CS 3750 Machine Learning

Lecture 1

Advanced Machine Learning

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square, x4-8845

http://www.cs.pitt.edu/~milos/courses/cs3750/

Administration

Study material

- **Handouts, course readings**
- **Primary textbook:**
  - Friedman, Hastie, Tibshirani. *Elements of statistical learning*. Springer, 2001.
- **Other books:**
  - C. Bishop. *Neural networks for pattern recognition*. Oxford U. Press, 1996.
  - Duda, Hart, Stork. *Pattern classification*. 2nd edition. J Wiley and Sons, 2000.
  - M. Jordan. Graphical models. unpublished
  - B. Scholkopf and A. Smola. *Learning with kernels.* MIT Press, 2002.

Administration

- **Classes:**
  - **Lectures**
  - **Paper discussions/Paper presentations**
- **No Homeworks and Exams**
- **Projects: 2 projects**
  - **Midterm project (assigned)**
  - **Final project (student writes a proposal)**

- **Grading:**
  - **Projects**
  - **Paper presentations/ discussions**

Tentative topics

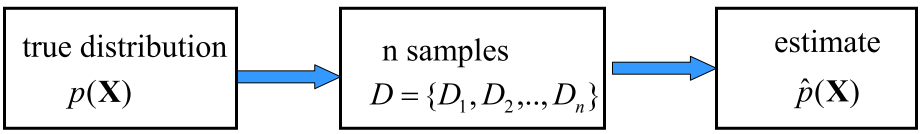
- **Review** of density estimation and classification methods.
- **Ridge regression**, regularization, prior smoothing.
- **Graphical models** of multivariate distributions.
  - Directed and undirected models.
  - Inference.
  - Learning of parameters and structure.
- **Variational approximations** for inference and learning.
  - Mean-field approximations. Variational Bayes.
- **Kernel methods**
  - Kernel methods, Kernel-PCA, string kernels, etc.

Density estimation

Data: $D = \{D_1, D_2, .., D_n\}$

$D_i = \mathbf{x}_i$ a vector of attribute values

Objective: try to estimate the underlying true probability distribution over variables $\mathbf{X}$, $p(\mathbf{X})$, using examples in D



Standard (iid) assumptions: Samples

- **are independent of each other**
- **come from the same (identical) distribution (fixed** $p(\mathbf{X})$**)**

Density estimation

Types of density estimation:

Parametric

- the distribution is modeled using a set of parameters Θ

$$p(\mathbf{X} \mid \Theta)$$

- **Example:** mean and covariances of multivariate normal
- **Estimation:** find parameters $\hat{\Theta}$ that fit the data D the best

Non-parametric

- The model of the distribution utilizes all examples in D
- As if all examples were parameters of the distribution
- **Examples:** Nearest-neighbor

Semi-parametric

Density estimation

Types of density estimation:

Parametric

- the distribution is modeled using a set of parameters Θ

$$p(\mathbf{X} \mid \Theta)$$

- **Example:** mean and covariances of multivariate normal
- **Estimation:** find parameters $\hat{\Theta}$ that fit the data D the best

Non-parametric

- The model of the distribution utilizes all examples in D
- As if all examples were parameters of the disrtibution
- **Examples:** Nearest-neighbor

Semi-parametric

Parametric density estimation

In this lecture we consider **parametric density estimation**

Basic settings:

- A set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$
- **A model of the distribution** over variables in $\boldsymbol{X}$ with parameters Θ
- **Data** $D = \{D_1, D_2, .., D_n\}$

Objective: find parameters $\hat{\Theta}$ that fit the data the best

This lecture: basic parametric models

- **Models from the exponential family of distributions**

Basic criteria

What is the best set of parameters?

- **Maximum likelihood (ML)**

  maximize $p(D \mid \Theta, \xi)$

  ξ - represents prior (background) knowledge

- **Maximum a posteriori probability (MAP)**

  maximize $p(\Theta \mid D, \xi)$

  Selects the mode of the posterior

$$p(\Theta \mid D, \xi) = \frac{p(D \mid \Theta, \xi)\, p(\Theta \mid \xi)}{p(D \mid \xi)}$$

Example. Bernoulli distribution.

