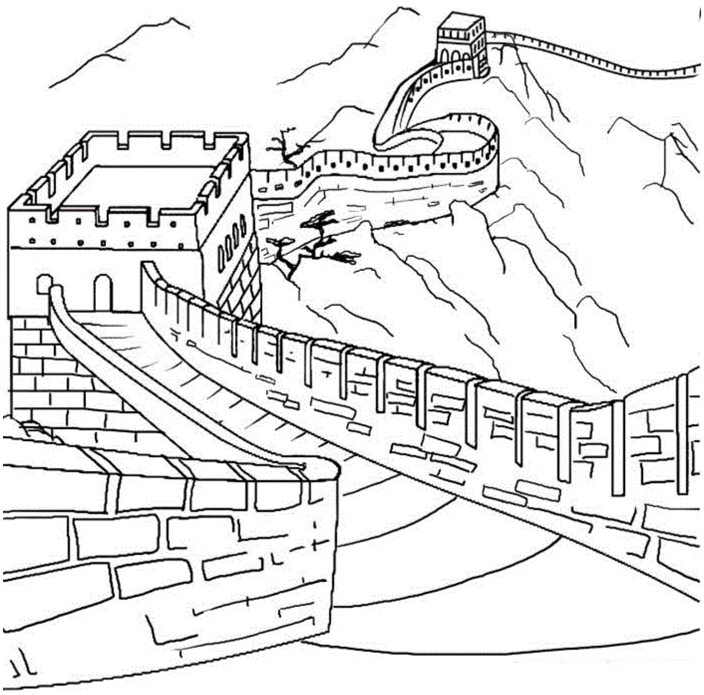
Machine Learning
And Intelligent Systems

LECTURE SLIDES

PART 3



Prof. Bernard Merialdo
Fall 2017

Data ScienceDepartment
EURECOM

Content Part 3

Support Vector Machines

(slides inspired from Martin Law, MSU and Andrew W. Moore, CMU)

Machine Learning: VC Dimension

(Vapnik-Chervonenkis dimension)

- Assume T training samples (x_i,y_i) i=1,…T
  - $x_i \in \mathbb{R}^n$, "label" $y_i \in \{-1,1\}$
- Assume the samples are iid from probability distribution P(x,y)
  - (independent and identically distributed)
- Consider a family of machines (classifiers) $f(x,\alpha)$

$$f(x,\alpha) : X \rightarrow \{-1,1\}$$

and try to learn the mapping $x_i \rightarrow y_i$

  - Find the value for α which minimizes the error rate

$$\min \#\{i: f(x_i,\alpha) \neq y_i\}$$

Machine Learning: VC Dimension

- Expected test error (risk):

$$R(\alpha) = \int_{x,y} \frac{1}{2} | y - f(x,\alpha) | \, dP(x,y)$$

  - This is the average error rate on all possible values (x,y)
  - This is also the probability of misclassification
  - This is ideal, but cannot be computed in practice
- Empirical risk:
  - We approximate by averaging only on the training data

$$R_{emp}(\alpha) = \frac{1}{2T} \sum_{i=1}^{T} | y_i - f(x_i,\alpha) |$$

  - Less ideal, can be computed, but may lead to overfit

Machine Learning: VC Dimension

- Generally, we manage overfit by watching the error rate of the classifier on validation data
- Idea of VC:
  - Instead of using validation data, we look at the complexity of the classifier
  - If the classifier is complex, it is easier to reduce the error rate on training and overfit
  - If the classifier is simple, it is more difficult to reduce the error rate, but less chances to overfit
  - VC tries to optimize a combination between error rate and classifier complexity

Machine Learning: VC Dimension

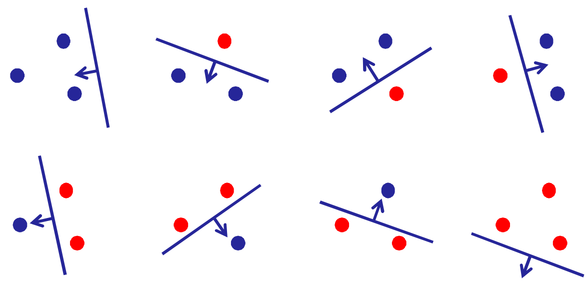
- N points can be labeled in 2^N ways
  - For each point, possible labels are -1, +1
- Let $\mathcal{H}$ a set of machines $\{f(x, \alpha)\}$
- Definition: $\mathcal{H}$ shatters N if there exists a set of N points such that for all 2^N possible labellings, there exists a consistent $f(x, \alpha) \in \mathcal{H}$
  - "Consistent" means: $f(x_i,\alpha)= y_i$ for all $i=1,\ldots N$
  - This means that any learning problem definable by those N examples can be learned with no error by an element of $\mathcal{H}$
  - We only need to find **one** such set of N points

Z

Machine Learning: VC Dimension

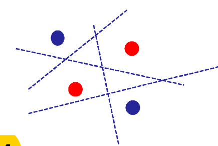
- Example: $\mathcal{H}$={lines in $\mathbb{R}^2$}
- N=3

OK



- N=4

impossible

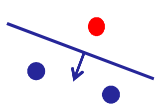


- $\mathcal{H}$ shatters 3, not 4

Machine Learning: VC Dimension

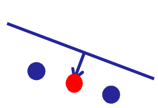
*****WARNING*****

- The definition is:
- $\mathcal{H}$ shatters *N* if there exists a set of N points such that for all 2^N possible labellings, there exists a consistent f(x, α)∈ $\mathcal{H}$
- The definition is NOT:
- $\mathcal{H}$ shatters *N* if for all sets of N points and for all 2^N possible labellings, there exists a consistent f(x, α)∈ $\mathcal{H}$
- Example*: linear classifiers in the plane*



There exists

I can select 3 non aligned points



For all

It has to be true for all sets, also for aligned points

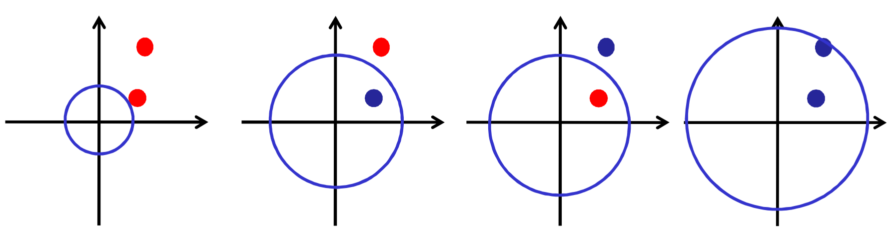
Machine Learning: VC Dimension

- Definition: the maximum value for N that $\mathcal{H}$ shatters is called the Vapnik-Chervonenkis dimension of $\mathcal{H}$:
  - $VC(\mathcal{H}) = N$
- For example:
  - $VC(\{\text{lines in } \mathbb{R}^2\}) = 3$
  - In general: $VC(\{\text{hyperplanes in } \mathbb{R}^d\}) = d+1$

The basic idea is that, when the VC is high, there are more chances to overfit to the training data, so we should try to find models with low VC

Machine Learning: VC Dimension

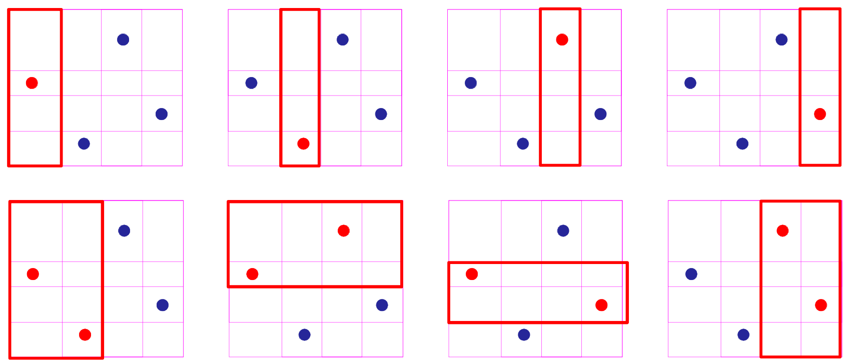
- Example: $\mathcal{H}$={circles centered at origin in $\mathbb{R}^2$}
  - $f(x, \alpha)=\text{sign}(\|x\|^2-\alpha)$ or $f(x, \alpha)=-\text{sign}(\|x\|^2-\alpha)$



- $VC(\mathcal{H}) = 2$

Machine Learning: VC Dimension

- Example: $\mathcal{H}$ ={rectangles with borders parallel to axes in $\mathbb{R}^2$}



- VC($\mathcal{H}$) = 4

Machine Learning: VC Dimension

- Let ε between 0 and 1. Vapnik (1995) showed that :

$$P\left(R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\log(2T/h)+1)-\log(\varepsilon/4)}{T}} \right) \ge 1-\varepsilon$$

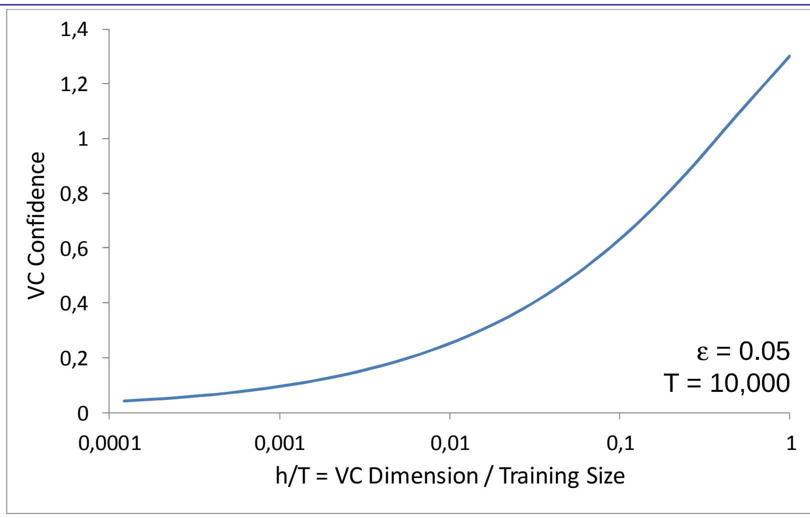
  - With VC($\mathcal{H}$) = h
  - Note the bound is independent of P(x,y)

Expected_Error ≤ Estimated_Error + VC_Confidence

Probable Upper bound

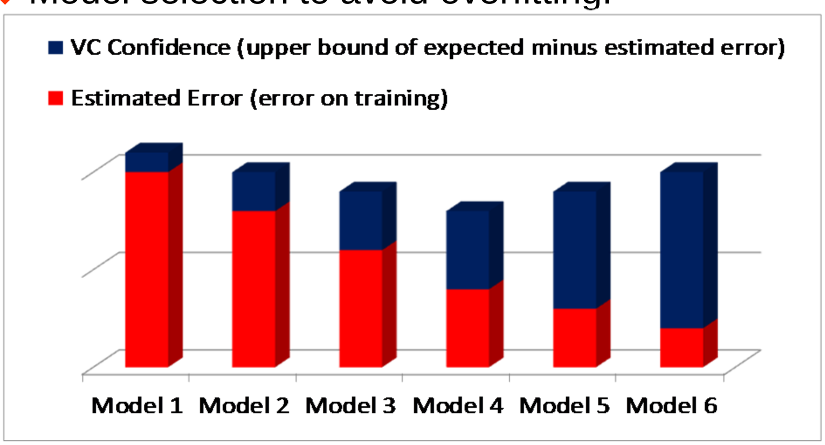
- When choosing between models with different VC, select the one with minimum upper bound

Machine Learning: VC Dimension



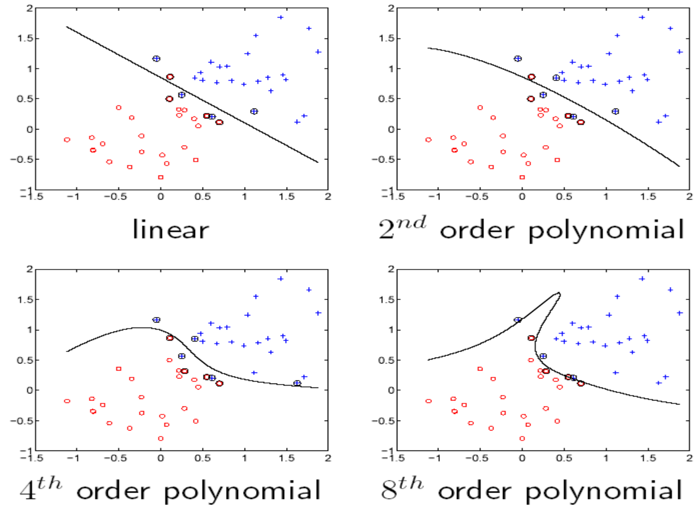
Machine Learning: VC Dimension

- Model selection to avoid overfitting:

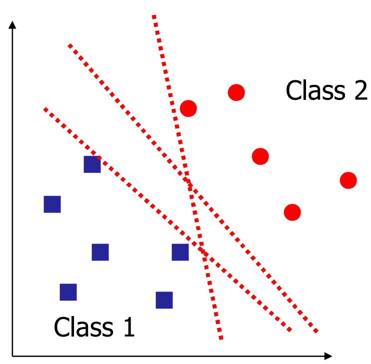


- M4 should be selected (min VC + est. Error)

Machine Learning: VC Dimension



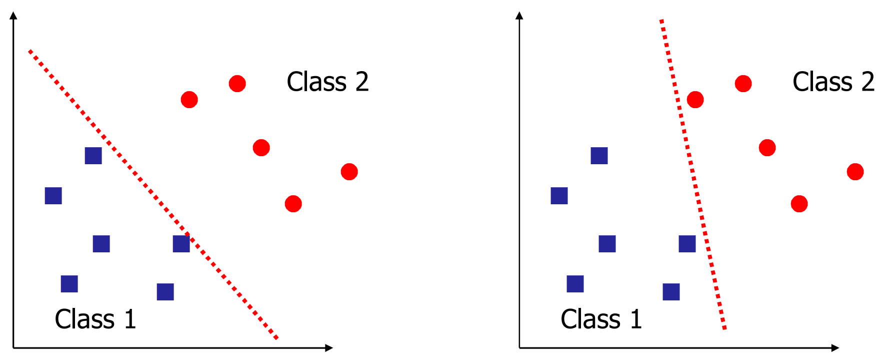
Linearly Separable Sets



- Assume two linearly separable sets
- Many decision boundaries can separate these two classes
- Which one should we choose?

Linearly Separable Sets

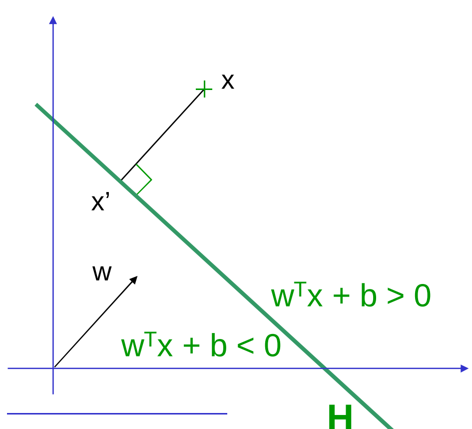
- Bad decision boundaries: high risk of misclassification for new data



Distance from an hyperplane

- Hyperplane H equation:

$$w^T x + b = 0$$



- Distance to hyperplane:

x' projection of x over H :

$$x' = x - \lambda w \qquad \lambda \text{ real}$$

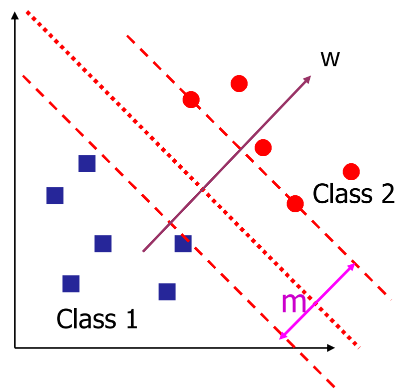
x' belongs to H :

$$w^T x' + b = 0$$

$$\text{so :} \quad \lambda = \frac{w^T x + b}{w^T w}$$

$$d(x,H) = \|x - x'\| = |\lambda| \|w\| = \frac{|w^T x + b|}{\|w\|}$$

Linearly Separable Sets: boundary margin



- Assume boundary B is given
- Equation: $w^T x + b = 0$
- Distance from class to boundary:

$$d(C_1,B) = \min_{x_1 \in C_1} \frac{|w^T x_1 + b|}{\|w\|}$$

- Margin m:

$$m = \min_{x_1,x_2} \frac{|w^T x_1 + b| + |w^T x_2 + b|}{\|w\|}$$

Linearly Separable Sets: best boundary

- Theorem (Vapnik, 1982)
  - Given N data points $X=\{x_1, . . . x_N\}$ in $\mathbb{R}^d$, with $\|x_i\| \leq A$
  - $\mathcal{H}\gamma$ =set of linear classifiers in $\mathbb{R}^d$ with margin γ on X
- Then, the VC dimension of $\mathcal{H}\gamma$ is bounded by:

$$VC(\mathcal{H}\gamma) \leq \min\left\{ d, \left\lceil \frac{4A^2}{\gamma^2} \right\rceil \right\}$$

- The larger is the margin, the lower is the upper bound on the VC dimension

Linearly Separable Sets: best boundary

- The two classes should be at the same distance (no bias):

$$d(C_1,B) = \min_{x_1 \in C_1} \frac{|w^T x_1 + b|}{\|w\|} = d(C_2,B) = \min_{x_2 \in C_2} \frac{|w^T x_2 + b|}{\|w\|}$$

$$\text{Let } c = \min_{x_1 \in C_1} |w^T x_1 + b| = \min_{x_2 \in C_2} |w^T x_2 + b|$$

- The margin should be maximal:

$$m = \max_{w,b} \min_{x_1, x_2} \frac{|w^T x_1 + b| + |w^T x_2 + b|}{\|w\|} = \max_{w,b} \frac{2c}{\|w\|}$$

Best Boundary: simplification

- We can assume c = 1:

(just use $w' = \frac{w}{c}$, $b' = \frac{b}{c}$, this is the same boundary)

- Then, the classes verify:

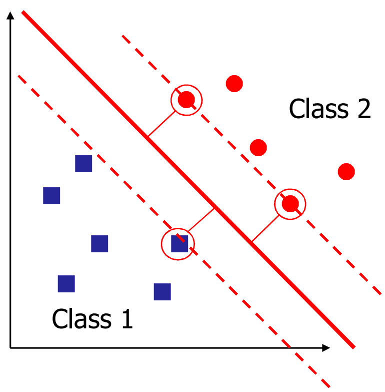
$$(1) \begin{cases} \forall x_1 \in C_1 & w^T x_1 + b \leq -1 \\ \forall x_2 \in C_2 & w^T x_2 + b \geq +1 \end{cases}$$

- The best marginal verifies:

$$m = \max_{w,b} \frac{2}{\|w\|} = \frac{2}{\min_w \|w\|}$$

with w, b which verify (1)

Support Vectors



- Points which lie on the margin are called « Support Vectors »
- Support vectors verify:

$$w^T x + b = 1$$

or

$$w^T x + b = -1$$

Best Boundary : how to find it ?

