CS 3750 Machine Learning

Lecture 3

Advanced Machine Learning

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

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

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
→ company stock price

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

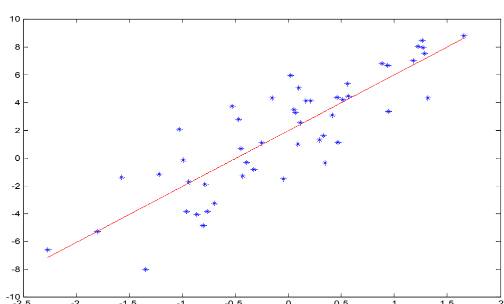
```
# ######
      ##
     ##
  ######
      ###
#      ##
# ##   ##
  #######
```

→ Label "3"

Handwritten digit (array of 0,1s)

Linear regression

- Model $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$

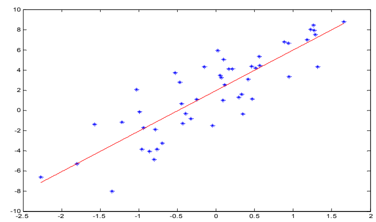


Mean Squared Error (or Loss):

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

Linear regression

- Model $y = f(x) = \mathbf{w}^{\mathrm{T}} \mathbf{x} + b$



Alternative view:

$$f(\mathbf{x}) \sim N(\mathbf{w}^T \mathbf{x} + b, \sigma)$$

Optimize the predictive loglikelihood

$$\log P(Y \mid X, \mathbf{w}) = \log \prod_{i=1,\ldots n} P(y_i \mid x_i, \mathbf{w}) = -C \sum_{i=1,..n} \left(y_i - \mathbf{w}^T \mathbf{x} - b\right)^2 + B$$

Regularization

Penalty for the model complexity

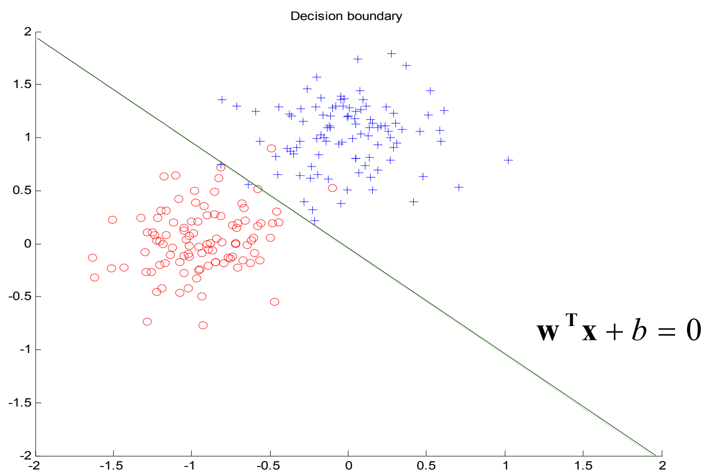
- **L1 (lasso) regularization penalty**
- **L2 (ridge) regularization penalty**

- Typically: the optimization of weights **w** looks as follows

$$\min_{\mathbf{w}} \quad \underbrace{Loss(D, \mathbf{w})}_{\text{fit}} + \underbrace{Q(\mathbf{w})}_{\text{Complexity penalty}}$$

- $Loss(D, \mathbf{w})$ functions:
  - Mean squared error
  - Negative log-likelihood
- Regularization penalty $Q(\mathbf{w})$:
  - L1 $\quad Q(\mathbf{w}) = \| \mathbf{w} \|_{L1} = \sum_{i=1,\ldots d} | w_i |$
  - L2 $\quad Q(\mathbf{w}) = \| \mathbf{w} \|_{L2} = \left(\sum_{i=1,\ldots d} w_i^2 \right)^{\frac{1}{2}}$

Classification: Linear decision boundary



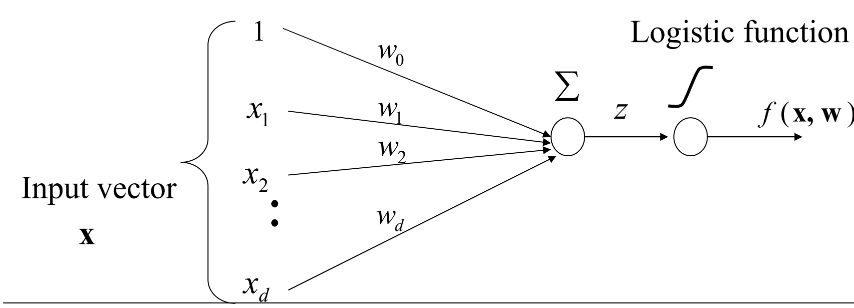
Logistic regression model

- **Discriminant functions:**

$$g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x}) \qquad\qquad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$$