Outcomes: two possible values – 0 or 1 (head or tail)
Data: D a sequence of outcomes x_i with 0,1 values

Model: probability of an outcome 1 θ
probability of 0 $(1-\theta)$

$$P(x_i \mid \theta) = \theta^{x_i}(1-\theta)^{(1-x_i)}$$ **Bernoulli distribution**

Objective:

We would like to estimate the probability of seeing 1: $\hat{\theta}$

Maximum likelihood (ML) estimate.

Likelihood of data:

$$P(D \mid \theta, \xi) = \prod_{i=1}^{n} \theta^{x_i} (1-\theta)^{(1-x_i)}$$

Maximum likelihood estimate

$$\theta_{ML} = \arg\max_{\theta} P(D \mid \theta, \xi)$$

Optimize log-likelihood

$$l(D,\theta) = \log P(D \mid \theta, \xi) = \log \prod_{i=1}^{n} \theta^{x_i} (1-\theta)^{(1-x_i)} =$$

$$\sum_{i=1}^{n} x_i \log\theta + (1-x_i)\log(1-\theta) = \log\theta \underbrace{\sum_{i=1}^{n} x_i} + \log(1-\theta) \underbrace{\sum_{i=1}^{n} (1-x_i)}$$

N_1 - number of 1s seen

N_2 - number of 0s seen

Maximum likelihood (ML) estimate.

Optimize log-likelihood

$$l(D,\theta) = N_1 \log\theta + N_2 \log(1-\theta)$$

Set derivative to zero

$$\frac{\partial l(D,\theta)}{\partial \theta} = \frac{N_1}{\theta} - \frac{N_2}{(1-\theta)} = 0$$

Solving

$$\theta = \frac{N_1}{N_1 + N_2}$$

ML Solution:

$$\theta_{ML} = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

Maximum a posteriori estimate

Maximum a posteriori estimate

– Selects the mode of the posterior distribution

$$\theta_{MAP} = \arg\max_{\theta} p(\theta \mid D, \xi)$$

$$p(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) p(\theta \mid \xi)}{P(D \mid \xi)} \quad \textbf{(via Bayes rule)}$$

$P(D \mid \theta, \xi)$ - is the likelihood of data

$$P(D \mid \theta, \xi) = \prod_{i=1}^{n} \theta^{x_i} (1-\theta)^{(1-x_i)} = \theta^{N_1} (1-\theta)^{N_2}$$

$p(\theta \mid \xi)$ - is the prior probability on θ

How to choose the prior probability?

Prior distribution

Choice of prior: Beta distribution

$$p(\theta \mid \xi) = Beta(\theta \mid \alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \theta^{\alpha_1 - 1} (1-\theta)^{\alpha_2 - 1}$$

Why?

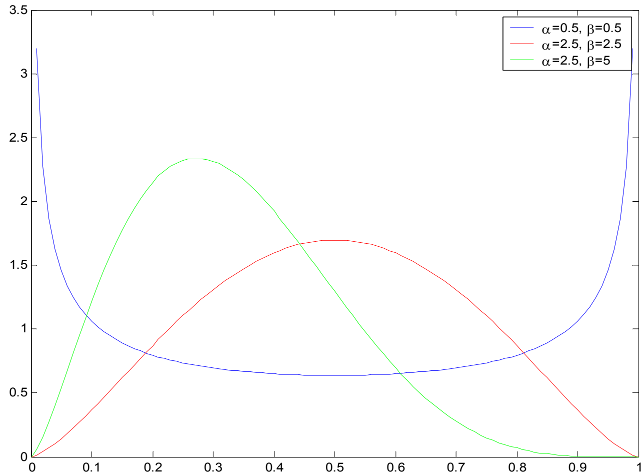
Beta distribution "**fits**" binomial sampling - **conjugate choices**

$$P(D \mid \theta, \xi) = \theta^{N_1} (1-\theta)^{N_2}$$

$$p(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) Beta(\theta \mid \alpha_1, \alpha_2)}{P(D \mid \xi)} = Beta(\theta \mid \alpha_1 + N_1, \alpha_2 + N_2)$$

MAP Solution: $$\theta_{MAP} = \frac{\alpha_1 + N_1 - 1}{\alpha_1 + \alpha_2 + N_1 + N_2 - 2}$$

Beta distribution



Bayesian learning

- **Both ML or MAP pick one parameter value**
  - Is it always the best solution?
- **Full Bayesian approach**
  - Remedies the limitation of one choice
  - Keeps and uses a complete posterior distribution
- **How is it used? Assume we want:** $P(\Delta \mid D, \xi)$
  - Considers all parameter settings and averages the result

$$P(\Delta \mid D, \xi) = \int_{\theta} P(\Delta \mid \theta, \xi) p(\theta \mid D, \xi) d\theta$$