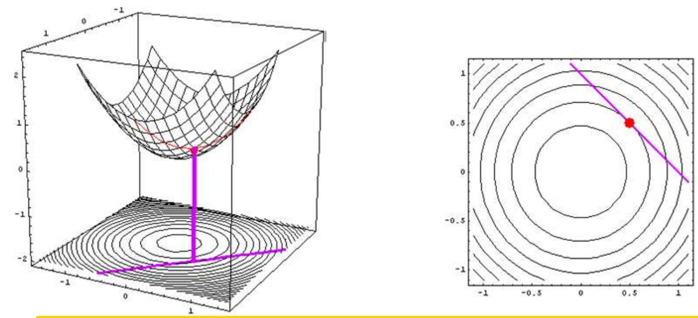
- Let $\{x_1, \ldots, x_n\}$ be the sample points
- Let $y_i \in \{-1,1\}$ be the class label of x_i
- The best boundary verifies:

$$\begin{cases} \min_{w} \frac{1}{2}\|w\|^2 \\ \text{constraints}: \forall i \quad y_i\left(w^T x_i + b\right) \geq 1 \end{cases}$$

- This is an problem of optimization with constraints
  - The objective function is quadratic
  - The constraints are linear

QP Optimization

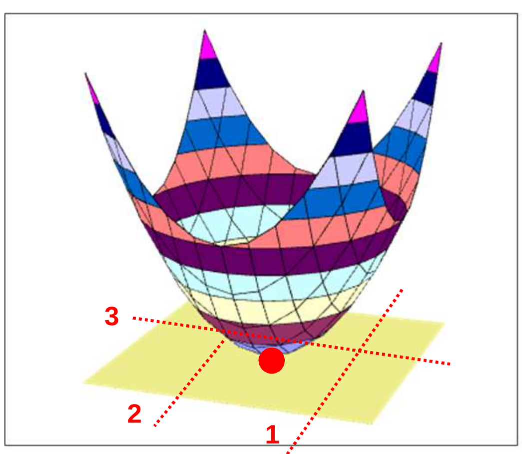
- A Quadratic Programming problem can be solved by standard techniques and softwares



- It is a convex problem, so we can guarantee a global minimum

QP Optimization

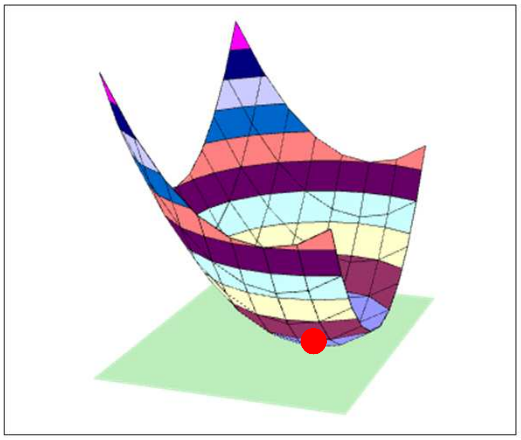
- min f(x) without constraints



$$g_1(x) = 0$$
$$g_2(x) = 0$$
$$g_3(x) = 0$$

QP Optimization

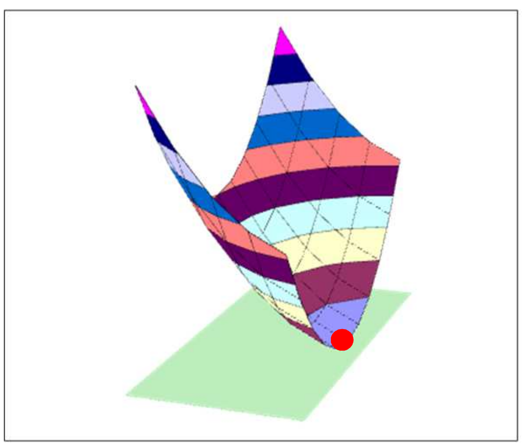
- min f(x) with constraints



$g_1(x) \leq 0$

QP Optimization

- min f(x) with constraints

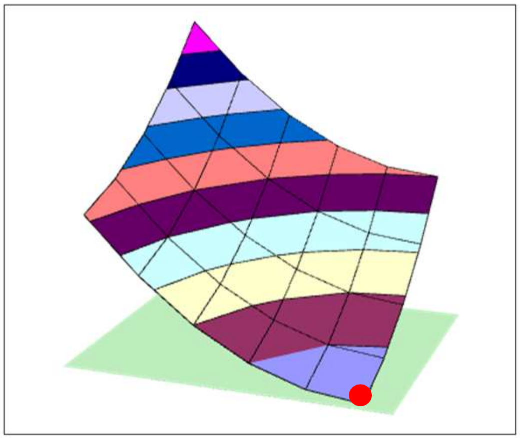


$g_1(x) \leq 0$

$g_2(x) \leq 0$

QP Optimization

- min f(x) with constraints



$g_1(x) \leq 0$
$g_2(x) \leq 0$
$g_3(x) \leq 0$

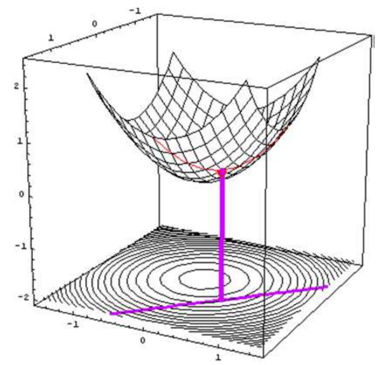
Lagrange optimization

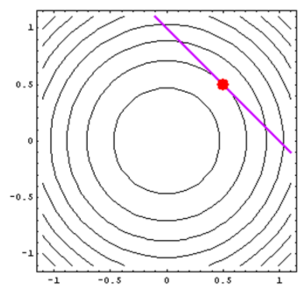
- Optimization without constraints
- Let f(x): $\mathbb{R}^n \rightarrow \mathbb{R}$ differentiable
- How to find min f(x) ?

1. Compute $\frac{\partial f(x)}{\partial x_i}$
2. Find x such that $\frac{\partial f(x)}{\partial x_i} = 0$ for all i=1,2…n
3. Check if x is a minimum or a maximum

Lagrange optimization

- Optimization with one equality constraint
- How to find min f(x) with constraint g(x) = 0 ?



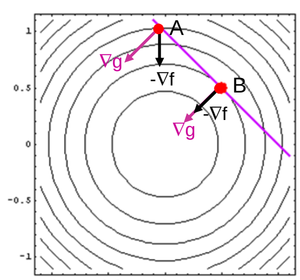


Lagrange optimization

- Optimization with one equality constraint
- How to find min f(x) with constraint g(x) = 0 ?
  - In A, it is possible to decrease f and keep the constraint
  - In B, it is not possible because:

    $\nabla f = -\lambda \nabla g$
  - Define Lagrangian:

    $L(x, \lambda) = f(x) + \lambda g(x)$
  - Then, in B we have: $\begin{cases} \dfrac{\partial L(x,\lambda)}{\partial x_i} = 0 \\ \dfrac{\partial L(x,\lambda)}{\partial \lambda} = 0 \end{cases}$



Lagrange optimization

- Optimization with several equality constraints
- How to find min f(x) with constraints $g_j(x) = 0$?
  1. Define Lagrangian $L(x, \lambda) = f(x) + \sum_j \lambda_j g_j(x)$

     λ_j Lagrange multipliers
  2. Find (x, λ) such that:
     $$\begin{cases} \dfrac{\partial L(x,\lambda)}{\partial x_i} = 0 \\ \dfrac{\partial L(x,\lambda)}{\partial \lambda_j} = 0 \end{cases}$$
  3. Check if x is a minimum or a maximum

Lagrange optimization

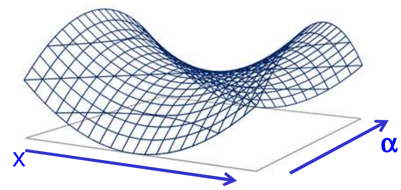
- Optimization with one inequality constraint $g(x) \leq 0$
- Lagrangian: $L(x, \alpha) = f(x) + \alpha\, g(x)$
- If x satisfies the constraint and $\alpha \geq 0$:

$$\max_{\alpha \geq 0} L(x, \alpha) = f(x)$$

- The original problem is : $\min_x \max_{\alpha \geq 0} L(x, \alpha)$
- The dual problem is : $\max_{\alpha \geq 0} \min_x L(x, \alpha)$
- Under certain conditions, both problems are equivalent

Lagrange optimization

- Optimization with one inequality constraint
- We look for a saddle point of $L(x,\alpha)$
- So: $\begin{cases} \dfrac{\partial L(x,\alpha)}{\partial x} = 0 \\ \dfrac{\partial L(x,\alpha)}{\partial \alpha} = 0 \end{cases}$



  - And x should verify the constraint
- Idea for solving:
  - Step 1: minimize in x: $\nabla f(x) + \alpha \nabla g(x) = 0 \rightarrow x = \varphi(\alpha)$
  - Step 2: maximize in α : $f(\varphi(\alpha)) + \alpha\, g(\varphi(\alpha))$ with $\alpha \geq 0$
  - Step 3: deduce value of x and check constraint

Lagrange optimization

- Optimization with several inequality constraints
- How to find min f(x) with constraints $g_j(x) \leq 0$?
  - Define Lagrangian $L(x,\alpha) = f(x) + \sum_j \alpha_j g_j(x)$
    - $\alpha_j \geq 0$ Lagrange multipliers
  - Find (x, α) such that:

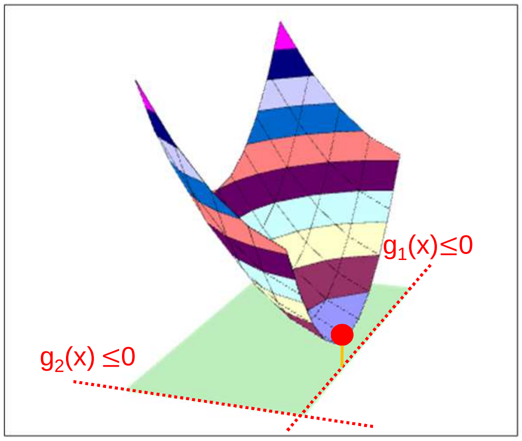
$$\begin{cases} \dfrac{\partial L(x,\alpha)}{\partial x_i} = 0 \\ g_j(x) \leq 0 \\ \alpha_j \geq 0 \\ \alpha_j\, g_j(x) = 0 \end{cases}$$

(Karush-Kuhn-Tucker conditions)

  - Check if min or max

Lagrange optimization

- min f(x) with constraints: $g_1(x) \leq 0 \quad g_2(x) \leq 0$



$g_1(x) \leq 0$
$g_2(x) \leq 0$

Best Boundary: Lagrange Optimization

- min $\frac{1}{2}\|w\|^2$ with constraints : $\forall i \quad y_i\left(w^T x_i + b\right) \geq 1$
- $L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i\left[y_i\left(w^T x_i + b\right) - 1\right]$
- Step 1: $\begin{cases} \dfrac{\partial L(w,b,\alpha)}{\partial w_j} = w_j - \sum_i \alpha_i y_i x_{ij} = 0 \\ \dfrac{\partial L(w,b,\alpha)}{\partial b} = -\sum_i \alpha_i y_i = 0 \end{cases}$
  - so $w = \sum_i \alpha_i y_i x_i \qquad \sum_i \alpha_i y_i = 0$

Best Boundary: Lagrange Optimization

$$w = \sum_i \alpha_i y_i x_i \qquad \sum_i \alpha_i y_i = 0$$

- Step 2:

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right] \\ &= \frac{1}{2}\|w\|^2 - w^T \sum_i \alpha_i y_i x_i - b \sum_i \alpha_i y_i + \sum_i \alpha_i \\ &= \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

Best Boundary: Lagrange Optimization

- Dual problem:

$$\begin{cases} \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{with} \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0 \end{cases}$$

- Should also satisfy Karush-Kuhn-Tucker conditions:

$$\alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right] = 0 \qquad \forall i$$

- Note that $\alpha_i > 0$ iff x_i is a support vector

Best Boundary: Summary

- Given samples $\{x_1, ..., x_n\}$ we want to find the best separator
- Solve the Dual problem:

$$\begin{cases} \max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{with} \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0 \end{cases}$$

  - This gives the values of α_i
- When $\alpha_i > 0$ then x_i is a support vector (lies on the margin)
- When $\alpha_i = 0$ then x_i is outside the margin

Best Boundary: Summary

- We can then compute w and b:

$$w = \sum_i \alpha_i y_i x_i$$ (linear combination of support vectors)

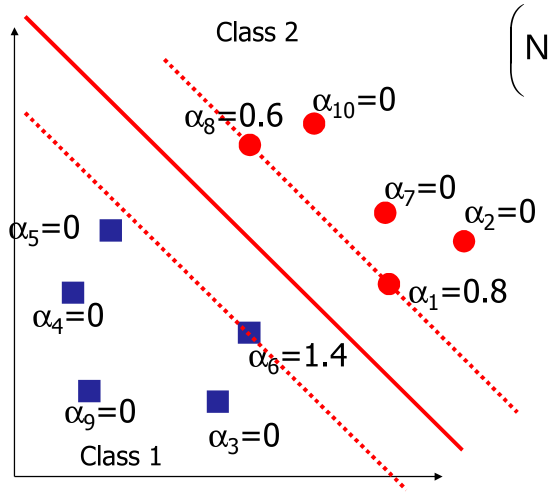
- b is computed from $\alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right] = 0 \quad \forall i$
  - If x_i is a support vector:

$$y_i \left(w^T x_i + b \right) = 1$$
$$b = y_i - w^T x_i$$

  - In practise, it is better to average:

$$b = \frac{1}{\#\{i : \alpha_i > 0\}} \sum_{i:\alpha_i > 0} \left(y_i - w^T x_i \right)$$

Geometrical Interpretation



$\left(\text{Note that } \sum_i \alpha_i y_i = 0 \right)$

Remarks

- A new point z can be easily classified:

  $w^T z + b < 0$ class 1

  $w^T z + b > 0$ class 2

- Note that data points are used through inner product only:
  - Training: $x_i^T x_j$ in $\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$
  - Test: $w^T z = \sum_i \alpha_i y_i x_i^T z$

(This will allow generalization to non-linear case)

Soft-margin classifiers

- What if the sets are not linearly separable ?
  - We want to minimize the number of errors (points on the wrong side of the margin)
  - We still want to maximize the margin
  - Solution: combine

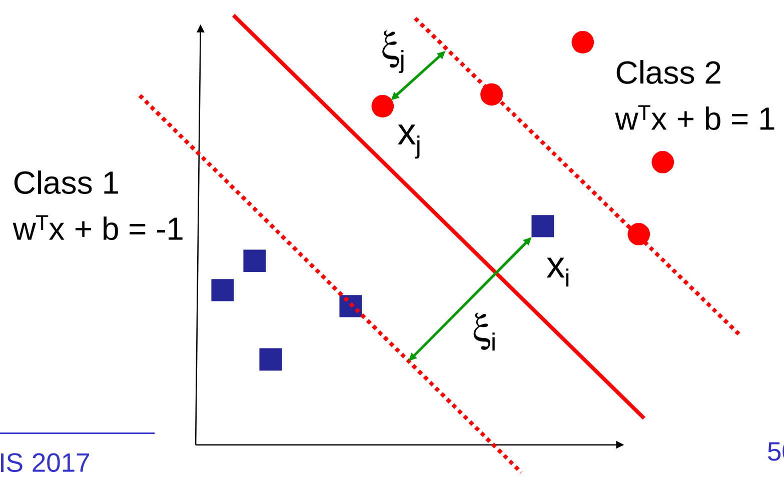
$$\min_{w}\left[\frac{1}{2}\|w\|^{2}+C\times\#\{errors\,on\,training\}\right]$$

  - But this is no longer a QP problem, so instead of counting the errors, we minimize their distance to the margin

$$\min_{w}\left[\frac{1}{2}\|w\|^{2}+C\times\sum_{i}\max\left(0,1-y_{i}\left(w^{T}x_{i}+b\right)\right)\right]$$

Soft-margin classifiers

- If the sets are not linearly separable, try to minimize the "error" ξ_i in classification



Soft-margin classifiers

- New constraints: $\begin{cases} w^T x_i + b \geq 1 - \xi_i & \text{if } y_i = 1 \\ w^T x_i + b \leq -1 + \xi_i & \text{if } y_i = -1 \\ \xi_i \geq 0 \end{cases}$

  ξ_i=0 if no error for x_i
- Maximize margin and minimize error:

$$\begin{cases} \min_{w,\xi_i} \left(\frac{1}{2} \|w\|^2 + C \sum_i \xi_i \right) \\ \text{constraints}: \forall i \quad y_i \left(w^T x_i + b \right) \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{cases}$$

  - C is a tradeoff parameter between error and margin

Soft-margin classifiers

- Lagrangian:

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 + \xi_i \right] - \sum_i \beta_i \xi_i$$

- We have to solve:

$$\min_{w,b,\xi} \max_{\alpha_i \geq 0, \beta_i \geq 0} \left\{ \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 + \xi_i \right] - \sum_i \beta_i \xi_i \right\}$$

- Which (under conditions) is equivalent to:

$$\max_{\alpha_i \geq 0, \beta_i \geq 0} \min_{w,b,\xi} \left\{ \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 + \xi_i \right] - \sum_i \beta_i \xi_i \right\}$$

