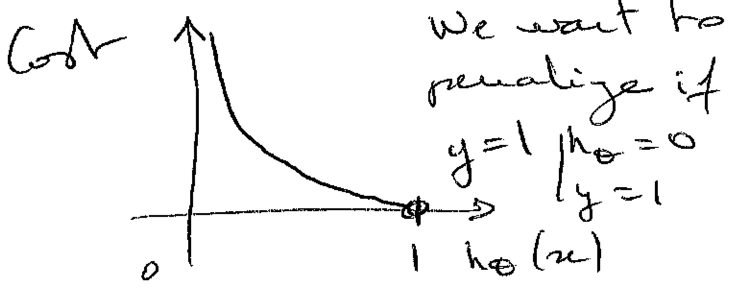
$\underline{\theta}^T \underline{x} = -1 + x_1 + x_2$

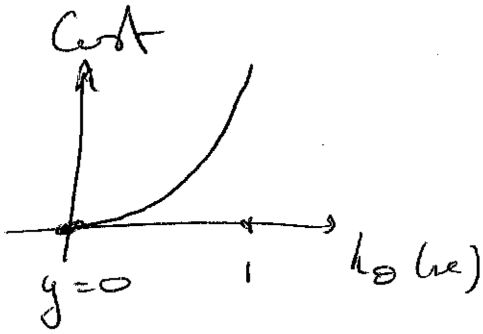
Works with larger order polynomials if $x_2 = x_1^2$ for instance.



$$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ \vdots \end{pmatrix}$$

Cost function





$$\text{Cost}: \begin{cases} -\log(h_\theta) & \text{if } y=1 \\ -\log(1-h_\theta) & \text{if } y=0 \end{cases}$$

Generalization:

$$J(\underline{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\Big[y^{(i)}\log\big(h_\theta(\underline{x}^{(i)})\big) + (1-y^{(i)})\log\big(1-h_\theta(\underline{x}^{(i)})\big)\Big]$$

(Analogy with Shannon entropy).

Simplification

$$h_\theta(\underline{x}) = \frac{1}{1+e^{-\underline{\theta}^T\underline{x}}}$$

$$\log h_\theta = -\log\left(1+e^{-\underline{\theta}^T\underline{x}}\right)$$

$$\frac{\partial}{\theta_j}\log h_\theta = -\frac{1}{1+e^{-\underline{\theta}^T\underline{x}}}\left(-x_j e^{-\underline{\theta}^T\underline{x}}\right) = \frac{x_j}{1+e^{\underline{\theta}^T\underline{x}}}$$

$$\log(1-h_\theta) = +\log\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}} = -\log\left(\frac{1}{e^{-\theta^T x}}+1\right)$$

$$\frac{\partial}{\partial\theta_j}\log(1-h_\theta) = -\frac{1}{1+e^{\underline{\theta}^T\underline{x}}}\left(x_j e^{+\theta^T x}\right)$$

$y^{(i)} \mathbin{\not} 2b.$

$$\frac{\partial}{\partial \theta_j}\left[y \log h_\theta + (1-y)\log(1-h_\theta)\right]$$

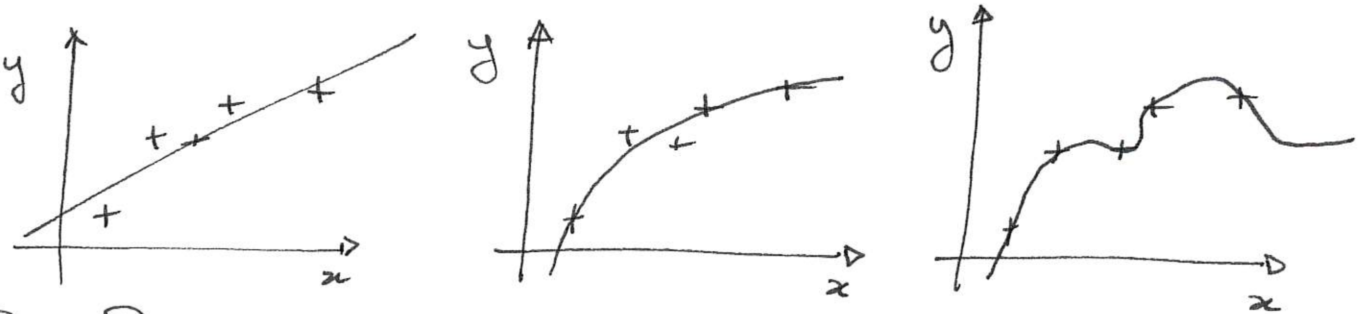
$$= x_j\left[\frac{1}{1+e^{\theta^T x}}\left(y - (1-y)\,e^{\theta^T x}\right)\right]$$