  - **Example:** predict the result of the next outcome
    - Choose outcome 1 if $P(x=1 \mid D, \xi)$ is higher

Supervised learning

Data: $D = \{d_1, d_2, .., d_n\}$ **a set of *n* examples**

$d_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i$ is input vector, and y is desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(x_i)$ for all $i = 1, .., n$

Two types of problems:

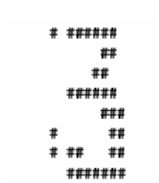
- **Regression:** X discrete or continuous $\rightarrow$
  Y is **continuous**
- **Classification:** X discrete or continuous $\rightarrow$
  Y is **discrete**

Supervised learning examples

- **Regression:** Y is **continuous**

Debt/equity
Earnings
Future product orders
$\longrightarrow$ company stock price

- **Classification:** Y is **discrete**

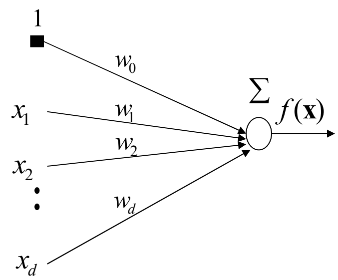

$\longrightarrow$ Label "3"

Handwritten digit (array of 0,1s)

Linear units

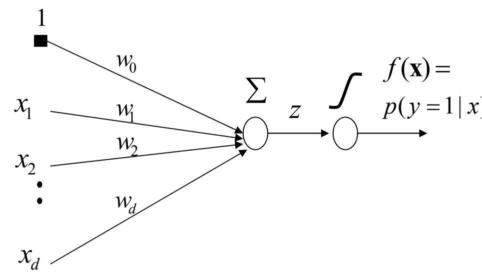
Linear regression

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$



Logistic regression

$$f(\mathbf{x}) = p(y=1\,|\,\mathbf{x},\mathbf{w}) = g(\mathbf{w}^T\mathbf{x})$$



On-line gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - f(\mathbf{x}))\mathbf{x}$$

The same

On-line gradient update:

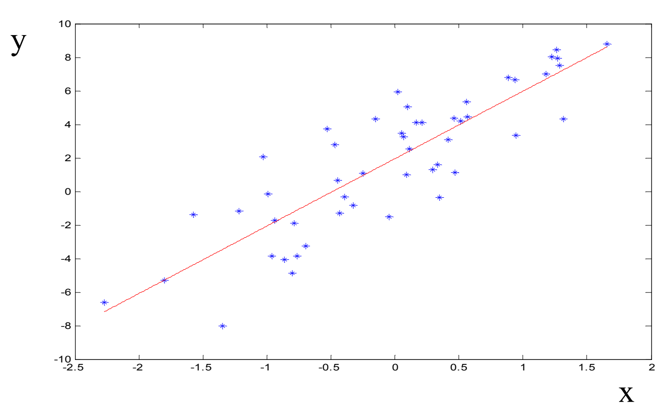
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - f(\mathbf{x}))\mathbf{x}$$

Linear regression

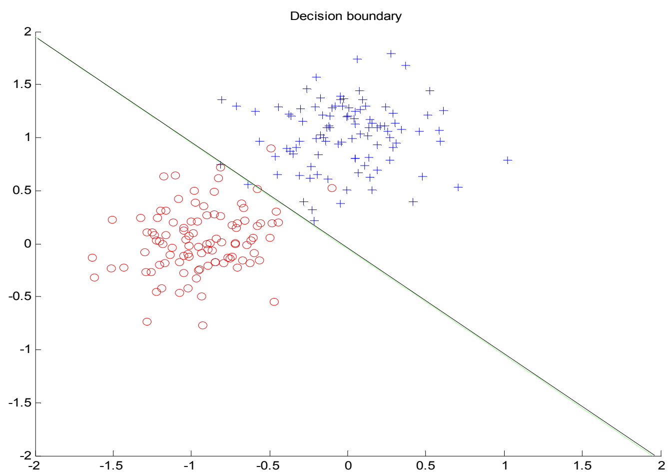
- Model

$$f(x) = ax + b + \varepsilon$$

$\varepsilon = N(0,\sigma)$ - random (normally distributed) noise



Logistic regression



Ridge regression

- For **high dimensional inputs** the prediction accuracy can be often improved by setting some coefficients to zero
- Error function for the standard least squares estimates:

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- **We seek:** $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$

- **Ridge regression:**

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|^2$$

- Where $\|\mathbf{w}\|^2 = \sum_{i=0}^{d} w_i^2$ and $\lambda \geq 0$
- What does the new error function do?