Soft-margin classifiers

- Lagrangian:

$$L(w,b,\xi,\alpha,\beta)=\frac{1}{2}\|w\|^2+C\sum_i \xi_i-\sum_i \alpha_i\left[y_i\left(w^T x_i+b\right)-1+\xi_i\right]-\sum_i \beta_i\xi_i$$

- We have to solve:

$$\min_{w,b,\xi}\ \max_{\alpha_i\geq 0,\beta_i\geq 0}\left\{\frac{1}{2}\|w\|^2+C\sum_i \xi_i-\sum_i \alpha_i\left[y_i\left(w^T x_i+b\right)-1+\xi_i\right]-\sum_i \beta_i\xi_i\right\}$$

- Which (under conditions) is equivalent to:

$$\max_{\alpha_i\geq 0,\beta_i\geq 0}\ \min_{w,b,\xi}\left\{\frac{1}{2}\|w\|^2+C\sum_i \xi_i-\sum_i \alpha_i\left[y_i\left(w^T x_i+b\right)-1+\xi_i\right]-\sum_i \beta_i\xi_i\right\}$$

Soft-margin classifiers

- Step 1: set partial derivatives of L to zero

$$\begin{cases}\dfrac{\partial L(w,b,\xi,\alpha,\beta)}{\partial w_j}=w_j-\sum_i \alpha_i y_i x_{ij}=0\\[2ex] \dfrac{\partial L(w,b,\xi,\alpha,\beta)}{\partial b}=-\sum_i \alpha_i y_i=0\\[2ex] \dfrac{\partial L(w,b,\xi,\alpha,\beta)}{\partial \xi_j}=C-\alpha_j-\beta_j=0\end{cases}\qquad \text{so}:\begin{cases}w_j=\sum_i \alpha_i y_i x_{ij}\\ \sum_i \alpha_i y_i=0\\ \alpha_j=C-\beta_j\end{cases}$$

- Also with KKT conditions:

$$\alpha_i\left[y_i\left(w^T x_i+b\right)-1+\xi_i\right]=0$$

$$\beta_i\xi_i=0$$

Soft-margin classifiers

- Step 2: replace in L

$$L(w,b,\xi,\alpha,\beta)=\frac{1}{2}\|w\|^2+C\sum_i \xi_i-\sum_i \alpha_i\left[y_i\left(w^T x_i+b\right)-1+\xi_i\right]-\sum_i \beta_i\xi_i$$

$$=\frac{1}{2}\|w\|^2+\sum_i (C-\beta_i)\xi_i-\sum_i \alpha_i y_i w^T x_i-b\underbrace{\sum_i \alpha_i y_i}_{=0}+\sum_i \alpha_i-\sum_i \alpha_i\xi_i$$

=0

$$=\sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

- The dual problem is:

$$\begin{cases}\max_{\alpha} \sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^T x_j\\ \text{with } \sum_i \alpha_i y_i=0,\quad C\geq\alpha_i\geq 0\end{cases}$$

Soft-margin classifiers

- The dual problem for soft margin is identical to the one for hard margin, with the exception of the upper bound C on α :

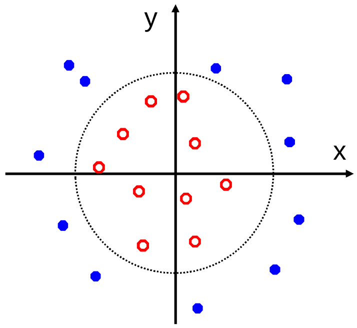
$$\begin{cases}\max_{\alpha} \sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^T x_j\\ \text{with } \sum_i \alpha_i y_i=0,\quad C\geq\alpha_i\geq 0\end{cases}$$

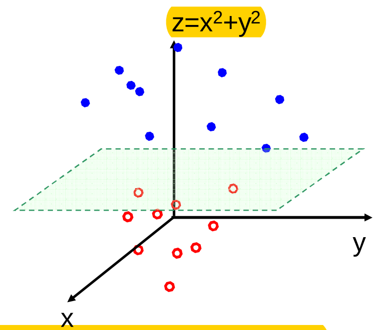
- When solved in α, w is again computed as

$$w=\sum_i \alpha_i y_i x_i$$

Non-linear decision boundary

- Sometimes the decision boundary is not linear

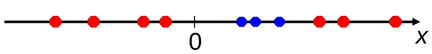




- But it may become if the set is projected in a higher dimension space

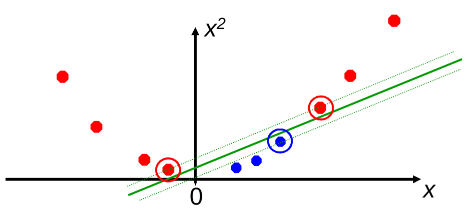
Non-linear decision boundary

- Idea: move data into a higher dimension space and search for a linear separator
- 1D problem:



not linearly separable

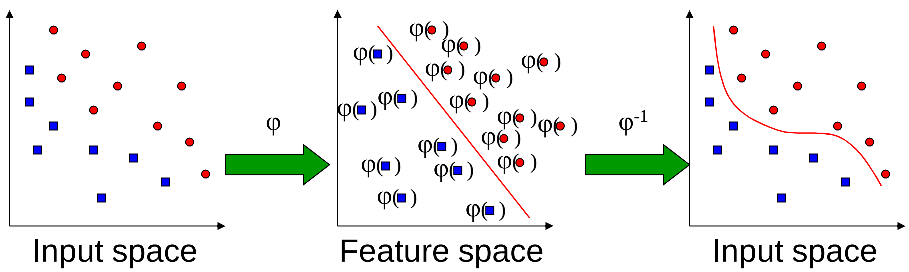
- 1D-2D transform: $x \rightarrow \varphi(x) = (x, x^2)$
- 2D problem:



linearly separable

Non-linear decision boundary

- Idea: define a transform φ from input space to feature space: $x \to \varphi(x)$



  - Solve a linear problem in the feature space
  - This gives a non-linear classifier in the input space

Kernel trick

- Remember: to solve the dual problem, we only need to know the scalar product $x_i^T x_j$
- In the feature space, we only need to know

$$K(x_i,x_j) = \varphi(x_i)^T \varphi(x_j)$$

  - K is a kernel, a similarity measure between x_i and x_j
- If we define K, when can we be sure that K is a kernel (that a suitable φ exist)?
- Mercer's theorem: every Positive Semi-Definite symmetric function K is a kernel
  - (PSD = no negative eigenvalues)
- Kernel Trick: we never have to compute φ

Kernel trick

- The dual problem becomes:

$$\begin{cases} \max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{with } \sum_i \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \end{cases}$$

- Solving provides the values for α_i
- Weight vector w is:

$$w = \sum_i \alpha_i y_i \varphi(x_i)$$

  - But we don't need to know w, just to be able to compute $w^T x$

Kernel trick

- Decision function:

$$\begin{aligned} f(x) &= w^T \varphi(x) + b \\ &= \sum_i \alpha_i y_i \varphi(x_i)^T \varphi(x) + b \\ &= \sum_i \alpha_i y_i K(x_i, x) + b \end{aligned}$$

- No need to know φ
- New data point z can be classified by:

$$\sum_i \alpha_i y_i K(x_i, z) + b < 0 \qquad \text{class1}$$

$$\sum_i \alpha_i y_i K(x_i, z) + b > 0 \qquad \text{class2}$$

Kernel Examples

- $x = (x_1, x_2)$
- $K(x_i, x_j) = (1 + x_i^T x_j)^2$

  $= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$
- We can check that:

  $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$

with:

$$\varphi(x) = (1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2)$$

- This defines a linear classifier in a feature space of dim 6, which corresponds to a non-linear classifier in the input space of dim 2

Kernel Examples

- d-th degree polynomial:

$$K(x, x') = (1 + x^T x')^d$$

If the input space is dim n, the feature space is dim $\binom{n+d}{d}$

- Radial basis:

$$K(x, x') = \exp(-\frac{1}{2\sigma^2}\|x - x'\|^2)$$

The feature space is of infinite dimension

- Sigmoid with parameters α and β:

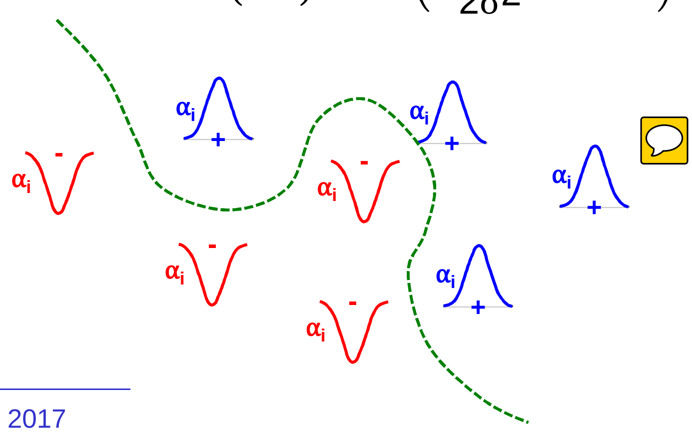
$$K(x, x') = \tanh(\alpha x^T x' + \beta)$$

Kernel Examples

- A linear combination of kernels is a kernel
- The product of two kernels is a kernel
- Kernel can be defined for other objects than vectors:
  - Trees
  - Strings
  - …

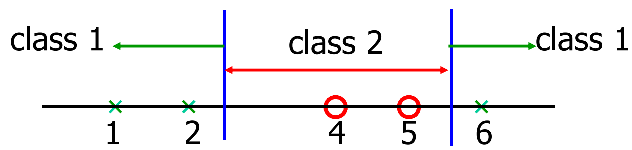
Kernel Example: RBF

- Decision function: $f(x)=\sum_i \alpha_i y_i K(x_i,x)+b$
- RBF kernel: $K\left(x,x'\right)=\exp\left(-\frac{1}{2\sigma^2}\|x-x'\|^2\right)$



SVM Example 1

- Suppose we have 5 data points in 1D space:
  - Class 1: $x_1=1$, $x_2=2$, $x_5=6$,
  - Class 2: $x_3=4$, $x_4=5$,
  - So $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$



- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100

SVM Example 1

- We need to find α_i (i=1, …, 5) :

$$\begin{cases} \max\limits_{\alpha} \sum_{i=1}^{5} \alpha_i - \frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{5} \alpha_i\alpha_j y_i y_j (x_i x_j + 1)^2 \\ \text{with} \quad \sum_{i=1}^{5} \alpha_i y_i = 0, \quad 100 \geq \alpha_i \geq 0 \end{cases}$$

- Using a QP solver, we get:
  - $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - Support vectors are $\{x_2=2, x_4=5, x_5=6\}$

SVM Example 1

- Discriminant function :

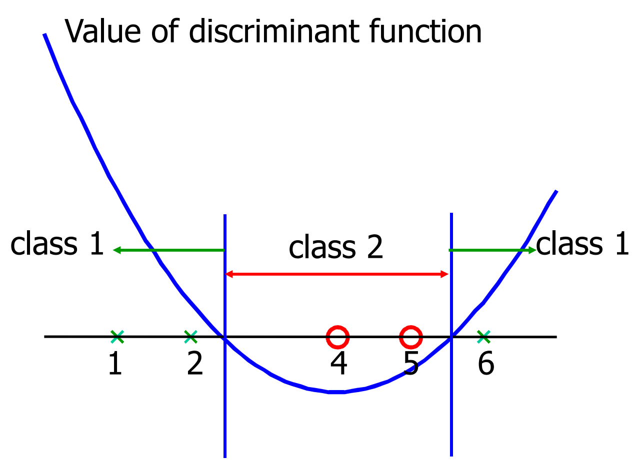
$$f(x) = \sum_i \alpha_i y_i K(x_i, x) + b$$

$$= 2.5(2x+1)^2 + 7.3(-1)(5x+1)^2 + 4.8(6x+1)^2 + b$$

- How to find b ?
  - Solve f(2)=1 or f(5)=-1 or f(6)=1
  - Result: b=9
- Discriminant function:

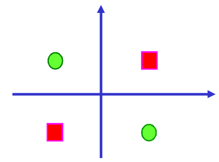
$$f(x) = 0.667x^2 - 5.333x + 9$$

SVM Example 1



SVM Example 2: XOR

- XOR:
  - Class 1: $x_1=(1,-1)$, $x_2=(-1,1)$
  - Class 2: $x_3=(1,1)$, $x_4=(-1,-1)$



- Kernel:
  - $K(x,y) = (x^T y + 1)^2 = (x_1 y_1 + x_2 y_2 + 1)^2$
  - Corresponds to:
    $\varphi(x) = (1, x_1^2, \sqrt{2}\, x_1 x_2, x_2^2, \sqrt{2}\, x_1, \sqrt{2}\, x_2)$
- QP problem:

$$\max_{\alpha} \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j (x_i^T x_j + 1)^2$$

$$\text{with } \alpha_i \geq 0, \ \sum_{i=1}^{4} \alpha_i y_i = \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$$

SVM Example 2: XOR

$$\max_{\alpha} \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \alpha^T H \alpha \quad \text{with } H = \begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix}$$

- Derivative:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix} \alpha = 0$$

- Solution: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$
  - Satisfies the constraints

SVM Example 2: XOR

- Weight vector:

$$w = \sum_{i=1}^{4} \alpha_i y_i \varphi(x_i) = \frac{1}{4}\left(\varphi(x_1) + \varphi(x_2) - \varphi(x_3) - \varphi(x_4)\right)$$

$$= \begin{bmatrix} 0 & 0 & -\sqrt{2} & 0 & 0 & 0 \end{bmatrix}$$

- Discriminant function:

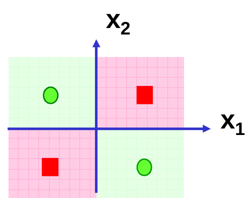
$$f(x) = w^T \varphi(x) + b = -2x_1x_2 + b$$

  - With b = 0 because of support vectors

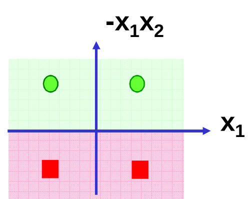
$$f(x) = -2x_1x_2$$

SVM Example 2: XOR

- $f(x) = -2x_1x_2$



Non-linear boundaries

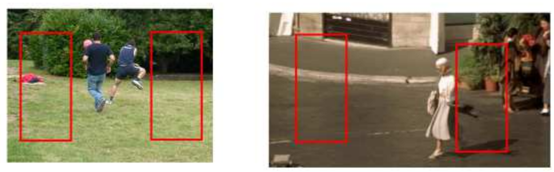


Linear boundaries

SVM Example 3: Pedestrian detection

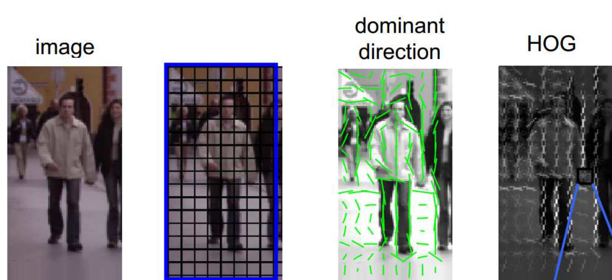
- Positive examples

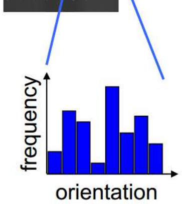
- Negative data



SVM Example 3: Pedestrian detection

- Histogram of Oriented Gradients





16 x 8 cells x 8 orientations
Feature dimension: 1024

SVM Example 3: Pedestrian detection

- SVM Training:
  - Training examples: $\{x_i, y_i\}$ $x_i \in \mathbb{R}^{1024}$ $y_i \in \{-1,+1\}$
  - Positive: $y_i=+1$, negative $y_i=-1$
  - Linear SVM: $f(x) = w.x + b$
  - Training algorithm: $\rightarrow$ values for w and b
- Detection: sliding window

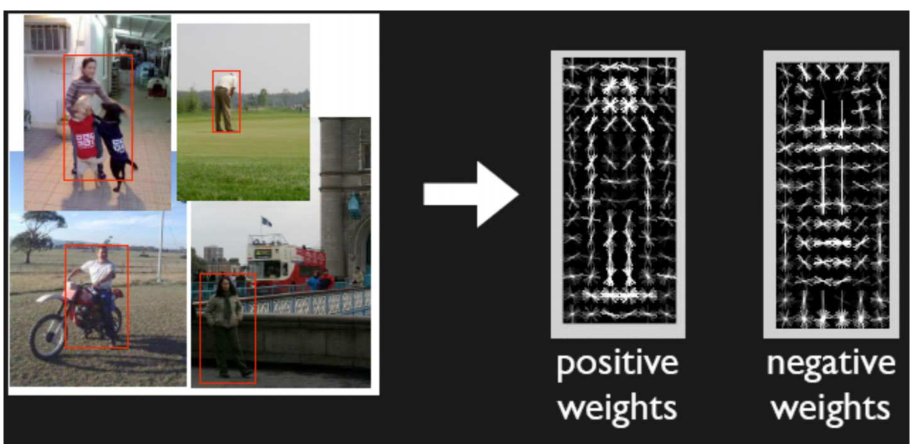
window $\rightarrow x \in \mathbb{R}^{1024}$

if $f(x) = w.x + b > 0$: person detected

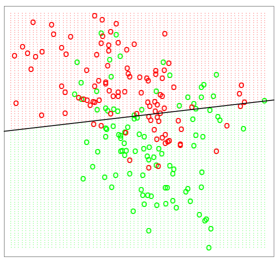
if $f(x) = w.x + b < 0$: no person detected