$$= x_j\left[\frac{-1}{1+e^{-\theta^T x}} + y\right]$$

$$\Longrightarrow \theta_j := \theta_j - \alpha \frac{1}{N}\sum_{i=1}^{N}\Big(\underbrace{h_\theta(x^{(i)})}_{\frac{1}{1+e^{-\theta^T x}}} - y^{(i)}\Big)x_j^{(i)}$$

Same formula as linear regression!

Regularization (Pb of overfitting)



$\theta_0 + \theta_1 x$ "underfitting"

$\theta_0 + \theta_1 x + \theta_2 x^2$ "feels just"

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ "overfitting"

Same if very large number of x_j's
Same with logistic regression...

* Intuition $\longrightarrow$ we want to keep θ_j's small

* Solution : new cost function

$$J(\underline{\theta}) = \frac{1}{2N}\left[\sum_{i=1}^{N}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \underbrace{\lambda\sum_{j=1}^{n}\theta_j^2}_{\text{new term}}\right]$$

⚠ starts at $j=1$
does not include θ_0

λ : regularisation parameter

* Application to linear regression

$$\left.\frac{\partial J}{\partial \theta_0} = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\underbrace{x_0^{(i)}}_{=1}\right\}\ \text{idem sans regularis}^{\circ}$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{N}\theta_j$$

Gradient descent

$$\theta_j := \theta_j - \alpha \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{N} \theta_j \right\}$$

$$\theta_j := \theta_j \underbrace{\left(1 - \frac{\alpha \lambda}{N} \right)}_{\substack{\in [0,1] \\ \text{typically } 0.99}} - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Normal equation

$$\underline{\theta} = \left(\underline{\underline{X}}^T \underline{\underline{X}}^{-1} + \lambda \begin{bmatrix} 0 & & 0 \\ & 1 & \\ 0 & & 1 \end{bmatrix} \right)^{-1} \underline{\underline{X}}^T \underline{y}$$

Can make the first matrix invertible when the number of examples is less than the number of features.

Logistic equation

Same gradient descent as linear regression.

Naive Bayes

Classifier based on conditional probability
$p(C_k | \underline{x})$

Probability of being of class C_k among K possible classes, given $\underline{x}$

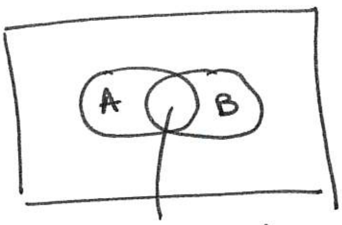
Bayes theorem:

prior

$$p(C_k | \underline{x}) = \frac{p(C_k)\, p(\underline{x} | C_k)}{p(\underline{x})}$$

support that C_k provides for $\underline{x}$

"Demonstration":



$p(A \cap B)$

$$p(A \cap B) = p(A)\,p(B|A) = p(B)\,p(A|B)$$

$$\Longrightarrow p(A|B) = p(A) \frac{p(B|A)}{p(B)}$$

$$p(C_k \cancel{|} \underline{x}) = p(C_k | \underline{x})\, p(\underline{x}) = p(C_k)\, p(\underline{x} | C_k)$$

$$p(C_k, \underline{x}) = p(x_1, \dots, x_n, C_k)$$

$$= p(x_1 | x_2, \dots, x_n, C_k)\, p(x_2, \dots x_n, C_k)$$

$$= p(x_1 | x_2, \dots x_n, C_k) \cdots p(x_n | C_k)\, p(C_k)$$

"naive" independent assumption

$$= p(x_1 | C_k) \dots p(x_n | C_k)\, p(C_k)$$

$$p(C_k \mid \underline{x}) = \frac{p(C_k, \underline{x})}{p(\underline{x})}$$

$$= \frac{p(C_k)}{p(\underline{x})} \prod_{i=1}^{n} p(x_i \mid C_k)$$

This can be used to construct a classifier

$$y = \underset{k \in [1..K]}{\operatorname{argmax}}\; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