Ridge regression

- **Standard regression:**

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- **Ridge regression:**

$$J_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|^2$$

- $\|\mathbf{w}\|^2 = \sum_{i=0}^{d} w_i^2$ penalizes non-zero weights with the cost proportional to λ (a shrinkage coefficient)
- If an input attribute x_j has a small effect on improving the error function it is "shut down" by the penalty term
- Inclusion of a shrinkage penalty is often referred to as **regularization**

Modeling complex multivariate distributions

How to model complex multivariate distributions $\hat{p}(\mathbf{X})$ with large number of variables?

One solution:

- **Decompose the distribution along conditional independence relations.**

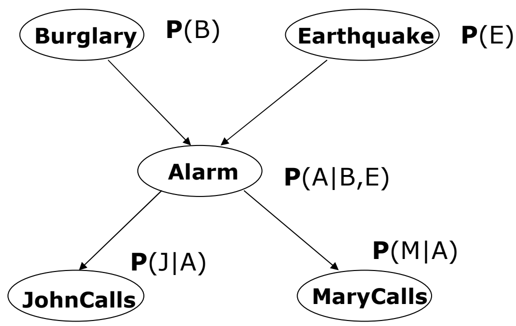
Two models:

- **Bayesian belief networks (BBNs)**
- **Markov Random Fields** (MRFs)

- **Learning.** Relies on the decomposition.

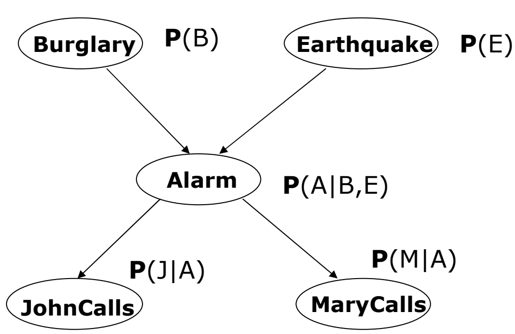
Bayesian belief network.

1. **Directed acyclic graph**
   - **Nodes** = random variables
   - **Links** = direct (causal) dependencies between variables.

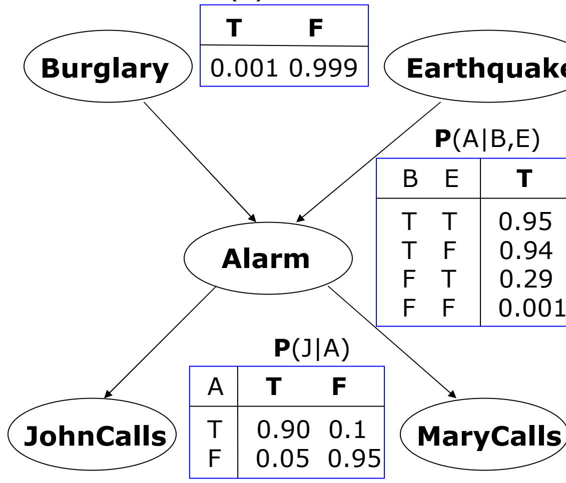


Bayesian belief network.

2. **Local conditional distributions**
   - relate variables and their parents



Bayesian belief network.



P(B)

T	F
0.001	0.999

P(E)

T	F
0.002	0.998

P(A|B,E)

B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

P(J|A)

A	T	F
T	0.90	0.1
F	0.05	0.95

P(M|A)

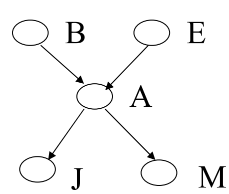
A	T	F
T	0.7	0.3
F	0.01	0.99

Full joint distribution in BBNs

Full joint distribution is defined in terms of local conditional distributions (obtained via the chain rule):

$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$

Example:



Assume the following assignment of values to random variables

$B=T, E=T, A=T, J=T, M=F$

Then its probability is:

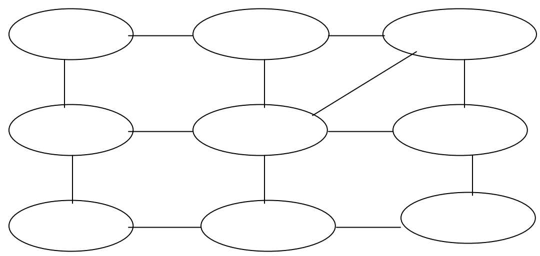
$$P(B=T, E=T, A=T, J=T, M=F) =$$
$$P(B=T)P(E=T)P(A=T \mid B=T, E=T)P(J=T \mid A=T)P(M=F \mid A=T)$$

Markov Random Fields (MRFs)

Undirected acyclic graph

- **Nodes** = random variables
- **Links** = direct relations between variables

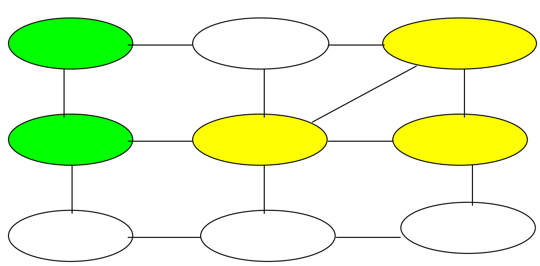
- BBNs used to model asymetric dependencies (most often causal),
- MRFs model symmetric dependencies (bidirectional effects) such as spatial dependences



Markov Random Fields (MRFs)

A probability distribution is defined in terms of potential functions defined over cliques of the graph

$$\mathbf{P}(X_1, X_2, .., X_n) = \frac{1}{Z} \prod_{C_i \in cliques(G)} \Psi(C_i)$$



Tentative topics

- **Review** of density estimation and classification methods.
- **Ridge regression**, regularization, prior smoothing.
- **Graphical models** of multivariate distributions.
  - Directed and undirected models.
  - Inference.
  - Learning of parameters and structure.
- **Variational approximations** for inference and learning.
  - Mean-field approximations. Variational Bayes.
- **Kernel methods**
  - Kernel methods, Kernel-PCA, string kernels, etc.

Linearly separable classes

There is a **hyperplane** that separates training instances with no error

Hyperplane:

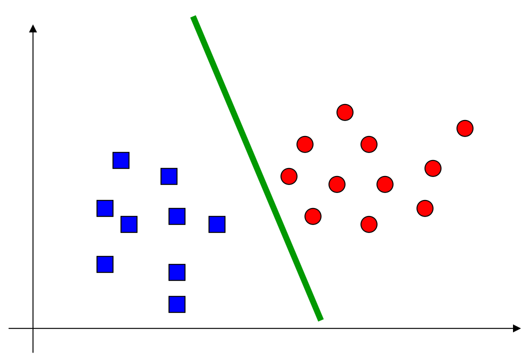
$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$

Class (+1)

$$\mathbf{w}^T\mathbf{x} + w_0 > 0$$

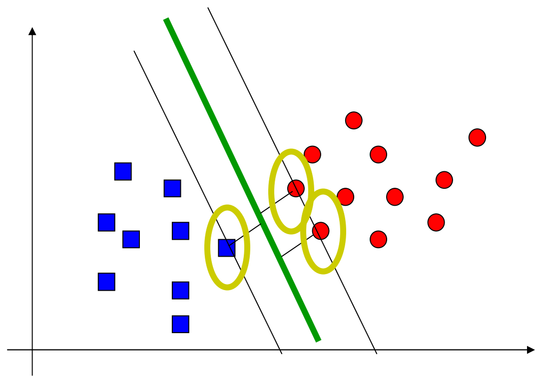
Class (-1)

$$\mathbf{w}^T\mathbf{x} + w_0 < 0$$



Maximum margin hyperplane

- For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
- These are called **support vectors**



Maximum margin hyperplane

- **We want to maximize** $d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$
- We do it by **minimizing**

$$\|\mathbf{w}\|^2 / 2 = \mathbf{w}^T \mathbf{w} / 2$$

$\mathbf{w}, w_0$ - variables

– But we also need to enforce the constraints on points:

$$\left[y_i (\mathbf{w}^T \mathbf{x} + w_0) - 1 \right] \geq 0$$

Maximum margin hyperplane

- **Solution: Incorporate constraints into the optimization**
- **Optimization problem** (Lagrangian)

$$J(\mathbf{w}, w_0, \alpha) = \|\mathbf{w}\|^2 / 2 - \sum_{i=1}^{n} \alpha_i \left[y_i (\mathbf{w}^T \mathbf{x} + w_0) - 1 \right]$$