SVM Example 3: Pedestrian detection

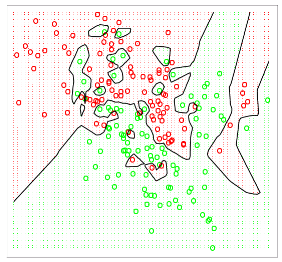
$f(x) = w.x + b$



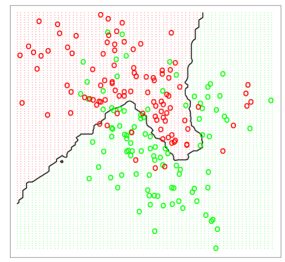
SVM: classification examples



Linear



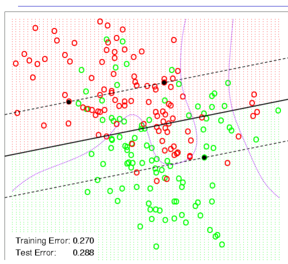
1-NN



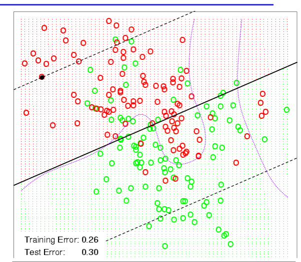
15-NN

SVM: classification examples

Linear kernel

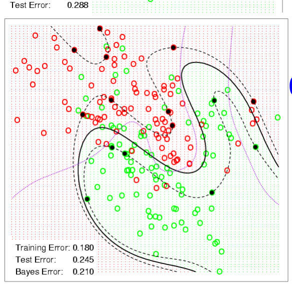


C=10000

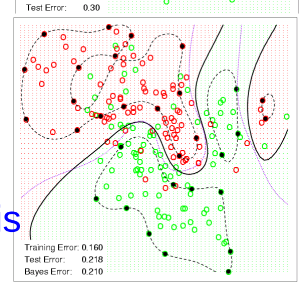


C=0.01

d°4 polynomial



Radial Basis

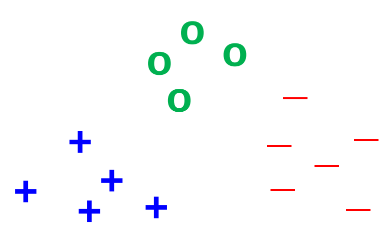


SVM Issues

- Choice of kernel
  - Radial Basis or polynomial are standard
  - For very large problems: linear kernel
    - Large dimension
    - Large number of samples
- Choice of hyper-parameters C (+ Kernel par.)
  - By grid search:
    - Train SVM for several values
    - Keep values with best performance on validation data

SVM Multi-class classification

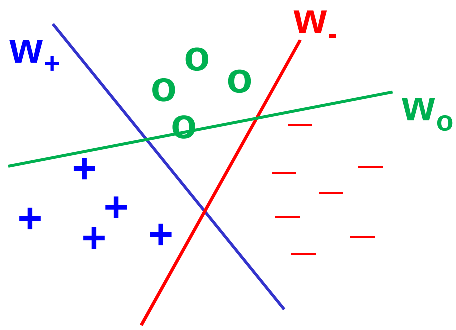
- SVM is a binary classifier



  - No easy large-margin generalization
  - 2 common approaches:
    - 1 versus all
    - 1 versus 1

SVM Multi-class classification

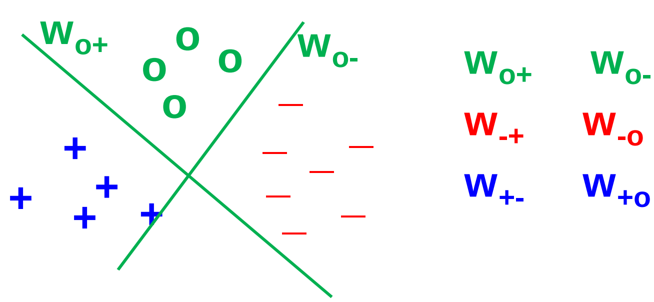
- 1 versus all (requires N classifiers)



  - Build one classifier for each class
  - Assign to class with largest score

SVM Multi-class classification

- 1 versus 1 (requires N(N-1)/2 classifiers)



  - Build one classifier for each pair of classes
  - Assign to class by majority vote

SVM Summary

Margin	Primal	Dual
Hard	$\begin{cases} \min_w \frac{1}{2}\Vert w\Vert^2 \\ with:\ \forall i\ \ y_i(w^T x_i + b) \geq 1 \end{cases}$	$\begin{cases} \max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \\ with: \sum_i \alpha_i y_i = 0\ and\ \ \alpha_i \geq 0 \end{cases}$ $then:\ w = \sum_i \alpha_i y_i\, x_i$
Soft	$\min_w \begin{cases} \frac{1}{2}\Vert w\Vert^2 + \\ C.\sum_i max\left(0{,}1 - y_i(w^T x_i + b)\right) \end{cases}$ C is an hyper-parameter	$\begin{cases} \max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \\ with: \sum_i \alpha_i y_i = 0\ and\ \ C \geq \alpha_i \geq 0 \end{cases}$ $then:\ w = \sum_i \alpha_i y_i\, x_i$

SVM Summary

- With kernel $K(x, x')$:
  - Dual: $\begin{cases} \max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ with: \sum_i \alpha_i y_i = 0\ and\ \ C \geq \alpha_i \geq 0 \end{cases}$
- Classification:

$$f(x) = \sum_i \alpha_i y_i K(x_i, x) + b > 0 \qquad class\ 1$$

$$f(x) = \sum_i \alpha_i y_i K(x_i, x) + b < 0 \qquad class\ 2$$

SVM vs NN

- Optimization:
  - SVM: quadratic programming (efficient, convex, global optimum)
  - NN: non-linear optimization (local optimum)
- Feature map:
  - SVM: via kernel
  - NN: via hidden layer neurons
- Generalization capability:
  - SVM: based on largest margin principle
  - NN: try and test

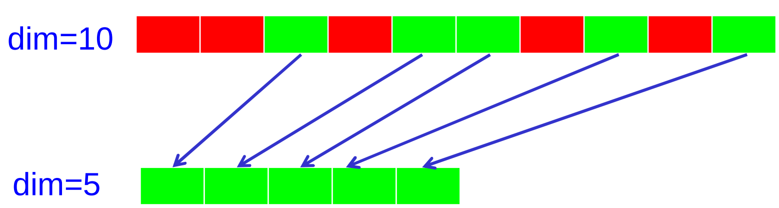
Dimensionality Reduction

Dimensionality Reduction

- We often handle vectors in $\mathbb{R}^d$ with d large
  - Requires lot of storage
  - Requires lot of computation
  - Requires models with lot of parameters
- But sometimes the true dimensionality of the data is smaller than d
- So it is useful to reduce dimension:
  - We save storage
  - We save computation
  - We can use models with less parameters, easier to estimate
- We can reduce the dimensionality exactly, but also approximately

Dimensionality Reduction

- The first idea is Feature Selection
  - Select a subset of the dimensions



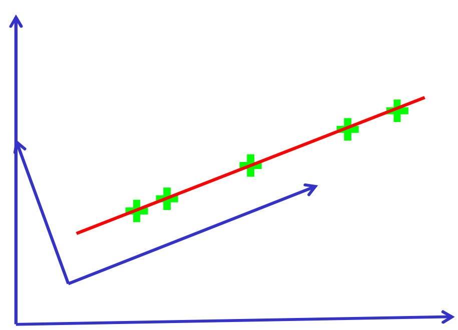
- Two possible strategies (generally supervised):
  - Uni-variate: independent selections, fast but ignore correlations
  - Multi-variate: subset selection, more accurate but computationally expensive

Dimensionality Reduction

- Uni-variate selection
  - Several criteria, for example:
  - Mutual Information $I(Y; x_i)$
  - Correlation between Y et x_i
- Multi-variate selection
  - Several strategies, for example:
  - Greedy growing: add one dimension with high correlation with Y and low correlation with subset
  - Backward reduction: remove dimension one by one
  - Floating search: start with subset of right size, replace one dimension by a better one

Dimensionality Reduction

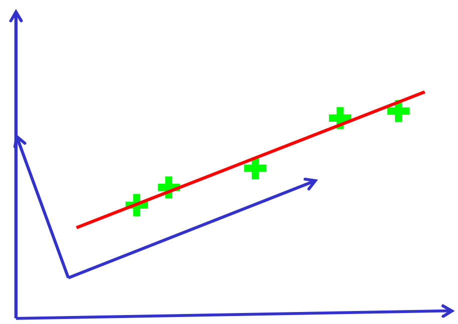
- Sometimes the data lies in a sub-space



- If we rotate the axes, one dimension would be enough

Dimensionality Reduction

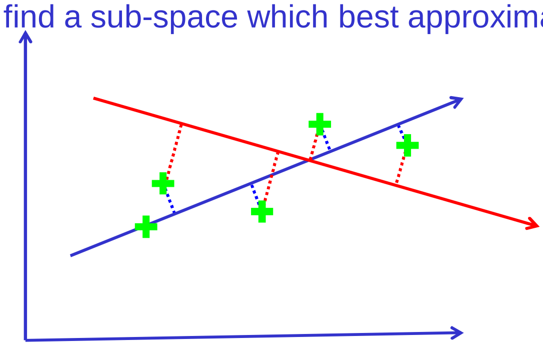
- Sometimes the data almost lies in a sub-space



- If we rotate the axes, one component would provide a good approximation of the data

Dimensionality Reduction - PCA

- PCA Principal Component Analysis
  - Idea: find a sub-space which best approximates the data



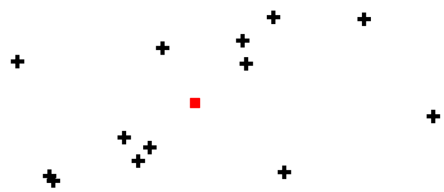
- Choose the sub-space which minimizes the distortion with the data

Dimensionality Reduction - PCA

- Approximation of dimension 0:

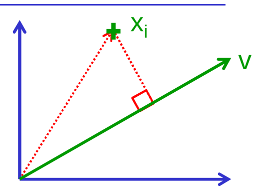
$$\underset{\mu \in \mathbb{R}^d}{\operatorname{argmin}} \sum_i \|x_i - \mu\|^2 = \frac{1}{|X|} \sum_i x_i$$

- Center dataset: $x_i \rightarrow x_i - \mu$



Dimensionality Reduction - PCA

- Approximation of dimension 1:



$\|x_i\|^2 = (v.x_i)^2 + d(x_i,v)^2$

$\sum_i \|x_i\|^2 = \sum_i (v.x_i)^2 + \sum_i d(x_i,v)^2$

$\underset{v}{\operatorname{argmin}} \sum_i d(x_i,v)^2 = \underset{v}{\operatorname{argmax}} \sum_i (v.x_i)^2$

- X matrix nxd, X^TX is symetric, positive, so diagonalizable:

$X^TX = U^TDU$ with $U^TU = UU^T = I$, D diagonal

$\sum_i (v.x_i)^2 = (Xv)^TXv = v^TX^TXv = v^TU^TDUv = (Uv)^TDUv$

Dimensionality Reduction - PCA

If D $= diag(\lambda_1, \lambda_2, \ldots \lambda_d)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$

- $\sum_i (\mathrm{v.x}_i)^2 = (\mathrm{Uv})^T \mathrm{DUv} = \sum_j \lambda_j {\mathrm{u}_j}^2$

is maximum when Uv=(1,0,0,…,0)

- v is the eigenvector of X^TX for the largest eigenvalue
- The sub-space of dimension 1 which best approximates X is the first principal axis
- The sub-space of dimension k which best approximates X is defined by the k first principal axes

Dimensionality Reduction - PCA

- How to use PCA ?
  - Compute covariance matrix X^TX
  - Compute eigenvectors u_i for k largest eigenvalues
  - Project on sub-space:

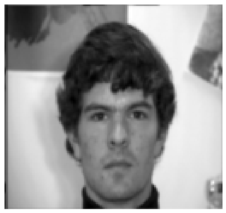
$$(x.u_1, x.u_2, \ldots, x.u_k)$$

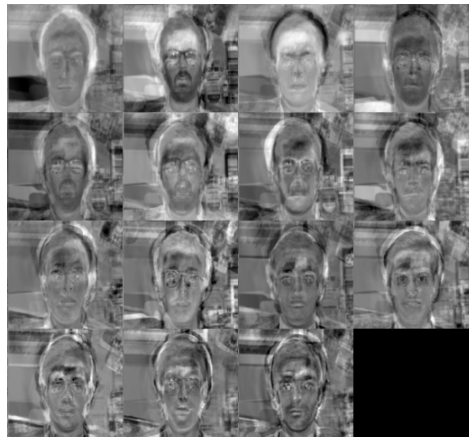
- How to choose k ?
  - $\|X\|_2^2 = \sum_j \lambda_j$ is the total energy (or variance, information, volume)
  - Choose the smallest k such that:

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j} \geq \theta$$

Dimensionality Reduction - PCA

- Example: EigenFaces [Turk, Pentland 1991]
  - Face database





- Represented with 15 eigenvectors

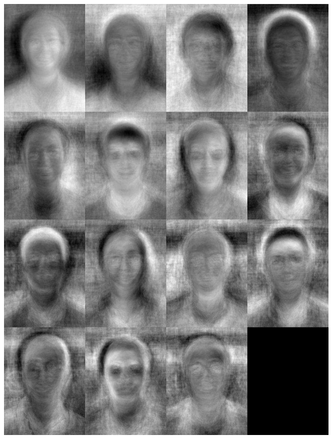
Dimensionality Reduction - PCA

- Example: EigenFaces [Turk, Pentland 1991]

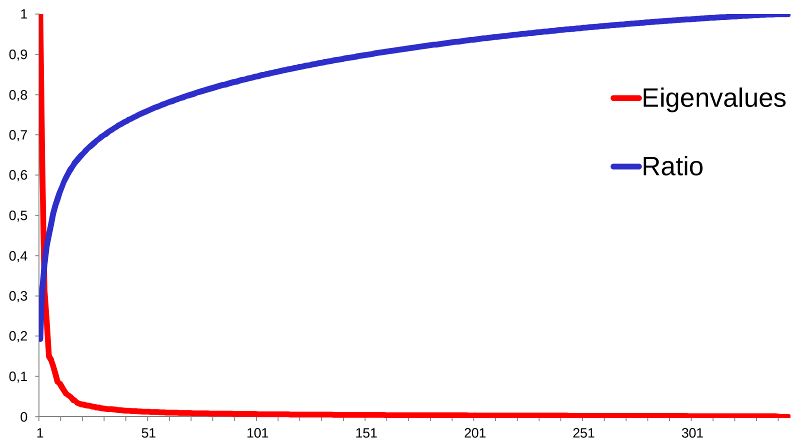
Dimensionality Reduction - PCA

- Mean vector

Eigenfaces



Dimensionality Reduction - PCA



Dimensionality Reduction - PCA

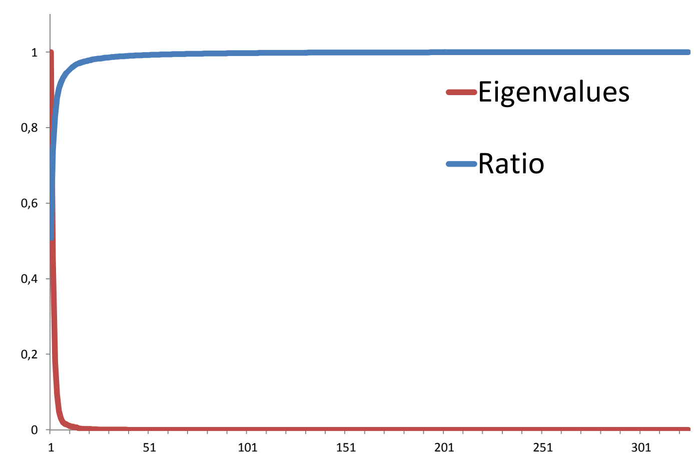
- Example: Stock market
  - 357 stocks, 253 values (full year)