- **where** $g(z) = 1/(1+e^{-z})$ - is a logistic function

$$f(\mathbf{x}, \mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Linear decision boundary

- Logistic regression model defines a **linear decision boundary**
- **Why?**
- **Answer:** Compare two **discriminant functions**.
- **Decision boundary:** $g_1(\mathbf{x}) = g_0(\mathbf{x})$
- For the boundary it must hold:

$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{1 - g(\mathbf{w}^{\mathbf{T}}\mathbf{x})}{g(\mathbf{w}^{\mathbf{T}}\mathbf{x})} = 0$$

$$\log \frac{g_o(\mathbf{x})}{g_1(\mathbf{x})} = \log \frac{\dfrac{\exp -(\mathbf{w}^{\mathbf{T}}\mathbf{x})}{1+\exp -(\mathbf{w}^{\mathbf{T}}\mathbf{x})}}{\dfrac{1}{1+\exp -(\mathbf{w}^{\mathbf{T}}\mathbf{x})}} = \log \exp -(\mathbf{w}^{\mathbf{T}}\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} = 0$$

Logistic regression: parameter learning

Likelihood of outputs

- **Let**

$$D_i = < \mathbf{x}_i, y_i > \qquad \mu_i = p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = g(z_i) = g(\mathbf{w}^T \mathbf{x})$$

- **Then**

$$L(D, \mathbf{w}) = \prod_{i=1}^{n} P(y = y_i \mid \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i}$$

- **Find weights w that maximize the likelihood of outputs**
  - Apply the log-likelihood trick. The optimal weights are the same for both the likelihood and the log-likelihood

$$l(D, \mathbf{w}) = \log \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i} = \sum_{i=1}^{n} \log \mu_i^{y_i} (1 - \mu_i)^{1 - y_i} =$$

$$= \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log (1 - \mu_i)$$

Regularization

The same way as for the linear regression model we can penalize non-zero weights of the logistic regression model

- **L1 (lasso) regularization penalty**
- **L2 (ridge) regularization penalty**

- The optimization of weights $\mathbf{w}$ looks as follows

$$\min_{\mathbf{w}} \quad \underbrace{Loss(D,\mathbf{w})}_{\text{fit}} + \underbrace{Q(\mathbf{w})}_{\text{Complexity penalty}}$$

- $Loss(D,\mathbf{w})$ functions: $-\ l(D,\mathbf{w})$
- Regularization penalty $Q(\mathbf{w})$:
  - L1 $\quad Q(\mathbf{w}) = \| \mathbf{w} \|_{L1} = \sum_{i=1,\dots d} | w_i |$
  - L2 $\quad Q(\mathbf{w}) = \| \mathbf{w} \|_{L2} = \left(\sum_{i=1,\dots d} w_i^2 \right)^{\frac{1}{2}}$

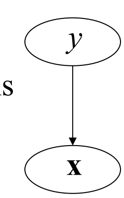
Generative approach to classification

Idea:

1. **Represent and learn the distribution** $p(\mathbf{x}, y)$
2. **Use it to define probabilistic discriminant functions**

   E.g. $g_o(\mathbf{x}) = p(y = 0 \mid \mathbf{x}) \qquad g_1(\mathbf{x}) = p(y = 1 \mid \mathbf{x})$

Typical model $\qquad p(\mathbf{x}, y) = p(\mathbf{x} \mid y) p(y)$



- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  binary classification: two class-conditional distributions

  $$p(\mathbf{x} \mid y = 0) \qquad p(\mathbf{x} \mid y = 1)$$

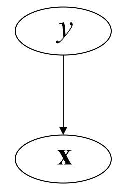
- $p(y)$ = **Priors on classes** - probability of class y

  binary classification: Bernoulli distribution

  $$p(y = 0) + p(y = 1) = 1$$

Quadratic discriminant analysis (QDA)

Model:



- **Class-conditional distributions**
  - **multivariate normal distributions**

$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad \text{for} \quad y = 0$$

$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{for} \quad y = 1$$

Multivariate normal $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- **Priors on classes (class 0,1)** $y \sim Bernoulli$
  - **Bernoulli distribution**

