Lecture 7: Kernels for Classification and Regression
CS 194-10, Fall 2011

Laurent El Ghaoui
EECS Department
UC Berkeley

September 15, 2011
Outline

Motivations

Linear classification and regression
   Examples
   Generic form

The kernel trick
   Linear case
   Nonlinear case

Examples
   Polynomial kernels
   Other kernels
   Kernels in practice
Motivations

Linear classification and regression
  Examples
  Generic form

The kernel trick
  Linear case
  Nonlinear case

Examples
  Polynomial kernels
  Other kernels
  Kernels in practice
A linear regression problem

Linear auto-regressive model for time-series: $y_t$ linear function of $y_{t-1}, y_{t-2}$

$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2}, \quad t = 1, \ldots, T.$$  

This writes $y_t = w^T x_t$, with $x_t$ the “feature vectors”

$$x_t := (1, y_{t-1}, y_{t-2}), \quad t = 1, \ldots, T.$$  

Model fitting via least-squares:

$$\min_w \| X^T w - y \|_2^2$$

*Prediction rule*: $\hat{y}_{T+1} = w_1 + w_2 y_T + w_3 y_{T-1} = w^T x_{T+1}$. 
Nonlinear regression

Nonlinear auto-regressive model for time-series: \( y_t \) quadratic function of \( y_{t-1}, y_{t-2} \)

\[
y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2} + w_4 y_{t-1}^2 + w_5 y_{t-1} y_{t-2} + w_6 y_{t-2}^2. 
\]

This writes \( y_t = w^T \phi(x_t) \), with \( \phi(x_t) \) the augmented feature vectors

\[
\phi(x_t) := \left(1, y_{t-1}, y_{t-2}, y_{t-1}^2, y_{t-1} y_{t-2}, y_{t-2}^2 \right).
\]

Everything the same as before, with \( x \) replaced by \( \phi(x) \).
Non-linear classification

Non-linear (e.g., quadratic) decision boundary

\[ w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + b = 0. \]

Writes \( w^T \phi(x) + b = 0 \), with \( \phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2) \).
Challenges

In principle, it seems can always augment the dimension of the feature space to make the data linearly separable. (See the video at http://www.youtube.com/watch?v=3liCbRZPrZA)

How do we do it in a computationally efficient manner?
Outline

Motivations

Linear classification and regression
  Examples
  Generic form

The kernel trick
  Linear case
  Nonlinear case

Examples
  Polynomial kernels
  Other kernels
  Kernels in practice
Linear least-squares

\[
\min_w \|X^T w - y\|_2^2 + \lambda \|w\|_2^2
\]

where

- \( X = [x_1, \ldots, x_n] \) is the \( m \times n \) matrix of data points.
- \( y \in \mathbb{R}^m \) is the “response” vector,
- \( w \) contains regression coefficients.
- \( \lambda \geq 0 \) is a regularization parameter.

**Prediction rule:** \( y = w^T x \), where \( x \in \mathbb{R}^n \) is a new data point.
Support vector machine (SVM)

$$\min_w \sum_{i=1}^{m} (1 - y_i (w^T x_i + b)) + \lambda \|w\|_2^2$$

where

- \(X = [x_1, \ldots, x_m]\) is the \(n \times m\) matrix of data points in \(\mathbb{R}^n\).
- \(y \in \{-1, 1\}^m\) is the label vector.
- \(w, b\) contain classifier coefficients.
- \(\lambda \geq 0\) is a regularization parameter.

In the sequel, we’ll ignore the bias term (for simplicity only).

**Classification rule:** \(y = \text{sign}(w^T x + b)\), where \(x \in \mathbb{R}^n\) is a new data point.
Generic form of problem

Many classification and regression problems can be written
\[
\min_w \ L(X^T w, y) + \lambda \| w \|^2_2
\]
where
- \( X = [x_1, \ldots, x_n] \) is a \( m \times n \) matrix of data points.
- \( y \in \mathbb{R}^m \) contains a response vector (or labels).
- \( w \) contains classifier coefficients.
- \( L \) is a “loss” function that depends on the problem considered.
- \( \lambda \geq 0 \) is a regularization parameter.