FR0000077687	60,00	61,00	61,40	61,31	61,45	61,90	61,95	62,20	62,50	62,35	62,00	62,85	63,15	63,30	63,51	66,85	66,50	65,80	65,30
FR0000120404	26,93	26,93	26,76	27,19	27,19	27,03	26,73	26,90	26,91	26,89	26,94	27,03	27,70	28,35	28,26	28,90	28,31	28,62	29,00
FR0004026151	2,56	2,55	2,55	2,55	2,51	2,45	2,43	2,42	2,50	2,57	2,56	2,56	2,50	2,51	2,44	2,42	2,57	2,57	2,50
FR0000120859	48,48	49,65	49,55	50,04	48,78	49,04	49,05	48,55	48,80	48,32	47,23	47,26	49,45	48,94	49,88	48,76	49,93	49,86	49,74
FR0010636555	61,49	62,48	63,28	63,34	63,42	63,96	63,97	63,57	63,29	64,24	64,87	67,25	67,32	67,05	67,14	67,32	67,47	66,75	64,55
FR0000001703	37,40	37,76	37,77	38,43	38,18	38,43	38,90	39,38	39,18	38,55	38,80	38,63	38,81	38,88	38,79	38,70	38,29	38,70	38,85
FR0000131906	41,14	40,53	41,14	40,67	40,39	39,92	40,14	40,49	40,23	40,76	40,75	40,78	41,96	43,68	43,33	43,08	42,30	42,88	43,98
FR0000121501	5,58	5,51	5,66	6,22	6,54	6,20	6,39	6,20	6,14	6,42	6,44	6,36	6,55	6,28	6,24	6,23	6,18	6,15	6,19
FR0010417345	8,20	8,20	8,20	8,35	8,35	8,40	8,31	8,40	8,24	8,30	8,33	8,40	8,35	8,40	8,40	8,26	8,20	8,30	8,20
FR0000125684	84,70	85,39	84,94	85,54	84,99	84,77	85,69	83,40	81,88	81,04	82,86	81,83	80,51	82,49	82,01	81,79	81,89	81,57	81,47
FR0004044600	3,49	3,59	4,20	4,50	4,33	4,30	4,24	4,20	4,25	4,15	4,15	4,00	4,00	4,00	4,15	4,14	4,14	4,10	4,14
FR0000061244	2,75	2,74	2,74	2,75	2,70	2,70	2,67	2,67	2,67	2,75	2,74	2,74	2,75	2,75	2,74	2,74	2,74	2,75	2,73
FR0010240994	12,29	12,37	12,37	12,35	12,37	12,60	13,28	13,44	13,37	13,32	13,45	13,60	13,80	13,80	13,91	14,08	14,13	14,28	14,00
FR0000185514	12,06	12,45	12,55	12,48	12,52	12,75	12,98	13,20	13,38	13,49	13,54	13,64	13,74	14,25	14,47	14,70	14,60	14,54	14,65
FR0000076887	3,16	3,17	3,28	3,26	3,29	3,27	3,27	3,20	3,15	3,16	3,15	3,18	3,17	3,12	3,16	3,16	3,17	3,15	3,14
FR0000035370	7,37	7,79	7,94	7,98	7,90	7,87	7,90	8,01	8,30	8,75	8,70	8,85	8,83	8,79	8,70	8,65	8,64	8,50	8,45
FR0004029411	1,16	1,15	1,15	1,16	1,17	1,17	1,17	1,17	1,17	1,19	1,18	1,18	1,19	1,18	1,27	1,27	1,26	1,30	1,27
FR0000062796	20,77	20,84	20,61	20,65	20,35	20,21	20,40	20,35	20,21	20,60	20,41	20,87	21,68	21,50	21,80	22,00	22,00	21,80	22,00
FR0000031684	17,30	17,15	17,16	17,18	17,15	17,10	17,15	17,15	17,14	17,00	17,00	17,10	16,99	17,22	17,09	17,10	17,11	17,10	17,27
FR0000077570	1,02	1,00	1,00	1,01	0,99	1,00	1,00	1,01	1,03	1,02	1,00	1,00	1,00	1,01	1,01	1,00	1,00	1,06	1,03
FR0004529147	79,97	81,00	79,54	79,00	80,00	80,61	81,80	82,50	82,30	81,50	82,21	82,68	80,01	80,00	80,13	81,30	82,00	81,95	82,70
FR0004036036	18,02	18,02	17,90	17,34	17,70	17,40	17,30	17,30	17,21	16,92	17,00	16,90	17,26	17,11	17,26	17,31	17,69	17,80	17,52
FR0010298620	2,25	2,43	2,55	2,40	2,45	2,56	2,79	2,78	2,83	2,60	2,55	2,56	2,55	2,52	2,61	2,55	2,45	2,37	2,35
FR0007056841	99,60	102,28	103,55	103,58	102,59	102,84	103,36	102,22	101,84	101,68	102,40	102,19	102,50	103,22	102,95	104,14	104,39	104,02	104,16
FR0007052782	37,00	37,12	36,98	37,19	36,87	37,16	37,04	37,08	37,13	36,92	36,87	37,05	37,46	37,46	37,55	37,40	37,10	37,39	37,72
FR0004180578	12,56	12,60	12,51	12,67	12,97	12,99	13,30	13,16	13,20	13,10	13,20	13,24	13,60	13,90	13,87	13,87	13,85	13,89	13,85
FR0004180537	24,16	24,40	23,70	23,00	23,33	23,27	23,11	23,00	23,15	23,30	23,30	23,40	23,60	24,10	24,50	24,80	24,21	24,15	24,61
FR0000074122	4,00	3,99	3,99	3,94	3,99	4,00	4,02	4,03	4,04	4,15	4,17	4,20	4,18	4,20	4,19	4,20	4,15	4,14	4,12

...

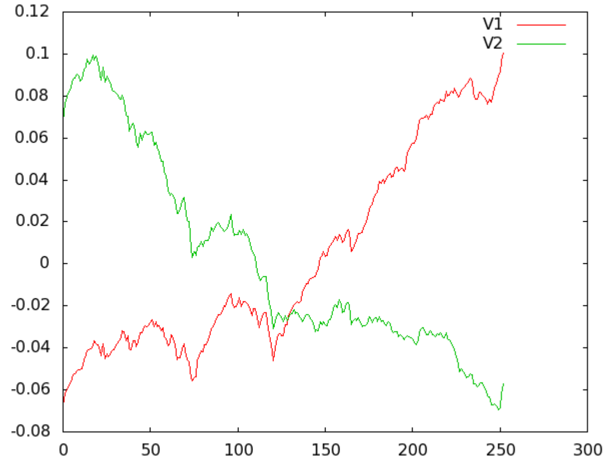
...

Dimensionality Reduction - PCA



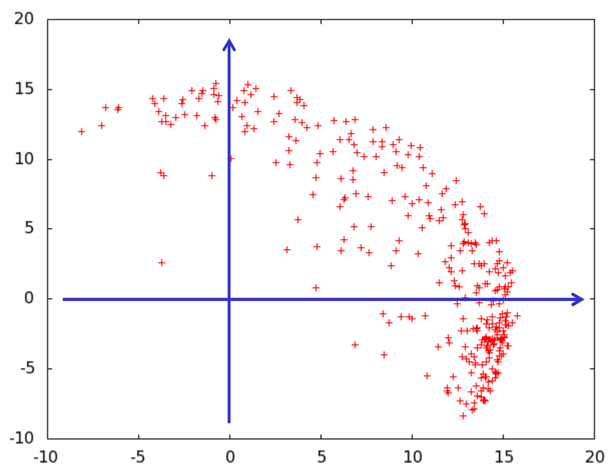
Dimensionality Reduction - PCA

- First two eigenvectors



Dimensionality Reduction - PCA

- Projection on first two principal axes



Decision Trees

Decision Trees

- Powerful mechanism to analyse large amounts of data:
  - To describe the main characteristics of the data
  - To classify data into disjoints classes
  - To generalize for predicting new data
- Widely used in data mining applications, because of the possibility to extract human-readable rules
- Now a basic technique in Machine Learning

Decision tree applications

- Existing applications:
  - Financial: fraud detection
  - Language: statistical parser
  - Medicine: automated diagnosis
  - Biology: Genome analysis
  - Pharmacology: classification of drugs
  - Games
  - …

MasterMind Game

Identification Game

- Guess who is X by asking questions

Name	Gender	Size	Eyes	Hair
Arthur	*Male*	*Tall*	*Blue*	*Black*
Bob	*Male*	*Medium*	*Green*	*Black*
Charlie	*Male*	*Short*	*Brown*	*Red*
Denis	*Male*	*Tall*	*Brown*	*Blond*
Edmond	*Male*	*Medium*	*Brown*	*Black*
Frida	*Female*	*Medium*	*Green*	*Blond*
Georgia	*Female*	*Short*	*Blue*	*Blond*
Helen	*Female*	*Short*	*Blue*	*Blond*
Irene	*Female*	*Tall*	*Green*	*Black*
Jane	*Female*	*Medium*	*Brown*	*Black*

Identification Game

- Who is X ?

Name	Gender	Size	Eyes	Hair
Arthur	?	?	?	?
Bob	?	?	?	?
Charlie	?	?	?	?
Denis	?	?	?	?
Edmond	?	?	?	?
Frida	?	?	?	?
Georgia	?	?	?	?
Helen	?	?	?	?
Irene	?	?	?	?
Jane	?	?	?	?

Identification Game

- What is X’s gender ? *X is male*

Name	Gender	Size	Eyes	Hair
Arthur	***Male***	?	?	?
Bob	***Male***	?	?	?
Charlie	***Male***	?	?	?
Denis	***Male***	?	?	?
Edmond	***Male***	?	?	?
Frida	*Female*	?	?	?
Georgia	*Female*	?	?	?
Helen	*Female*	?	?	?
Irene	*Female*	?	?	?
Jane	*Female*	?	?	?

Identification Game

- What is X’s size ? *X is tall*

Name	Gender	Size	Eyes	Hair
Arthur	***Male***	***Tall***	?	?
Bob	*Male*	*Medium*	?	?
Charlie	*Male*	*Short*	?	?
Denis	***Male***	***Tall***	?	?
Edmond	*Male*	*Medium*	?	?
Frida	*Female*	*Medium*	?	?
Georgia	*Female*	*Short*	?	?
Helen	*Female*	*Short*	?	?
Irene	*Female*	*Tall*	?	?
Jane	*Female*	*Medium*	?	?

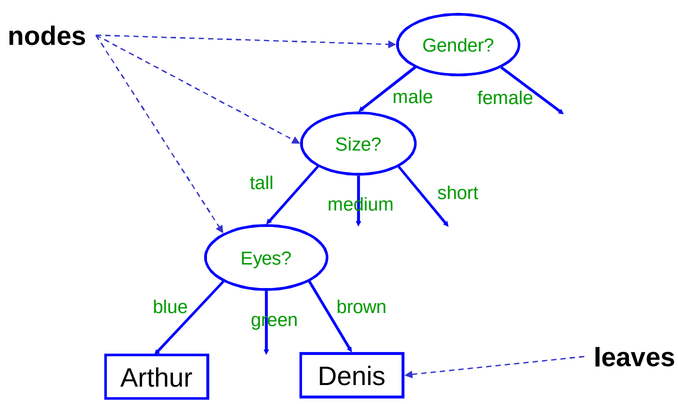
Identification Game

- What is X's eyes color ?   *X has blue eyes*

Name	Gender	Size	Eyes	Hair
Arthur	***Male***	***Tall***	***Blue***	?
Bob	*Male*	*Medium*	*Green*	?
Charlie	*Male*	*Short*	*Brown*	?
Denis	*Male*	*Tall*	*Brown*	?
Edmond	*Male*	*Medium*	*Brown*	?
Frida	*Female*	*Medium*	*Green*	?
Georgia	*Female*	*Short*	*Blue*	?
Helen	*Female*	*Short*	*Blue*	?
Irene	*Female*	*Tall*	*Green*	?
Jane	*Female*	*Medium*	*Brown*	?

Decision Tree Solution

- Game: questions to identify someone ?



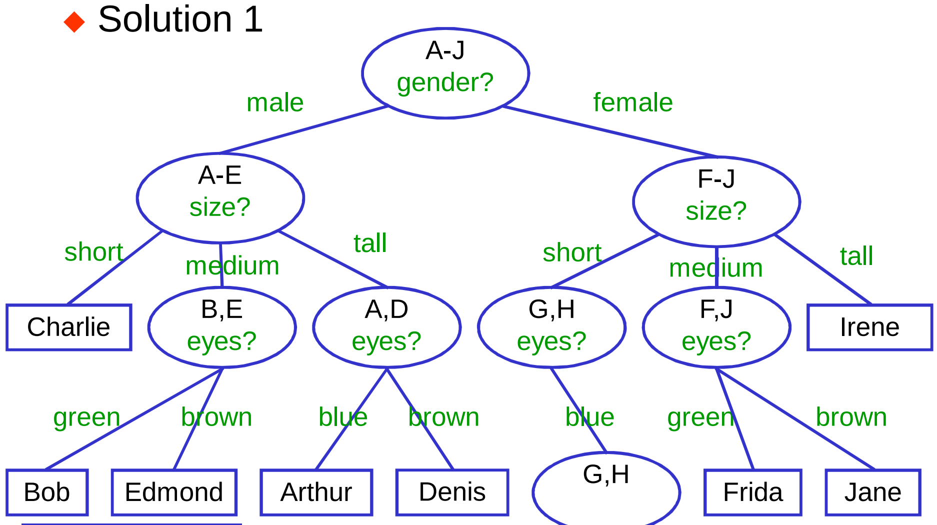
Identification Game

- What to ask for first ?

Name	Gender	Size	Eyes	Hair
Arthur	*Male*	*Tall*	?	?
Bob	*Male*	*Medium*	?	?
Charlie	*Male*	*Short*	?	?
Denis	*Male*	*Tall*	?	?
Edmond	*Male*	*Medium*	?	?
Frida	*Female*	*Medium*	?	?
Georgia	*Female*	*Short*	?	?
Helen	*Female*	*Short*	?	?
Irene	*Female*	*Tall*	?	?
Jane	*Female*	*Medium*	?	?

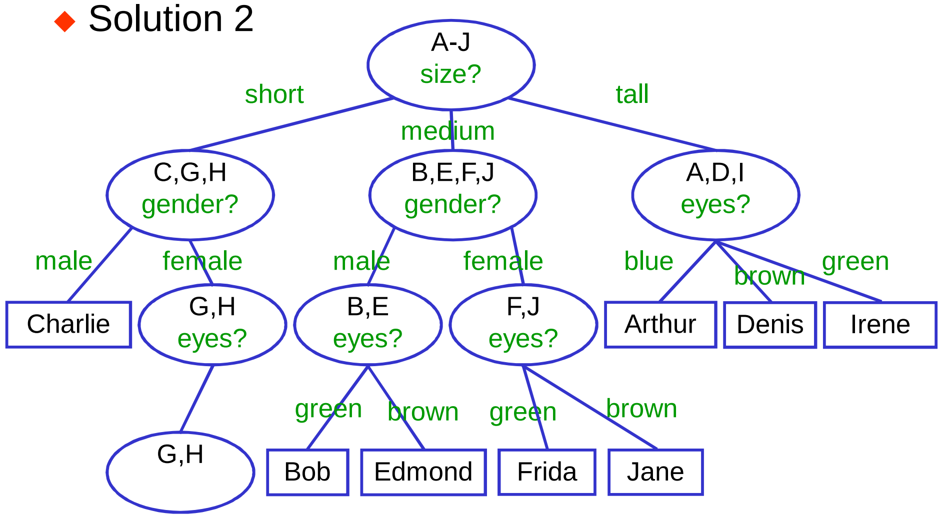
Identification Game

- Solution 1



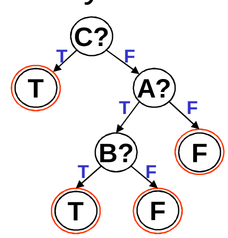
Identification Game

- Solution 2

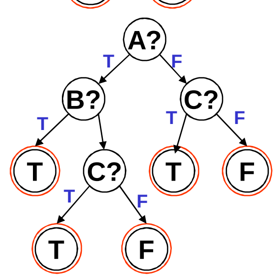


Decision Trees: Expressive Power

- Decision trees can implement any boolean function:
  - Example
    $[A(x) \wedge B(x)] \vee C(x)$

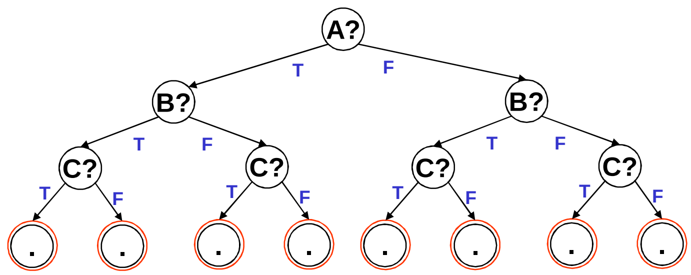


- Several trees implement the same function
  - Logical formulas with the same value table



Decision Trees: Expressive Power

- Decision trees can implement any boolean function:



  - But may need an exponential number of nodes
    n predicates $\rightarrow O(2^n)$ nodes

Decision Trees: Efficiency

- Some trees are more efficient than others
  - Number of questions to ask before getting result
  - We should prefer trees with minimum mean depth
  - This means enumerating a lot of possible trees to find the best one
- Unfortunately, the problem of finding the best tree is NP-complete
  - We have to use heuristics to try to find “good” trees rather than the best one

Decision Trees: Efficiency

- How many distinct decision trees with n binary attributes ?
  - It is the number of boolean functions = 2^{2^n}

1	2	4
2	4	16
3	8	256
4	16	65536
5	32	4,29E+09
6	64	1,84E+19

  - And this does not count trees with different structures and same output…

Decision Trees Construction

- We assume that we have a lot of data
  - Real world data
  - It may not contain all cases
  - There might be no simple model

- We want to construct a good decision tree to model this data
  - Efficient, but not necessarily perfect
    - Performance on test data
  - Not too expensive to build
    - Small tree: number of nodes, depth

Probabilistic Formalisation

- Random variables: $X_1, X_2, \ldots X_n, Y$
  - Finite values (for now)
  - possibly non-comparable: (X_1 = color, Y=name)
  - X_i are the attributes
  - Y is the class to predict
- Goal:
  - Predict the value of Y based on the values of $X_1, X_2, \ldots X_n$
- Approach:
  - Use training examples to estimate probabilities

$$(x^j_1, x^j_2, \ldots x^j_n, y^j) \qquad j = 1,2\ldots T$$

Identification game

Y	X_1	X_2	X_3	X_4	
Name	**Gender**	**Size**	**Eyes**	**Hair**	
Arthur	Male	Tall	Blue	Black	j=1
Bob	Male	Medium	Green	Black	j=2
Charlie	Male	Short	Brown	Red	
Denis	Male	Tall	Brown	Blond	
Edmond	Male	Medium	Brown	Black	
Frida	Female	Medium	Green	Blond	
Georgia	Female	Short	Blue	Blond	
Helen	Female	Short	Blue	Blond	
Irene	Female	Tall	Green	Black	j=T=10
Jane	Female	Medium	Brown	Black	

MALIS Grades

- MALIS started in 2015 (previously IS)
- 190 students passed the exam
  - Y = grade(MALIS)
- They also passed other exams
  - X_i = grade($course_i$)
- Goal: predict Y=grade(MALIS) from X_i
- Grades are quantized:

  A for grade ≥ 17 | B for grade ≥ 13
  C for grade ≥ 10 | D for grade ≥ 5
  E for grade > 0 | F for grade = 0
  NA: student did not attend

MALIS Grades

MALIS	Clouds	SoftDev	Manag Intro	MMIR	ADST	Optim	WebSem	SecAppli	…
E	NA	F	C	E	NA	NA	NA	NA	…
B	B	A	NA	NA	NA	NA	NA	NA	…
B	A	A	NA	NA	NA	NA	NA	NA	…
D	A	A	NA	NA	NA	NA	NA	NA	…
B	D	NA	NA	NA	A	NA	B	B	…
F	NA	NA	NA	NA	NA	NA	NA	NA	…
F	NA	NA	NA	NA	NA	F	NA	NA	…
C	A	A	NA	NA	A	NA	NA	NA	…
C	A	A	B	NA	NA	NA	NA	NA	…
C	NA	NA	NA	NA	B	NA	NA	C	…
B	B	B	NA	NA	B	NA	NA	NA	…
B	NA	NA	NA	NA	NA	NA	NA	NA	…
C	B	NA	NA	B	C	C	NA	NA	…
C	A	A	A	NA	NA	NA	NA	NA	…
D	A	NA	B	D	NA	NA	NA	NA	…
D	B	NA	NA	B	NA	NA	NA	NA	…
…	…	…	…	…	…	…	…	…	…