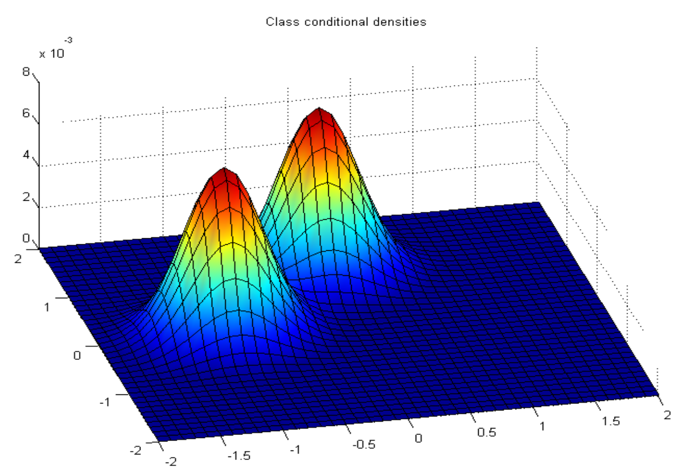
$$p(y, \theta) = \theta^y (1 - \theta)^{1-y} \qquad y \in \{0,1\}$$

Linear discriminant analysis (LDA)

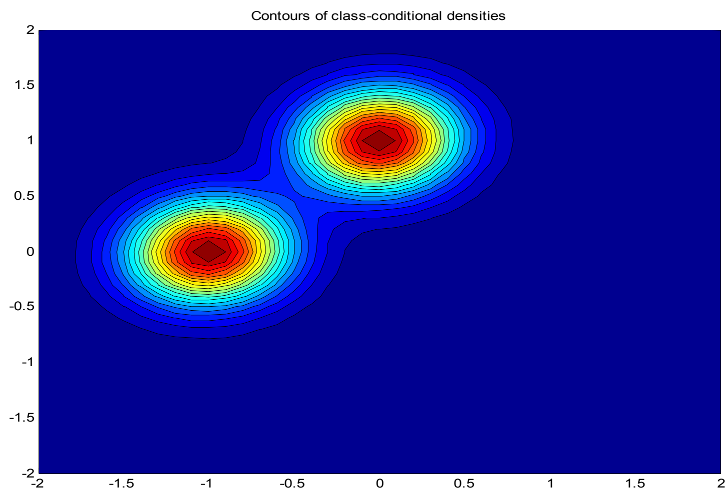
- When covariances are the same

$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}),\ y = 0$$

$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}),\ y = 1$$



LDA: Linear decision boundary



LDA: linear decision boundary

When is the logistic regression model correct?

- **Members of the exponential family can be often more naturally described as**

$$f(\mathbf{x} \mid \boldsymbol{\theta}, \boldsymbol{\varphi}) = h(x, \boldsymbol{\varphi}) \exp\left\{ \frac{\boldsymbol{\theta}^T \mathbf{x} - A(\boldsymbol{\theta})}{a(\boldsymbol{\varphi})} \right\}$$

$\boldsymbol{\theta}$ - A location parameter $\quad \boldsymbol{\varphi}$ - A scale parameter

- **Claim:** A logistic regression is a correct model when class conditional densities are from the same distribution in the exponential family and have **the same scale factor** $\boldsymbol{\varphi}$
- **Very powerful result !!!!**
  - **We can represent posteriors of many distributions with the same small network**

Fisher linear discriminant

Error: $$J(\mathbf{w}) = \frac{m_2 - m_1}{s_1^2 + s_2^2}$$

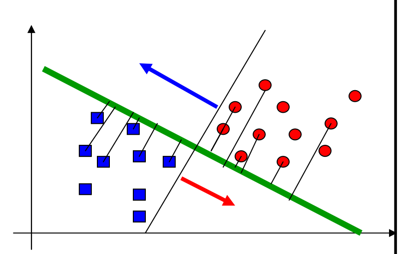
Within class variance after the projection

$$s_k^2 = \sum_{i \in C_k} (y_i - m_k)^2$$

Optimal solution:

$$\mathbf{w} \approx \mathbf{S}_{\mathbf{w}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

$$\mathbf{S}_{\mathbf{w}} = \sum_{i \in C_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T + \sum_{i \in C_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^T$$



Other algorithms

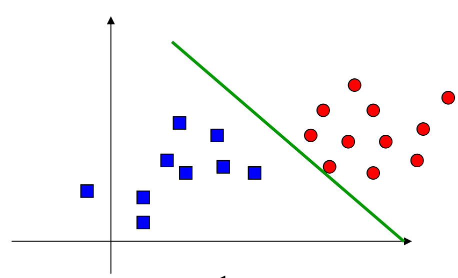
Perceptron algorithm:

- Simple iterative procedure for modifying the weights of the linear model
- Works for inputs $\mathbf{x}$ where each x_i is in [0,1]
- **guaranteed convergence if the classes are linearly separable**

Winow algorithm:

- Similar to perceptron. Different weight update
- Guaranted convergence even for nonseparable classes

Algorithms for linearly separable sets



Linear program solution:

- Finds weights that satisfy the following constraints:

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 0 \qquad \text{For all i, such that } y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq 0 \qquad \text{For all i, such that } y_i = -1$$