**Prediction/classification rule:** depends only on \( w^T x \), where \( x \in \mathbb{R}^n \) is a new data point.
Loss functions

- Squared loss: (for linear least-squares regression)
  \[ L(z, y) = \|z - y\|_2^2. \]

- Hinge loss: (for SVMs)
  \[ L(z, y) = \sum_{i=1}^{m} \max(0, 1 - y_i z_i) \]

- Logistic loss: (for logistic regression)
  \[ L(z, y) = - \sum_{i=1}^{m} \log(1 + e^{-y_i z_i}). \]
Outline

Motivations

Linear classification and regression
  Examples
  Generic form

The kernel trick
  Linear case
  Nonlinear case

Examples
  Polynomial kernels
  Other kernels
  Kernels in practice
Key result

For the generic problem:

$$\min_{w} \ L(X^T w) + \lambda \|w\|_2^2$$

the optimal $w$ lies in the span of the data points $(x_1, \ldots, x_m)$:

$$w = Xv$$

for some vector $v \in \mathbb{R}^m$. 
Proof

Any \( w \in \mathbb{R}^n \) can be written as the sum of two orthogonal vectors:

\[
w = Xv + r
\]

where \( X^T r = 0 \) (that is, \( r \) is in the nullspace \( \mathcal{N}(X^T) \)).

Figure shows the case \( X = A = (a_1, a_2) \).
Consequence of key result

For the generic problem:

$$\min_w L(X^T w) + \lambda \|w\|^2_2$$

the optimal $w$ can be written as $w = Xv$ for some vector $v \in \mathbb{R}^m$.

Hence training problem depends only on $K := X^TX$:

$$\min_v L(Kv) + \lambda v^TKv.$$
Kernel matrix

The training problem depends only on the “kernel matrix” \( K = X^T X \)

\[ K_{ij} = x_i^T x_j \]

\( K \) contains the scalar products between all data point pairs.

The prediction/classification rule depends on the scalar products between new point \( x \) and the data points \( x_1, \ldots, x_m \):

\[ w^T x = v^T X^T x = v^T k, \quad k := X^T x = (x^T x_1, \ldots, x^T x_m). \]
Computational advantages

Once $K$ is formed (this takes $O(n)$), then the training problem has only $m$ variables.

When $n \gg m$, this leads to a dramatic reduction in problem size.
How about the nonlinear case?

In the nonlinear case, we simply replace the feature vectors $x_i$ by some “augmented” feature vectors $\phi(x_i)$, with $\phi$ a non-linear mapping.

**Example**: in classification with quadratic decision boundary, we use

$$\phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2).$$

This leads to the modified kernel matrix

$$K_{ij} = \phi(x_i)^T \phi(x_j), \ 1 \leq i, j \leq m.$$
The kernel function

The kernel function associated with mapping \( \phi \) is

\[
k(x, z) = \phi(x)^T \phi(z).
\]

It provides information about the metric in the feature space, e.g.:

\[
\|\phi(x) - \phi(z)\|_2^2 = k(x, x) - 2k(x, z) + k(z, z).
\]

The computational effort involved in

- solving the training problem;
- making a prediction,

depends only on our ability to quickly evaluate such scalar products.

We can’t choose \( k \) arbitrarily; it has to satisfy the above for some \( \phi \).
Outline

Motivations

Linear classification and regression
  Examples
  Generic form

The kernel trick
  Linear case
  Nonlinear case

Examples
  Polynomial kernels
  Other kernels
  Kernels in practice
Quadratic kernels

Classification with quadratic boundaries involves feature vectors

\[ \phi(x) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2). \]

**Fact**: given two vectors \( x, z \in \mathbb{R}^2 \), we have

\[ \phi(x)^T \phi(z) = (1 + x^T z)^2. \]
Polynomial kernels

More generally when $\phi(x)$ is the vector formed with all the products between the components of $x \in \mathbb{R}^n$, up to degree $d$, then for any two vectors $x, z \in \mathbb{R}^n$,

$$\phi(x)^T \phi(z) = (1 + x^T z)^d.$$  

Computational effort grows linearly in $n$.

This represents a dramatic reduction in speed over the “brute force” approach:

- Form $\phi(x), \phi(z)$;
- evaluate $\phi(x)^T \phi(z)$.

Computational effort grows as $n^d$. 
Other kernels

Gaussian kernel function:

\[ k(x, z) = \exp \left( -\frac{\|x - z\|^2}{2\sigma^2} \right), \]

where \( \sigma > 0 \) is a scale parameter. Allows to ignore points that are too far apart. Corresponds to a non-linear mapping \( \phi \) to infinite-dimensional feature space.

There is a large variety (a zoo?) of other kernels, some adapted to structure of data (text, images, etc).
In practice

- Kernels need to be chosen by the user.
- Choice not always obvious; Gaussian or polynomial kernels are popular.
- Control over-fitting via cross validation (wrt say, scale parameter of Gaussian kernel, or degree of polynomial kernel).
- Kernel methods not well adapted to $l_1$-norm regularization.