Entropy

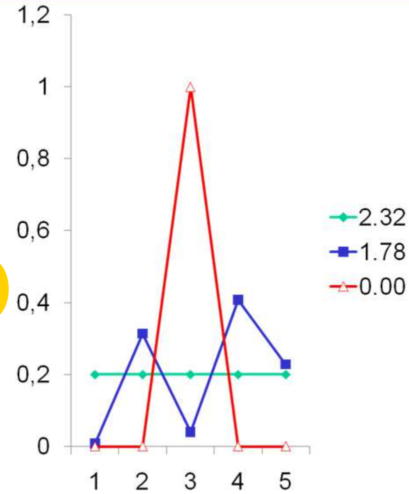
- Probability:
  - Count on training set:

$$p(Y=y)=\frac{1}{T}\mathrm{card}\{i : y^i = y\}$$

- Entropy:
  - Uncertainty on Y value:

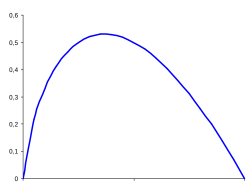
$$H(Y)=-\sum_y p(Y=y)\log_2 p(Y=y)$$

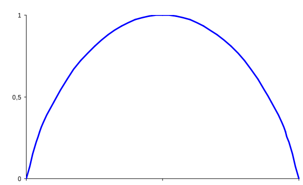
H(Y) maximal if p(Y) uniform, minimal if p(Y) is a Dirac distribution (value is certain)



Entropy

$F(t) = - t\, Log_2(t)$ $H(p,1-p)=-p.\log_2(p)-(1-p)\log_2(1-p)$





- $P(X=x) = 1/n$ $H(X) = \log_2(n)$
- $P(X=x) = \delta_a(x)$ $H(X) = 0$

Entropy: identification game

Name	Gender	Size	Eyes	Hair
Arthur	Male	Tall	Blue	Black
Bob	Male	Medium	Green	Black
Charlie	Male	Short	Brown	Red
Denis	Male	Tall	Brown	Blond
Edmond	Male	Medium	Brown	Black
Frida	Female	Medium	Green	Blond
Georgia	Female	Short	Blue	Blond
Helen	Female	Short	Blue	Blond
Irene	Female	Tall	Green	Black
Jane	Female	Medium	Brown	Black

$$p(Y = \text{Arthur}) = p(Y = \text{Bob}) = \ldots = \frac{1}{10}$$

$$H(\text{Name}) = -\sum_{y} \frac{1}{10} \log_2 \frac{1}{10} = \log_2 10 = 3.32$$

MALIS Grades: Entropy

- MALIS: 190 students:

Grade	A	B	C	D	E	F	NA
Nb students	13	70	70	20	6	11	0
Probability	0,07	0,37	0,37	0,11	0,03	0,06	0

- Entropy:

$$\begin{aligned} H(\text{MALIS}) = & -\frac{13}{190}\text{Log}_2\left(\frac{13}{190}\right) - \frac{70}{190}\text{Log}_2\left(\frac{70}{190}\right) - \frac{70}{190}\text{Log}_2\left(\frac{70}{190}\right) \\ & -\frac{20}{190}\text{Log}_2\left(\frac{20}{190}\right) - \frac{6}{190}\text{Log}_2\left(\frac{6}{190}\right) - \frac{11}{190}\text{Log}_2\left(\frac{11}{190}\right) \\ & -\frac{0}{190}\text{Log}_2\left(\frac{0}{190}\right) \\ & = 2.06 \end{aligned}$$

Conditional Entropy

- Assume we know that $X_1 = a_1$
  - Random variable $Y|X_1 = a_1$

$$p(Y = y|X_1 = a_1) = \frac{\mathrm{card}\{i : x^i{}_1 = a_1, y^i = y\}}{\mathrm{card}\{i : x^i{}_1 = a_1\}}$$

$$H(Y|X_1 = a_1) = -\sum_y p(Y = y|X_1 = a_1)\log_2 p(Y = y|X_1 = a_1)$$

- Example: 3 persons with blue eyes

$$H(\mathrm{Name}|\mathrm{Eyes} = \mathrm{blue}) = -\sum_y \frac{1}{3}\log_2\frac{1}{3} = \log_2 3$$

Conditional Entropy

- Assume that we know the value of X_1
  - average entropy $H(Y|X_1)$

$$\begin{aligned} H(Y|X_1) &= \sum_a p(X_1 = a)H(Y|X_1 = a) \\ &= -\sum_a p(X_1 = a)\sum_y p(Y = y|X_1 = a)\log_2 p(Y = y|X_1 = a) \\ &= -\sum_{ay} p(X_1 = a, Y = y)\log_2 p(Y = y|X_1 = a) \end{aligned}$$

Warning: this is not $\boxed{p(Y = y|X_1 = a)}$

Conditional Entropy

- Conditioned on the event $X_1 = a_1$

$$H(Y|X_1=a_1) = -\sum_{y} p(Y=y|X_1=a_1)\log_2 p(Y=y|X_1=a_1)$$

- Conditioned on the random variable X_1

$$H(Y|X_1) = -\sum_{ay} p(X_1=a, Y=y)\log_2 p(Y=y|X_1=a)$$

Conditional Entropy

- Names example, eyes attribute:

Name	**Eyes**
Arthur	*Blue*
Bob	*Green*
Charlie	*Brown*
Denis	*Brown*
Edmond	*Brown*
Frida	*Green*
Georgia	*Blue*
Helen	*Blue*
Irene	*Green*
Jane	*Brown*

Name	*Blue*	*Green*	*Brown*
Arthur	1	0	0
Bob	0	1	0
Charlie	0	0	1
Denis	0	0	1
Edmond	0	0	1
Frida	0	1	0
Georgia	1	0	0
Helen	1	0	0
Irene	0	1	0
Jane	0	0	1
	3	**3**	**4**

$$p(\text{Eyes}=\text{Blue}, \text{Name}=\text{Arthur}) = \frac{1}{10} \qquad p(\text{Name}=\text{Arthur}|\text{Eyes}=\text{Blue}) = \frac{1}{3}$$

Conditional Entropy

- Names example:
  - Eyes color : 3 blue, 3 green, 4 brown

$$H(\text{Name}) = \log_2 10 = 3.32$$

$$H(\text{Name}|\text{Eyes}=\text{blue}) = \log_2 3 = 1.58$$
$$H(\text{Name}|\text{Eyes}=\text{green}) = \log_2 3 = 1.58$$
$$H(\text{Name}|\text{Eyes}=\text{brown}) = \log_2 4 = 2$$

$$H(\text{Name}|\text{Eyes}) = 2\text{x}\frac{3}{10}\log_2 3 + \frac{4}{10}\log_2 4 = 1.75$$

MALIS Grades: Conditional Entropy

- Grades example, Clouds course:

MALIS/Clouds	A	B	C	D	E	F	NA
A	3	5	2	0	0	1	2
B	16	28	7	1	0	0	18
C	6	27	9	2	0	0	26
D	2	8	3	0	0	0	7
E	1	2	0	0	0	0	3
F	0	0	0	0	0	0	11
total	**28**	**70**	**21**	**3**	**0**	**1**	**67**

$$H(\text{MALIS}|\text{Clouds}) = -\sum_{a,b} p(M=a, C=b)\,\text{Log}_2\; p(M=a|C=b)$$

$$p(M=A, C=A) = \frac{3}{190} \qquad p(M=A|C=A) = \frac{3}{28}$$
$$p(M=A, C=B) = \frac{5}{190} \qquad p(M=A|C=B) = \frac{5}{70} \quad \ldots.$$

Conditional Entropy

Proof of $H(Y) \geq H(Y|X)$

$$H(Y) - H(Y|X) = -\sum_{y} p(Y=y)\log_2 p(Y=y) + \sum_{a,y} p(X=a, Y=y)\log_2 p(Y=y|X=a)$$

$$= -\sum_{a,y} p(X=a, Y=y)\log_2 p(Y=y) + \sum_{a,y} p(X=a, Y=y)\log_2 p(Y=y|X=a)$$

$$= -\sum_{a,y} p(X=a, Y=y)\log_2 \frac{p(Y=y)}{p(Y=y|X=a)}$$

$$= -\sum_{a} p(X=a)\sum_{y} p(Y=y|X=a) \log_2 \frac{p(Y=y)}{p(Y=y|X=a)}$$

$$\geq -\sum_{a} p(X=a)\log_2\left(\sum_{y} p(Y=y|X=a)\frac{p(Y=y)}{p(Y=y|X=a)}\right) \qquad \text{(convexity of log)}$$

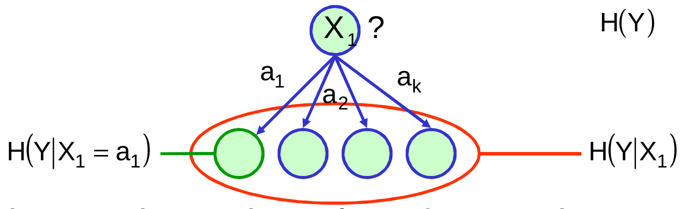
$$= -\sum_{a} p(X=a)\log_2\left(\sum_{y} p(Y=y)\right) = 0$$

Conditional Entropy

- Property:
  - it is always true that: $H(Y|X_1) \leq H(Y)$
  - so:
    $H(Y|X_1, X_2, \ldots X_{n-1}, X_n) \leq H(Y|X_1, X_2, \ldots X_{n-1}) \leq H(Y)$
- Knowing the answer to questions always improves our knowledge on Y
- By asking many questions, we get more and more knowledge about Y

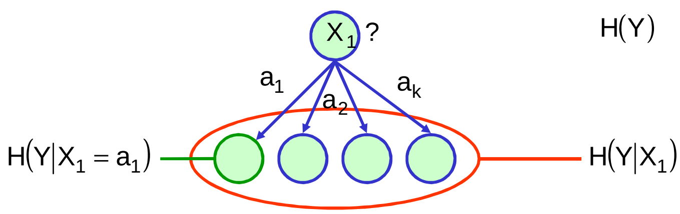
Asking Questions

- We start with uncertainty H(Y)
- We can ask about the value of X_1



- When we know the value of X_1, we have uncertainty $H(Y|X_1)$
- The best question to ask is: $\hat{\imath} = \underset{i}{\operatorname{argmin}}\ H(Y|X_i)$

Information Gain



Information gain for question i:

$$G(i) = H(Y) - H(Y|X_i) = I(Y;X_i)$$

Best question: $\hat{\imath} = \underset{i}{\operatorname{argmin}}\ H(Y|X_i) = \underset{i}{\operatorname{argmax}}\ G(i)$

$$G(\hat{\imath}) = H(Y) - H(Y|X_{\hat{\imath}}) \geq H(Y) - H(Y|X_i) = G(i)$$

Example

- Identification game:
  - H(name) = log 10 = 3.3

    H(name|gender) = 2.3

    H(name|size) = 1.75

    H(name|eyes) = 1.75

    H(name|hair) = 1.96
- Best question: size or eyes
  - entropy is reduced from 3.3 to 1.75
  - entropy gain is 3.3 – 1.75 = 1.55

MALIS Grades

- MALIS grades

Grade	A	B	C	D	E	F	NA
Nb students	13	70	70	20	6	11	0
Probability	0,07	0,37	0,37	0,11	0,03	0,06	0

$$H(MALIS) = -\sum_a p(MALIS = a)\log_2 p(MALIS = a) = 2.06$$

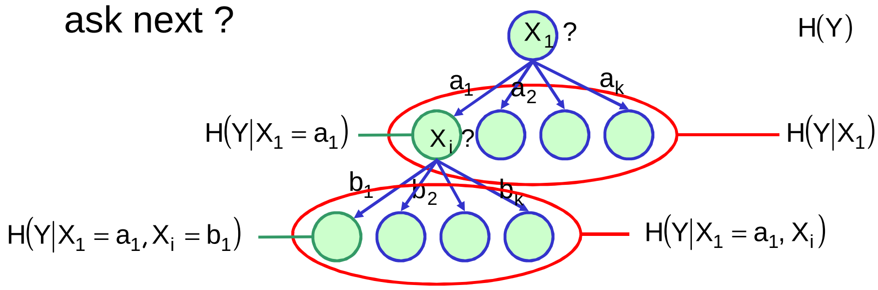
- Most informative course:

$$H(MALIS|X) = -\sum_{a,b} p(MALIS = a, X = b)\log_2 p(MALIS = a|X = b)$$

course X	SoftDev	Clouds	Optim	SecAppli	ImSecu	MMIR	ImProc	Forensics	Stat	ImCod
H(MALIS|X)	**1,88**	1,90	1,91	1,91	1,92	1,93	1,95	1,95	1,95	1,96

Second Best Question

- We have asked question X_1, which question to ask next ?



- What is the best second question to ask ?

$$\hat{\imath} = \underset{i}{\operatorname{argmin}}\ H(Y|X_1 = a_1, X_i)$$

MALIS Grades

- Most informative course: SoftDev
- Second most informative course X:

$$H(MALIS|SD = \alpha, X) =$$
$$-\sum_{ax} p(MALIS = a, X = x|SD = \alpha)\log_2 p(MALIS = a|X = x, SD = \alpha)$$

$\alpha = A$
X=Clouds

MALIS|SD=A, Clouds	A	B	C	D	E	F	NA
A	0	0	0	0	0	0	0
B	6	5	1	0	0	0	2
C	4	3	0	0	0	0	2
D	1	0	0	0	0	0	0
E	0	0	0	0	0	0	1
F	0	0	0	0	0	0	0
total	**11**	**8**	**1**	**0**	**0**	**0**	**5**

MALIS Grades

- Most informative course: SoftDev
- Second most informative courses:

	Students	Entropy	Best course
H(MALIS|SD=A,X)	25	1,37	ESP_Deb
H(MALIS|SD=B,X)	35	1,66	ManagIntro
H(MALIS|SD=C,X)	19	1,43	Entrep
H(MALIS|SD=D,X)	3	0,00	*
H(MALIS|SD=E,X)	2	0,00	*
H(MALIS|SD=F,X)	2	1,00	ManagIntro
H(MALIS|SD=NA,X)	104	2,27	ImSecu

MALIS Success

- Predict success S or failure F to MALIS
  - Instead of grade A, B, C, D, E, F

MALIS success	MALIS grade	Clouds	SoftDev	Manag Intro	MMIR	ADST	Optim	WebSem	SecAppli	...
F	E	NA	F	C	E	NA	NA	NA	NA	...
S	B	B	A	NA	NA	NA	NA	NA	NA	...
S	B	A	A	NA	NA	NA	NA	NA	NA	...
F	D	A	A	NA	NA	NA	NA	NA	NA	...
S	B	D	NA	NA	NA	A	NA	B	B	...
F	F	NA	NA	NA	NA	NA	NA	NA	NA	...
F	F	NA	NA	NA	NA	NA	F	NA	NA	...
S	C	A	A	NA	NA	A	NA	NA	NA	...
S	C	A	A	B	NA	NA	NA	NA	NA	...
S	C	NA	NA	NA	NA	B	NA	NA	C	...
S	B	B	B	NA	NA	B	NA	NA	NA	...
S	B	NA	NA	NA	NA	NA	NA	NA	NA	...
...	...	...	...	...	...	...	...	...	...	...

MALIS Success: Entropy

- MALIS: 190 students:

Grade	S	F
Nb students	153	37
Probability	0,81	0,19

- Entropy:

$$H(MALIS)=-\frac{153}{190}Log_2\left(\frac{153}{190}\right)-\frac{37}{190}Log_2\left(\frac{37}{190}\right) = 0.71$$

MALIS Success: Conditional Entropy

- Success example, Clouds course:

MALIS/Clouds	A	B	C	D	E	F	NA
S	25	60	18	3	0	1	46
F	3	10	3	0	0	0	21
total	**28**	**70**	**21**	**3**	**0**	**1**	**67**

$$H(MALIS|Clouds)=-\sum_{a,b} p(M=a,C=b)Log_2\ p(M=a|C=b)$$

$$p(M=S,C=A)=\frac{25}{190} \qquad p(M=S|C=A)=\frac{25}{28}$$

$$p(M=S,C=B)=\frac{60}{190} \qquad p(M=S|C=B)=\frac{60}{70} \quad ....$$

H(MALIS|Clouds)= 0.67

MALIS Success

- MALIS grades

Grade	S	F
Nb students	153	37
Probability	0,81	0,19

$$H(MALIS) = -\sum_a p(MALIS = a)\log_2 p(MALIS = a) = 0.71$$

- Most informative course:

$$H(MALIS|X) = -\sum_{a,b} p(MALIS = a, X = b)\log_2 p(MALIS = a|X = b)$$

course X	**ImProc**	Optim	ImCod	InfoTheo	ImSecu	SoftDev	MMIR	SecAppli	FRA_Ele_1	Clouds
H(MALIS|X)	**0,63**	0,64	0,65	0,66	0,66	0,66	0,67	0,67	0,67	0,67

MALIS Success

- Most informative course: ImProc

MALIS|IP	A	B	C	D	E	F	NA
S	4	5	0	0	0	0	144
F	0	1	4	2	0	0	30
total	**4**	**5**	**4**	**2**	**0**	**0**	**174**

  - H(MALIS|IP)=0.63

- Second most informative course X:

$$H(MALIS|IP = \alpha, X) = -\sum_{ax} p(MALIS = a, X = x|IP = \alpha)\log_2 p(MALIS = a|X = x, IP = \alpha)$$

$\alpha = A$
X=Clouds
$H(MALIS|IP = A, Cl) = 0$

MALIS|IP=A, Clouds	A	B	C	D	E	F	NA
S	2	1	1	0	0	0	0
F	0	0	0	0	0	0	0
total	**2**	**1**	**1**	**0**	**0**	**0**	**0**

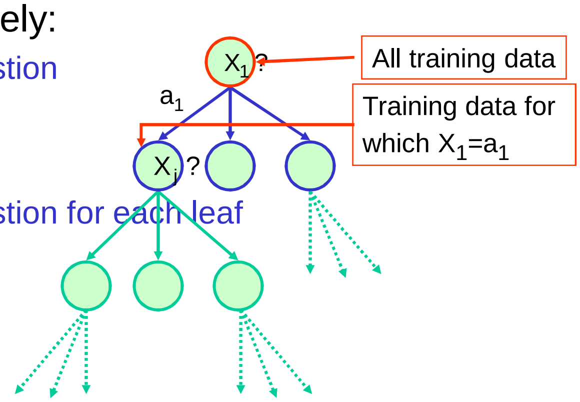
MALIS Success

- Most informative course: ImProc
- Second most informative courses:

	Students	Entropy	Best course
H(MALIS|IP=A,X)	4	0,00	*
H(MALIS|IP=B,X)	6	0,00	Entrep
H(MALIS|IP=C,X)	4	0,00	*
H(MALIS|IP=D,X)	2	0,00	*
H(MALIS|IP=E,X)	0	0,00	--
H(MALIS|IP=F,X)	0	0,00	--
H(MALIS|IP=NA,X)	174	0,60	ImSecu

Tree Growing

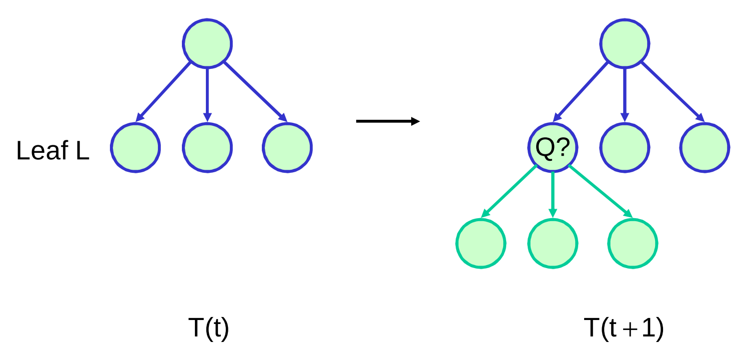
- Apply recursively:
  - find best question
  - split data
  - find best question for each leaf
  - iterate



Tree Growing

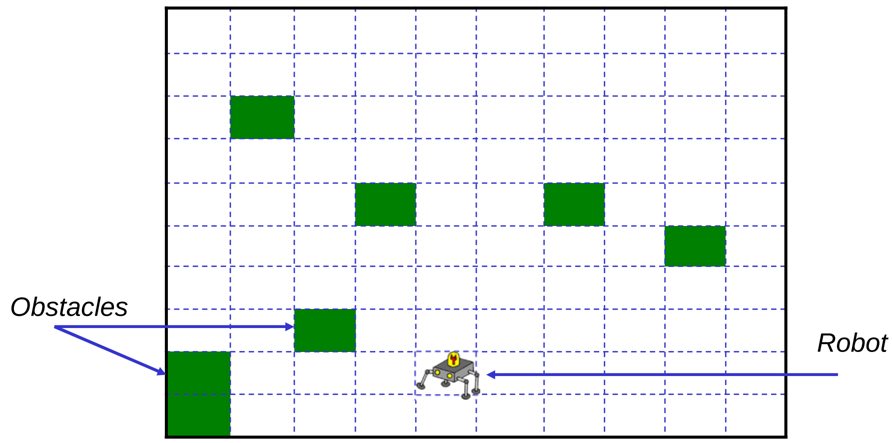
- Algorithm:
  - T(0) = {root}
  - choose an unmarked leaf L
  - find best question for L: Q
  - if Q improves entropy, then
    - T(t+1) = T(t) extended by question Q on L
    - else mark L, T(t+1) = T(t)
  - t=t+1, iterate

Tree Growing



Decision Tree Example

Robot can turn left & right, and move forward



Decision Tree Example

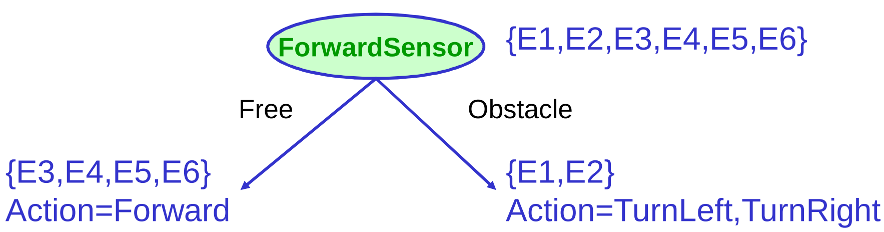
	Left Sensor	Right Sensor	Forward Sensor	Back Sensor	Previous Action	Action
E1	Obstacle	Free	Obstacle	Free	Forward	TurnRight
E2	Free	Free	Obstacle	Free	TurnLeft	TurnLeft
E3	Free	Obstacle	Free	Free	Forward	Forward
E4	Free	Obstacle	Free	Obstacle	TurnLeft	Forward
E5	Obstacle	Free	Free	Free	TurnRight	Forward
E6	Free	Free	Free	Obstacle	TurnRight	Forward

Decision Tree Example

- $H(A) = -1/6*\log_2(1/6)-1/6*\log_2(1/6)-4/6*\log_2(4/6) = 1.25$
- H(A|LS) = 2/6*H(A|LS=O) + 4/6*H(A|LS=F)
  = 2/6*1 + 4/6*0.811 = 0.874
- H(A|RS) = 2/6*H(A|RS=O) + 4/6*H(A|RS=F)
  = 2/6*0 + 4/6*1.5 = 1.
- H(A|FS) = 2/6*H(A|FS=O) + 4/6*H(A|FS=F)
  = 2/6*1 + 4/6*0 = **.333**
- H(A|BS) = 2/6*H(A|BS=O) + 4/6*H(A|BS=F)
  = 2/6*0 + 4/6*1.5 = 1.
- H(A|PA) = 2/6*H(A|PA=FW) + 2/6*H(A|PA=TL) -2/6*H(A|PA=TR)
  = 2/6*1+ 2/6*1 + 2/6*0 = 0.667

Decision Tree Example

- ForwardSensor provides the highest information gain:



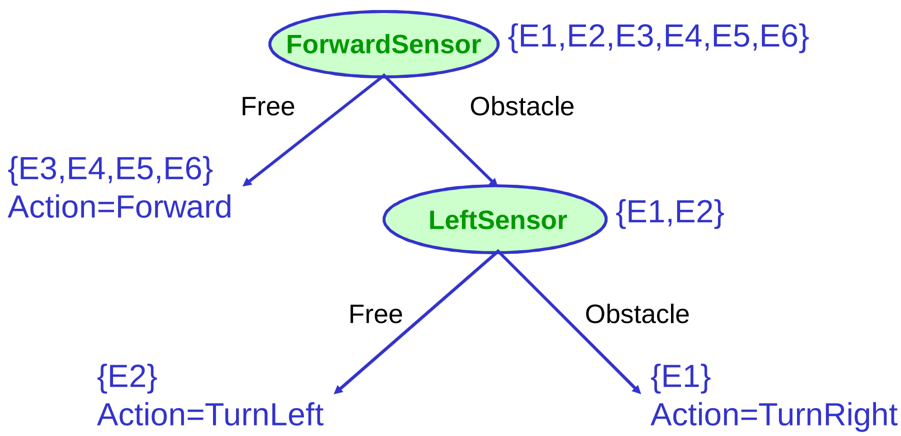
Decision Tree Example

	Left Sensor	Right Sensor	Forward Sensor	Back Sensor	Previous Action	Action
E1	Obstacle	Free	Obstacle	Free	Forward	TurnRight
E2	Free	Free	Obstacle	Free	TurnLeft	TurnLeft
E3	Free	Obstacle	Free	Free	Forward	Forward
E4	Free	Obstacle	Free	Obstacle	TurnLeft	Forward
E5	Obstacle	Free	Free	Free	TurnRight	Forward
E6	Free	Free	Free	Obstacle	TurnRight	Forward

Decision Tree Example

- H(A|FS=O) = -1/2*$\log_2$(1/2) -1/2*$\log_2$(1/2) = 1
- H(A|FS=O,LS) = 1/2*H(A|FS=O,LS=F)
  + 1/2*H(A|FS=O,LS=O)
  = 1/2*0 + 1/2*0 = 0
- H(A|FS=O,RS) = 1*H(A|FS=O,RS=F)
  +0*H(A|FS=O,RS=0) = 1
- H(A|FS=O,BS) = 1*H(A|FS=O,BS=F)
  +0*H(A|FS=O,BS=0) = 1
- H(A|FS=O,PA) = 1/2*H(A|FS=O,PA=F)
  + 1/2*H(A|FS=O,PA=TL)
  = 1/2*0 + 1/2*0 = 0
  - Select either LS or PA

Decision Tree Example



From Trees to Rules

- Note: we have discovered the rules:
  - if ForwardSensor=Free then Action=Forward
  - if ForwardSensor=Obstacle and LeftSensor=Free then Action=TurnLeft
  - if ForwardSensor=Obstacle and LeftSensor=Obstacle then Action=TurnRight

Tree Growing

- Tree gorwing is an iterative process, when do we stop splitting ?
  - Data at current node: $(x^j_1, x^j_2, \dots x^j_n, y^j)$
  - If the values of y^j are all equal
    - We can be sure that Y is equal to y^j, at least for the training data
  - If every attribute has only one value
    - We cannot ask any further question
  - What if all attributes have zero information gain ?

Tree Growing

- What if all attributes have zero information gain?

- Example : XOR
  - $H(Y) = 1$
  - $H(Y|X_1) = H(Y|X_2) = 1$
  - No information gain

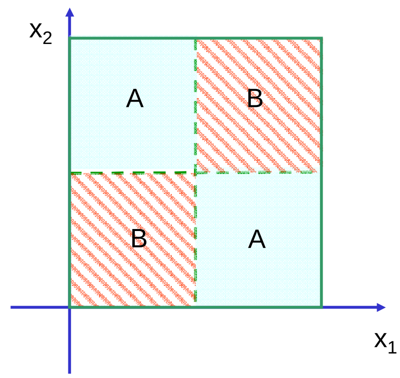
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

  - But XOR can be represented by a tree of depth 2

(Hopefully this does not happen so often in practise)

Tree Growing

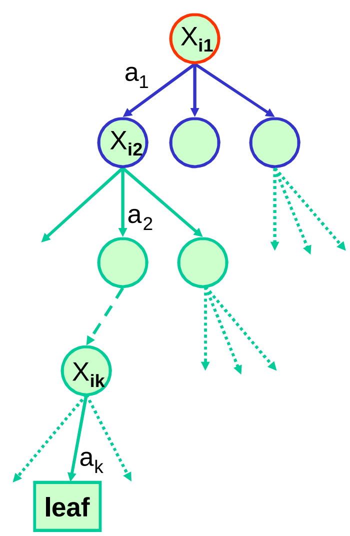
- Limitations of growing:
  - Splitting on x1 or x2 does not improve classification of A and B
  - Splitting on both at the same time would be optimal
  - Solution cannot be found by growing one node at a time only



Tree Entropy

- Each leaf of the tree corresponds to elements verifying conditions such as:

  $X_{i1}=a_1 \wedge X_{i2}=a_2 \wedge \dots \wedge X_{ik}=a_k$



Tree Entropy

- Entropy of the leaf:

$$p(Y = y|leaf) = \frac{card\{i : (x^i_1, x^i_2, ... x^i_n) \in leaf, y^i = y\}}{card\{i : (x^i_1, x^i_2, ... x^i_n) \in leaf\}}$$

$$H(Y|leaf) = -\sum_y p(Y = y|leaf) \log_2 p(Y = y|leaf)$$

- Average entropy of the tree:

$$H(Y|tree) = \sum_{leaf} p(leaf) H(Y|leaf)$$

When to stop growing ?

- Property: $H(Y|T(t+1)) \leq H(Y|T(t))$
- Stop when no question improves entropy on any leaf:
  - case 1: Y value is certain

Arthur	*Male*	*Tall*	*Blue*	*Black*

  - case 2: there remains an inherent uncertainty

Georgia	*Female*	*Short*	*Blue*	*Blond*
Helen	*Female*	*Short*	*Blue*	*Blond*

- but entropy is computed on training data...

Training and Test data

- Assume we have other test data:

$$(x^j_1, x^j_2, \ldots x^j_n, y^j) \qquad j = 1,2\ldots T_{test}$$

- What happens ?

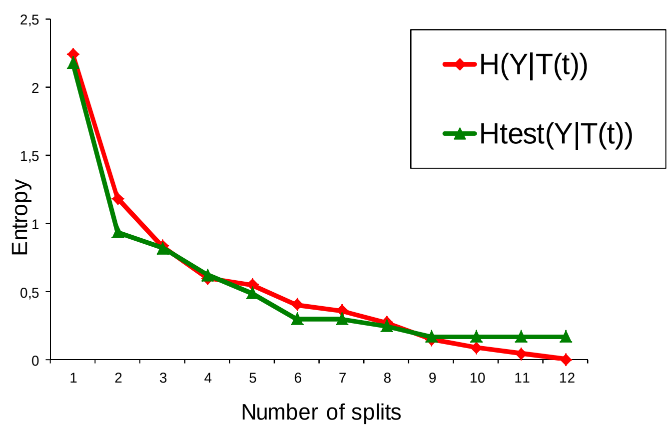
$$p_{test}\left(Y = y \middle| leaf\right) = \frac{card\left\{ j : (x^j_1, x^j_2, \ldots x^j_n) \in leaf, y^j = y \right\}}{card\left\{ j : (x^j_1, x^j_2, \ldots x^j_n) \in leaf \right\}}$$

$$H_{test}\left(Y \middle| leaf\right) = -\sum_y p_{test}(Y = y|leaf) \log_2 p_{test}(Y = y|leaf)$$

$$H_{test}\left(Y \middle| tree\right) = \sum_{leaf} p_{test}(leaf) H_{test}\left(Y \middle| leaf\right)$$

- How do $H_{test}(Y|T(t))$ compare to $H(Y|T(t))$?

Overfitting



Overfitting

- Intuitive:
  - At the beginning of the tree, we have enough data at nodes to find good “general” questions
  - As we go down the tree, we have less data, and we find too “specific” questions
  - If we end up asking too specific questions, it might be that entropy on test data will not decrease !
- This is a problem of statistical significance and insufficient data
- How can we grow a tree on training data that will be good on test data ?

Overfitting

- Two strategies:
  - Pre-pruning: decide to stop growing at some point
  - Post-pruning: grow the full tree and remove “bad” nodes
- Many criteria:
  - Use some validation data, separate from training data
  - Use statistical significance tests
  - Set bounds (number of nodes, size of node, information gain…)

Pre-pruning Criteria

- 1: stop when $H_{valid}(Y|T(t))$ does not improve
  - (or does not improve enough)
- 2: require a minimum improvement

$$H(Y|T(t)) - H(Y|T(t+1)) > \theta$$

- 3: use Minimum Description Length MDL:

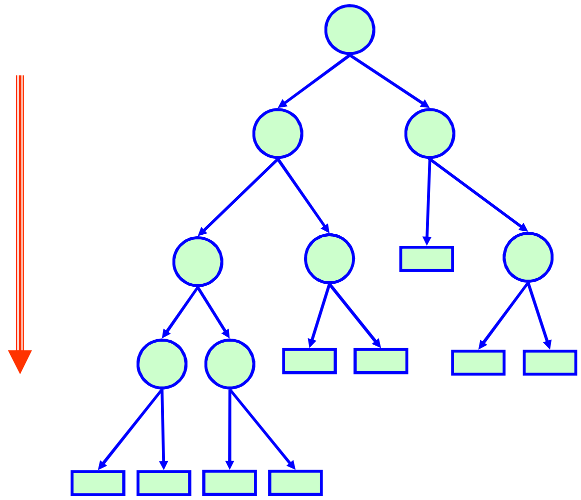
$$\min_t \left[size(T(t)) + \lambda . H(Y|T(t)) \right]$$

  - trade-off between accuracy and complexity

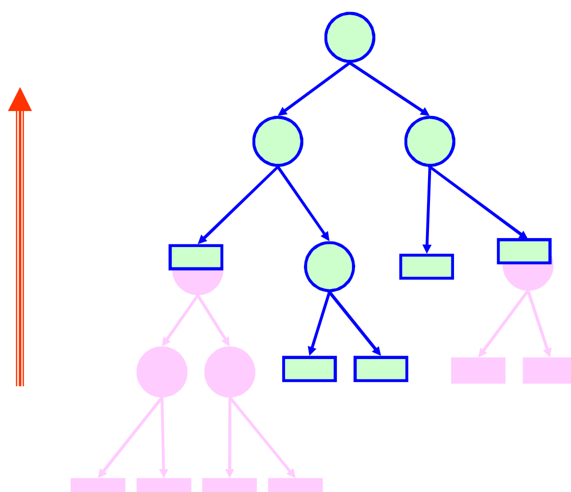
Post-pruning

- Grow the tree at maximal size (using only training data)
- Start using validation data:
  - For all nodes N in tree:
    - replace subtree from N by a leaf : new tree T_N
    - compute classification accuracy of T_N on validation data
  - Prune node N that gives greatest improvement
  - Continue until no improvements

Tree Growing

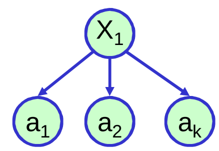


Tree Pruning



Binary Trees

- Problem:
  - X_i has k possible values
  - $H(Y) - H(Y|X) = H(X) - H(X|Y)$
  - Maximum entropy gain:
    $\log_2 k$ (upper bound)



  - Questions about attributes which have many values are likely to be more effective
  - But data is split very rapidly
- Solution:
  - Restrict to binary (Yes/No) questions

Binary Tree

- Interest of binary questions:
  - All attributes have equal chances to produce good questions
  - Data is fragmented less rapidly
- Which binary questions:
  - If attribute X has a set of value S, a binary question is characterized by a subset $S' \subset S$
    - Question: $X \in S'$?
    - Answer is yes or no
  - There are 2^k subsets, $2^{k-1}-1$ different questions.
  - May be generalized to real values: $X_i < \theta_i$?