(we can drop $p(\underline{x})$ which is a constant)

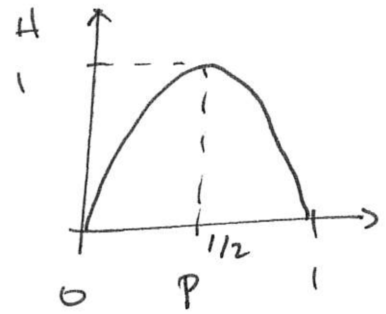
$p(C_k)$ and $p(x_i \mid C_k)$ can be learned from training dataset.

Decision trees

* Crash course on information theory (Shannon)

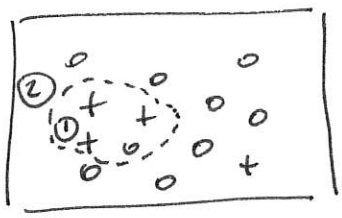
Shannon entropy (in bits per symbol):

$$H = -\sum_i p_i \log_2 (p_i)$$



binary entropy:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$



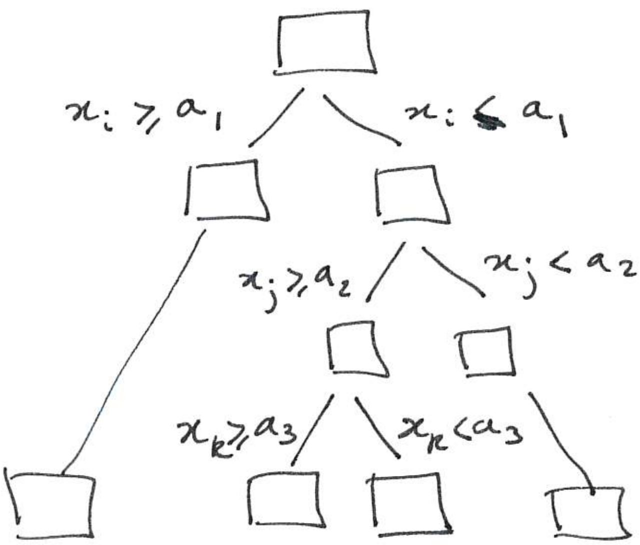
H : entropy of the total set N

$$H_{split} = \frac{N_1}{N} H_1 + \frac{N_2}{N} H_2$$

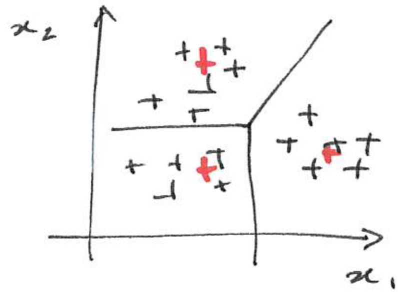
If $H_1 = H_2 = H \implies H_{split} = H$ no gain

$H_1 = H_2 = 0$ (when only one label in each subset). $\implies H_{split} = H$

The idea is to maximize the entropy loss. (or information gain) at each split.



k-means



We want to partition the space into k cells (Voronoi cells defined by their centers)

Pb computationally difficult (NP-hard)

k-means is an efficient heuristic algorithm.

The objective of k-means clustering is to find

$$\operatorname{argmin} \sum_{j=1}^{K} \sum_{\underline{x} \in S_j} \| \underline{x} - \underline{\mu}_j \|^2$$

where $\underline{\mu}_i$ is the mean of points in S_i

Standard algorithm

- Randomly choose K centroids $\underline{\mu}_j \in \mathbb{R}^n$
- For the N $\underline{x}^{(i)}$ find the cluster $c^{(i)} \in \{1 \dots K\}$ to which it belongs by computing $\| x^{(i)} - \underline{\mu}_j \|^2$
- Calculate the average of points assigned to each cluster
- update cluster centroid positions

repeat

Possibility to measure efficiency with objective function

$$J(\underline{\mu}_1 \dots \underline{\mu}_K) = \frac{1}{N} \sum_{i=1}^{N} \| \underline{x}^{(i)} - \underline{\mu}_{c^{(i)}} \|^2$$

index of cluster

→ Random initialization and then select result with smallest J

→ Choosing number of cluster