Together: $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 0$

Property: if there is a hyperplane separating the examples, the linear program finds the solution

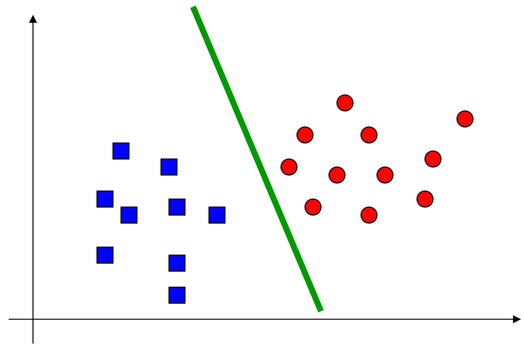
Linearly separable classes

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

Hyperplane:

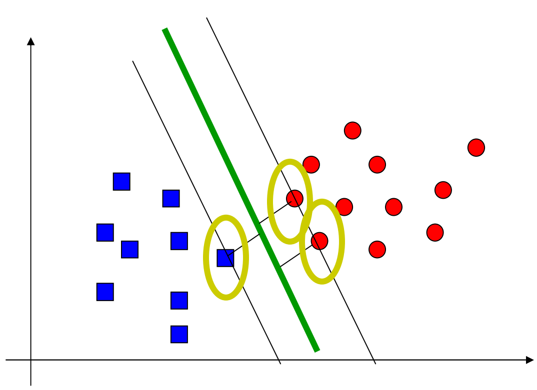
$$\mathbf{w}^T \mathbf{x} + w_0 = 0$$

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)

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

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:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$

$\hat{\alpha}_i$ 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:**

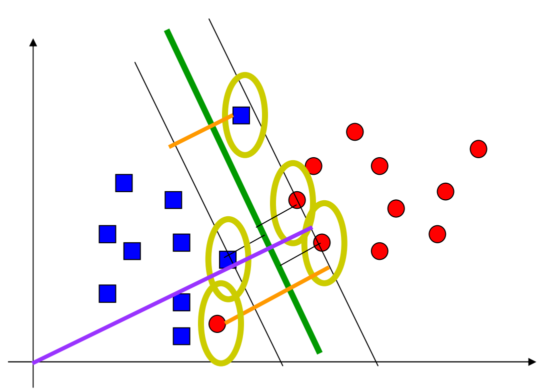
$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

Note:

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

Extension to a linearly non-separable case

- **Idea:** Allow some flexibility on crossing the separating hyperplane



Extension to the linearly non-separable case

- Relax constraints with variables $\xi_i \geq 0$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$

- Error occurs if $\xi_i \geq 1$, $\sum_{i=1}^{n} \xi_i$ is the upper bound on the number of errors
- Introduce a penalty for the errors

minimize $\|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^{n} \xi_i$ **Hinge loss**

Regularization penalty

Subject to constraints

C – set by a user, larger C leads to a larger penalty for an error

Extension to linearly non-separable case

- Lagrange multiplier form (primal problem)

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

- Dual form after $\mathbf{w}, w_0$ are expressed (ξ_i s cancel out)

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

Subject to: $0 \leq \alpha_i \leq C$ for all i, and $\sum_{i=1}^{n} \alpha_i y_i = 0$

Solution: $\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$

The difference from the separable case: $0 \leq \alpha_i \leq C$

The parameter w_0 is obtained through KKT conditions

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:**

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

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

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

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

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

- 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

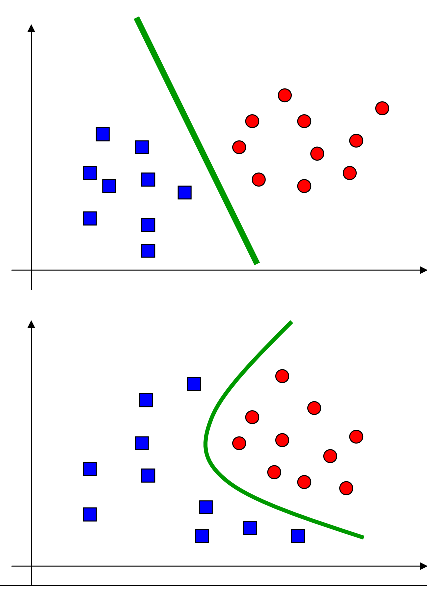
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

- Kernel function for the feature space:

$$\begin{aligned} K(\mathbf{x}', \mathbf{x}) &= \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1x_2x'_1\, x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1 \\ &= (x_1x'_1 + x_2x'_2 + 1)^2 \\ &= (1 + (\mathbf{x}^T\mathbf{x}'))^2 \end{aligned}$$

- 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**