Binary Questions

- How to find binary questions ?
  - if X_i has k possible values:
    - there are 2^k possible subsets S_i
    - there are 2^{k-1}-1 different questions (removing symetries and empty question)
  - if k is small:
    - Enumerate and select the best one

      k = 3 2^{k-1} -1= 3

      k = 26 2^{k-1}-1= 33,554,431
  - if k is large:
    - Use predefined questions built by hand
    - Try to find an algorithm to build good questions

Fixed Questions Example

- Phonetization rules:

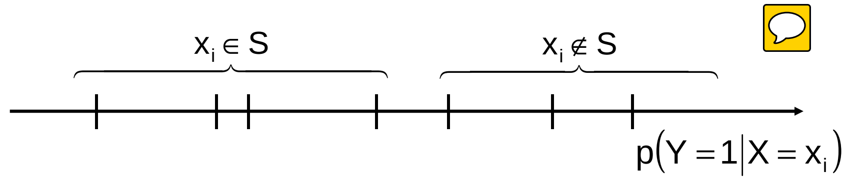
X_{-k}	…	X_{-1}	X_0	X_1	X_2	…	X_k	Y
			g	r	e	e	n	G
l	a	u	g	h				F

  - X_0 character to be pronounced
  - X_{-i}, X_i context characters
  - Y phoneme pronounced
- Possible questions:
  - Singletons {a}, {b},{c}…{z}
  - Vowels S_1 = {a, e, i, o, u, y}
  - Fricatives S_2 = {f, v, s, z, j} …

Optimal Binary Splitting

- Optimal subset $H(Y|X \in S?) = \underset{S'}{\operatorname{argmin}} \; H(Y|X \in S'?)$
- Theorem (Breiman 1984):
  - if Y is binary (values 0 and 1) and X has finite values, then there exists an optimal subset S such that:

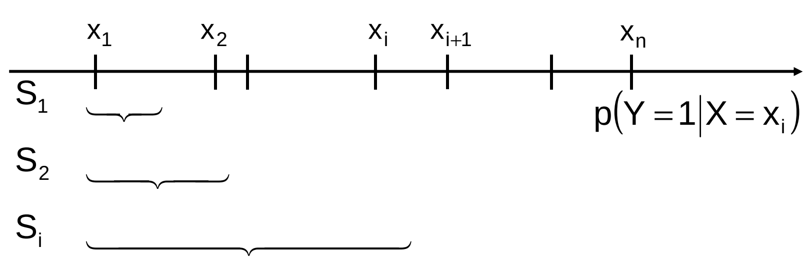
$$\forall x_1 \in S, \forall x_2 \notin S \quad p(Y=1|X=x_1) \leq p(Y=1|X=x_2)$$



Optimal Binary Splitting

- Consequence:
  - renumber x_i by increasing $p(Y=1|X=x_i)$
  - there is an optimal subset among:

$S_i = \{x_1, x_2, \dots x_i\}$ $\quad$ i=1, 2, … k-1



  - search k-1 subsets instead of 2^{k-1}-1

Optimal Binary Splitting

k	k-1	2^{k-1}-1
2	1	1
3	2	3
4	3	7
5	4	15
10	9	511
15	14	16,383
20	19	524,287
25	24	16,777,215
26	25	33,554,431

Optimal Binary Splitting

	Y=1	Y=0	p(Y=1|X=x)
X=a	1	6	0.14
X=b	5	3	0.63
X=c	0	5	0

- Ordering: c - a - b
  - H(Y|X∈{c}?) = 0.73
  - H(Y|X∈{c, a}?) = **0.63**
- Check:
  - H(Y|X∈{c,b}?) = 0.83

Flip-Flop Algorithm

- (Nadas 1991)
- Heuristic: when Y has more than 2 values
  - choose random subset S_0 of X values, set i=0
  - $X'_i = (X \in S_i?)$ is binary
  - find optimal subset V_i for Y based on X'_i
  - $Y'_i = (Y \in V_i?)$ is binary
  - find optimal subset S_{i+1} for X based on Y'_i
  - i=i+1, and iterate

Flip-flop Example

- X = α, β, γ, δ
- Y = a, b, c, d, e, f

N(X,Y)	X=α	X=β	X=γ	X=δ
Y=a	1	0	5	0
Y=b	2	0	0	0
Y=c	2	0	0	1
Y=d	1	0	2	1
Y=e	0	0	1	2
Y=f	1	0	0	1

Flip-flop Example

- Initial subset on x values(random)
  - $S_0 = \{\alpha\}$

- Initial entropy
  - $H(Y \mid X \in S_0?) = 2.13$

	α	β, γ, δ
Y=a	1	5
Y=b	2	0
Y=c	2	1
Y=d	1	3
Y=e	0	3
Y=f	1	1

Flip-flop Example

- Optimal binary split on Y for subset $S_0 = \{\alpha\}$

	α	β, γ, δ	$p(X \in S_0? \mid Y=y)$	$H(X \in S_0? \mid Y \in \{y_1, \ldots y_i\}?)$
e	0	3	0.	0.83
a	1	5	0.17	0.77
d	1	3	0.25	**0.70**
f	1	1	0.5	0.72
c	2	1	0.67	0.77
b	2	0	1.0	

- Best subset
  - $V_0 = \{e, a, d\}$

Flip-flop Example

- Optimal binary split on X for subset V_0={e,a,d}

	e a d	f c b	$p(Y \in V_0? \mid X=x)$	$H(Y \in V_0? \mid X \in \{x_1,\ldots x_i\}?)$
β	0	0	0.	0.93
α	2	5	0.29	0.70
δ	3	2	0.6	**0.59**
γ	8	0	1.	

- Best subset S_1 for X
  - $S_0 = \{\alpha\}$
  - $S_1 = \{\alpha, \beta, \delta\}$

Flip-flop Example

- $S_1 = \{\alpha, \beta, \delta\}$

	α, β, δ	γ
a	1	5
b	2	0
c	3	0
d	2	2
e	2	1
f	2	0

- New entropy:
  - $H(Y \mid X \in S_1?) = 2.03 < H(Y \mid X \in S_0?) = 2.13$

Numerical Values

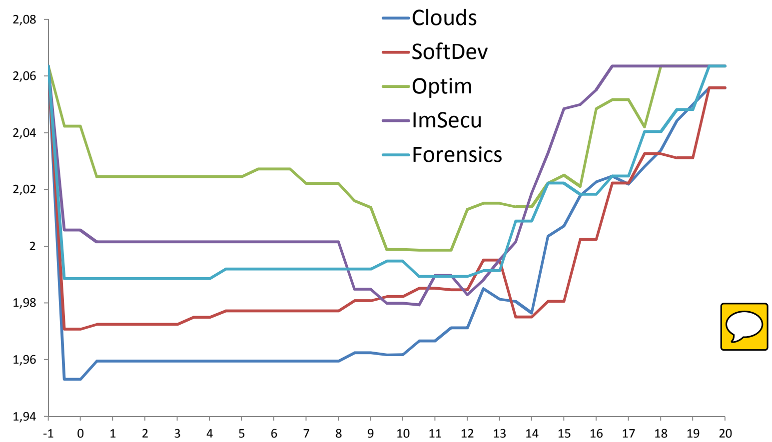
- When X_i has numerical (infinite) values
  - example: size, duration, length, grade, ...
- Possible questions: $X_i < \theta_i$?
- How to find questions ?
  - Training data $(x^j_1, x^j_2, \ldots x^j_n, y^j)$ $j = 1,2\ldots T$
  - Order the different values of $\{ x_i \}$: $\{ v_i \}$ with
    $v^j_i < v^{j+1}_i$
  - Choose $\theta_i = (v^j_i + v^{j+1}_i)/2$ $j = 1,2\ldots V\text{-}1$
  - try these V-1 questions and keep best one

MALIS Entropy per grade

	MALIS	Clouds	SoftDev	ManagIntro	MMIR	ADST	Optim	…
s1	E	NA	0	10.5	2.5	NA	NA	…
s2	B	16	19	NA	NA	NA	NA	…
s3	B	17	19	NA	NA	NA	NA	…
s4	D	17	18	NA	NA	NA	NA	…
s5	B	8	NA	NA	NA	16.5	NA	…
s6	F	NA	NA	NA	NA	NA	NA	…
s7	F	NA	NA	NA	NA	NA	0	…
s8	C	18	18	NA	NA	17	NA	…
s9	C	17	19	16	NA	NA	NA	…
s10	C	NA	NA	NA	NA	14.5	NA	…
s11	B	14	13	NA	NA	15.5	NA	…
s12	B	NA	NA	NA	NA	NA	NA	…
s13	C	15	NA	NA	13	10	12	…
s14	C	17	18	16.5	NA	NA	NA	…
s15	D	18	NA	14.5	7.5	NA	NA	…
s16	D	14	NA	NA	14	NA	NA	…
s17	F	NA	NA	NA	NA	NA	NA	…
…	…	…	…	…	…	…	…	…

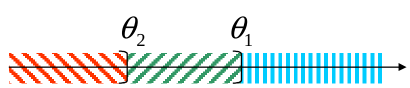
MALIS Entropy per grade

- H(MALIS|grade(X)<θ)

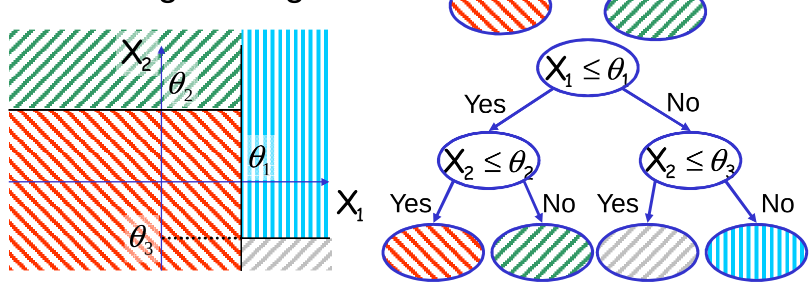


Numerical Questions

- Can model intervals



- Or rectangular regions



Ensemble Methods

Ensemble Methods

- A classifier depends on the training data and the model
  - It may have different performance on different test sets
  - If we build another classifier using a different training data and/or a different model, it will have different performance (than the first one) on test sets
- Idea: build several classifiers and combine them
  - Pro: generally better performance
  - Cons: more expensive computation

Ensemble Methods

- Assume binary classifiers $h_i: X \to \{-1,+1\}$
  - Sometimes $h_i: X \to \mathbb{R}$ classified by $\mathrm{sign}(h_i(x))$
- Combination techniques:
  - Majority vote: $\mathrm{sign}\left(\sum_i h_i(x)\right)$
  - Weighted combination: $\mathrm{sign}\left(\sum_i w_i h_i(x)\right)$
    - Useful if some classifiers are better than others
- Classifier construction:
  - Use different models
  - Use different parameters
  - Use different training sets

Ensemble Methods - Bagging

- Idea: different training sets
- Generate new training sets by randomly sampling with replacement

Training	1	2	3	4	5	6	7
Sample1	7	5	2	2	5	7	5
Sample2	2	7	4	3	5	7	6
Sample3	4	7	3	7	3	4	3
Sample4	1	5	4	5	2	4	7

- Build a classifier for each new training set
- Average predictions

Ensemble Methods - Bagging

- Advantages of bagging:
- Simple to implement: no need to change the training algorithm
- Reduces the risk of overfitting
  - Each classifier may overfit, but on different samples, so the average is less prone to overfitting

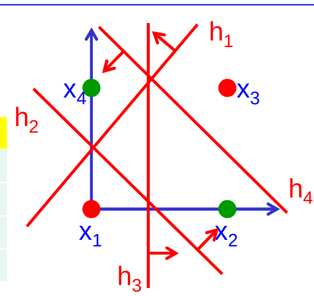
- Drawbacks:
- Requires more computation

Ensemble Methods - Bagging

- Example: XOR problem
  - With linear classifiers

Training	x_1	x_2	x_3	x_4
Sample1	x_3	x_4	x_3	x_1
Sample2	x_4	x_1	x_2	x_1
Sample3	x_2	x_2	X_2	x_1
Sample4	x_2	x_3	X_4	x_2

	h_1	h_2	h_3	h_4	sum
x_1	-1	-1	-1	1	-2
x_2	-1	1	1	1	2
x_3	-1	1	1	-1	0
x_4	1	1	-1	1	2



- Bagging could produce a perfect classifier

Ensemble Methods - Boosting

- Idea: sequentially construct a new classifier to correct the errors of the previous ones
- Weight the examples instead of sampling
- Strong learner:
  - algorithm which, given $\varepsilon<1/2$, $\delta<1/2$, can produce a classifier with a probability $>1-\delta$ that the error is $<\varepsilon$
- Weak learner:
  - algorithm which can produce a classifier with a probability $>1-\delta_0$ that the error is $<\varepsilon_0$ for some $\varepsilon_0 <1/2$, $\delta_0<1/2$
- Boosting can produce a strong learner from a weak learner

Ensemble Methods - Boosting

- Idea of boosting:
  - Train weak classifier h_1
  - Train weak classifier h_2 to correct errors from h_1
  - Train weak classifier h_3 to correct errors from h_2
  - …
  - Then combine all classifiers
- Idea of Adaboost (adaptive boosting)
  - [Freund-Schapire 1995]
  - Weight each training sample
  - Update weights after each classifier

Ensemble Methods - Boosting

- Use: $\mathbf{1}(true) = 1,\qquad \mathbf{1}(false) = 0$
- Adaboost:
  - Training samples (x_i, y_i) i=1,2,…T $\qquad y_i \in \{-1, +1\}$
  - Initial weights $w_i = 1/T$
  - Weighted error: $E(h) = \sum_i w_i\, \mathbf{1}(h(x_i) \neq y_i)$
  - For k=1, 2, …K
    - Train a classifier h_k with currents weights
    - Choose $\alpha_k \in \mathbb{R}, \quad \alpha_k \geq 0$
    - Update $w_i \to \dfrac{w_i \exp(-\alpha_k\, y_i\, h_k(x_i))}{Z_k}$

      (Z_k for normalization)
  - Final classifier: $H(x) = \text{sign}(\sum_k \alpha_k\, h_k(x))$

Ensemble Methods - Boosting

- Choice of α_k :
  - Weighted training error:

$\varepsilon_k = E(hk) = \sum_i w_i\, \mathbf{1}(h_k(x_i) \neq y_i) \qquad$ assume $\varepsilon_k \leq \varepsilon < 1/2$

  - $\alpha_k = \frac{1}{2} ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$
- Then:
  - Bound on the final error (theorem):

$$E(H) = \frac{1}{T}\sum_i \mathbf{1}(H(x_i) \neq y_i) \leq \exp\left(-2\sum_k \left(\frac{1}{2} - \varepsilon_k\right)^2\right)$$

$$\leq \exp\left(-2\left(\frac{1}{2} - \varepsilon\right)^2 K\right)$$

Error decreases exponentially with K

Ensemble Methods - Boosting

- Proof:
  - Let $f(x) = \sum_k \alpha_k h_k(x)$ so $H(x) = \text{sign}(f(x))$
  - Weights:

$$\frac{1}{T} \rightarrow \frac{1}{T}\frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1}$$

$$\rightarrow \frac{1}{T}\frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1}\frac{\exp(-\alpha_2 y_i h_2(x_i))}{Z_2}$$

…

$$\rightarrow \frac{1}{T}\frac{\exp(-y_i f(x_i))}{\prod_k Z_k}$$

  - Weights sum to 1, so: $\frac{1}{T}\sum_i \exp(-y_i f(x_i)) = \prod_k Z_k$

Ensemble Methods - Boosting

- $H(x_i) \neq y_i \;\Rightarrow\; y_i f(x_i) \leq 0 \;\Rightarrow\; \exp(-y_i f(x_i)) \geq 1$

So $\mathbf{1}(H(x_i) \neq y_i) \leq \exp(-y_i f(x_i))$

$E(H) = \frac{1}{T}\sum_i \mathbf{1}(H(x_i) \neq y_i) \leq \frac{1}{T}\sum_i \exp(-y_i f(x_i)) = \prod_k Z_k$

- $Z_k = \sum_i w_i \exp(-\alpha_k y_i h_k(x_i))$

$= \sum_{y_i \neq h_k(x_i)} w_i \exp(\alpha_k) + \sum_{y_i = h_k(x_i)} w_i \exp(-\alpha_k)$

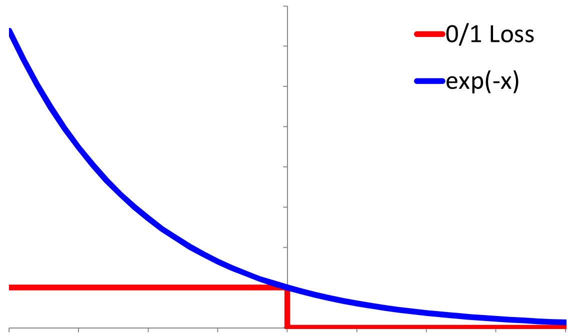
$= \varepsilon_k \exp(\alpha_k) + (1 - \varepsilon_k)\exp(-\alpha_k)$

- $\alpha_k = \frac{1}{2} ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$ is chosen to minimize Z_k

$$Z_k = 2\sqrt{\varepsilon_k(1-\varepsilon_k)} = \sqrt{1 - 4\left(\frac{1}{2} - \varepsilon_k\right)^2} \leq \exp\left(-2\left(\frac{1}{2} - \varepsilon_k\right)^2\right)$$

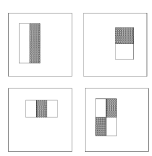
Ensemble Methods - Boosting

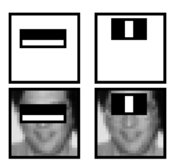
- Proof of boosting:
  - Find upper bound of 0/1 loss
  - Minimize upper bound



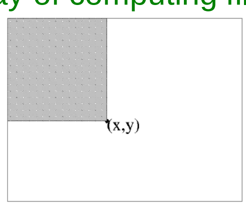
Ensemble Methods - Boosting

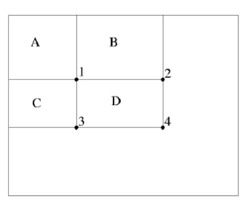
- Viola-Jones face detection [2001]
  - Haar features (rectangle filters):
    - Sum of pixels in white area minus sum of pixels I black area
    - 4 types





  - Integral images
    - Fast way of computing filter values





Ensemble Methods - Boosting

- 5K face images, 350 M non-face
- 180,000 possible filters
- Adaboost with 38 iterations
- First features selected:

- Very good performance and very fast
- Implemented in OpenCV