$\alpha_i \geq 0$ - **Lagrange multipliers**

- **Minimize** with regard to $\mathbf{w}, w_0$ (primal variables)
- **Maximize** with regard to $\boldsymbol{\alpha}$ (dual variables)

Lagrange multipliers enforce the satisfaction of constraints

If $\left[y_i (\mathbf{w}^T \mathbf{x} + w_0) - 1 \right] > 0 \implies \alpha_i \to 0$

Else $\implies \alpha_i > 0$ Active constraint

Max margin hyperplane solution

- Set derivatives to 0 (Kuhn-Tucker conditions)

$$\nabla_{\mathbf{w}} J(\mathbf{w}, w_0, \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \bar{0}$$

$$\frac{\partial J(\mathbf{w}, w_0, \alpha)}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

- Now we need to solve for Lagrange parameters (Wolfe dual)

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$ ⟸ **maximize**

Subject to constraints

$\alpha_i \geq 0$ for all i, and $\sum_{i=1}^{n} \alpha_i y_i = 0$

- **Quadratic optimization problem:** solution $\hat{\alpha}_i$ for all i

Maximum hyperplane solution

- The resulting parameter vector $\hat{\mathbf{w}}$ can be expressed as:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i \qquad \hat{\alpha}_i \text{ is the solution of the dual problem}$$

- The parameter w_0 is obtained through Karush-Kuhn-Tucker conditions

$$\hat{\alpha}_i \left[y_i (\hat{\mathbf{w}} \mathbf{x}_i + w_0) - 1 \right] = 0$$

Solution properties

- $\hat{\alpha}_i = 0$ for all points that are not on the margin
- $\hat{\mathbf{w}}$ is a **linear combination of support vectors only**
- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i{}^T \mathbf{x}) + w_0 = 0$$

Support vector machines

- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i{}^T \mathbf{x}) + w_0$$

- **The decision:**

$$\hat{y} = \operatorname{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i{}^T \mathbf{x}) + w_0 \right]$$

Note:

- Decision on a new **x** requires to compute the inner product between the examples $(\mathbf{x}_i{}^T \mathbf{x})$
- Similarly, optimization depends on $(\mathbf{x}_i{}^T \mathbf{x})$

Nonlinear case

- The linear case requires to compute $(\mathbf{x}_i^T \mathbf{x})$
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

$$\mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x})$$

- It is possible to use SVM formalism on feature vectors

$$\boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Kernel function**

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Crucial idea:** If we choose the kernel function wisely we can compute linear separation in the feature space implicitly such that we keep working in the original input space !!!!

Kernel function example

- Assume $\mathbf{x} = [x_1, x_2]^T$ and a feature mapping that maps the input into a quadratic feature set

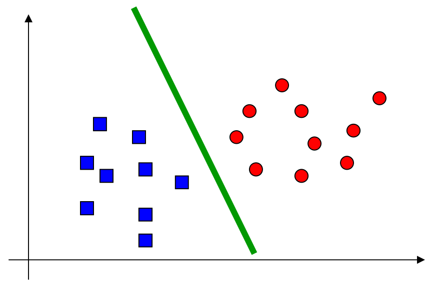
$$\mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

- Kernel function for the feature space:

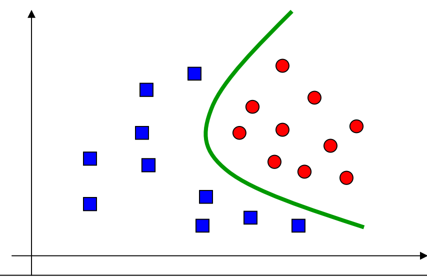
$$K(\mathbf{x}', \mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x})$$
$$= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_2 x_1' x_2' + 2x_1 x_1' + 2x_2 x_2' + 1$$
$$= (x_1 x_1' + x_2 x_2' + 1)^2$$
$$= (1 + (\mathbf{x}^T \mathbf{x}'))^2$$

- The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space

Kernel function example



Linear separator
in the feature space



Non-linear separator
in the input space

Kernel functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left[1 + \mathbf{x}^T \mathbf{x}'\right]^k$$

- **Radial basis kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|^2\right]$$

- **One view: kernels define a distance measure**