# Optimization Models

## EECS 127 / EECS 227AT

Laurent El Ghaoui

EECS department
UC Berkeley

Fall 2018

# LECTURE 22a

# Applications to Machine Learning (I): Supervised Learning

Learning:

---

Merriam–Webster Dictionary

# Outline

# What is supervised learning?

In supervised learning, we are given

- A matrix of data points $X = [x_1, \ldots, x_m]$, with $x_i \in \mathbf{R}^n$;
- A vector of "responses" $y \in \mathbf{R}^m$.

The goal is to use the data and the information in $y$ (the "supervised" part) to form a model. In turn the model can be used to *predict* a value $\hat{y}$ for a (yet unseen) new data point $x \in \mathbf{R}^n$.

# Generic form of problem

Many classification and regression problems can be written

$$\min_w \; L(X^T w, y) + \lambda p(w)$$

where

- $X = [x_1, \ldots, x_n]$ is a $m \times n$ matrix of data points.
- $y \in \mathbf{R}^m$ contains a response vector (or labels).
- $w$ contains classifier/regression coefficients.
- $L$ is a "loss" function that depends on the problem considered.
- $p$ is a *penalty function* that is usually data-independent (more on this later).
- $\lambda \geq 0$ is a regularization parameter.

*Prediction/classification rule:* depends only on $w^T x$, where $x \in \mathbf{R}^n$ is a new data point.

# Popular 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)

$$L(z, y) = -\sum_{i=1}^{m} \log(1 + e^{-y_i z_i}).$$

# Linear least-squares

*Basic model:*

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

where

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

*Prediction rule:* $y = w^T x$, where $x \in \mathbf{R}^n$ is a new data point.

# Example

## 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:* we minimize the sum-of-squares of errors

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

*Prediction rule*: once we've "learnt" $w$, we can make a prediction for time $T+1$: $\hat{y}_{T+1} = w_1 + w_2 y_T + w_3 y_{T-1} = w^T x_{T+1}$.

# Regularized least-squares

Often a penalty term is added, as in (Tikhonov) regularization:

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

where $\lambda$ is usually chosen via cross-validation.

*Motivation:* Assume that $X$ is additively perturbed by a matrix $U$ with independent Gaussian entries, with mean 0, and variance $\lambda$. The above corresponds to minimizing the expected value of the objective, since

$$\textbf{Expect}_U \ \|(X + U)^T w - y\|_2^2 = \|X^T w - y\|_2^2 + \lambda \|w\|_2^2.$$

# Robust Least-Squares

Another motivation comes from the robust least-squares problem

$$\min_{w} \max_{U\,:\,\|U\|\leq\lambda} \|(X+U)^T w - y\|_2,$$

where $\|\cdot\|$ stands for the largest singular value of its matrix argument.

The above can be expressed as

$$\min_{w} \|X^T w - y\|_2 + \lambda\|w\|_2,$$

As $\lambda$ scans $\mathbf{R}_+$, the path of solutions is the same as with regularized LS.

# LASSO

The LASSO model is

$$\min_{w} \ \|X^T w - y\|_2^2 + \lambda \|w\|_1,$$

where $\|\cdot\|_1$ is the $l_1$-norm (sum of absolute values) of its vector argument.

- Observed to produce very sparse solutions (many zeroes in $w$).
- The above has been extensively studied under various names (*e.g.*, compressed sensing).
- Can be motivated by component-wise uncertainty model on $X$.
- Can be solved by convex optimization methods.

# CVX syntax

Here is a matlab snippet that solves a LASSO problem via CVX, given $n \times m$ matrix $X$, $n$-vector $y$ and non-negative scalar $\lambda$ exist in the workspace:

```
cvx_begin
variable w(n,1);
variable r(m,1);
minimize( r'*r + lambda*norm(w,1))
subject to
r == X'*w-y;
cvx_end
```

- Can also use the syntax `(X'*w-y)'*(X'*w-y)` or `square_pos(norm(X'*w-y,2))`
- However the objective `norm(X'*w-y,2)^2` will be rejected by CVX, as it does know that what is squared is non-negative (the square of a *non-negative* convex function is convex).

# Binary classification via SVM

## Data

We are given a *training* data set:

- Feature vectors: data points $x_i \in \mathbf{R}^p$, $i = 1, \ldots, n$.
- Labels: $y_i \in \{-1, 1\}$, $i = 1, \ldots, n$.

Examples:

| Feature vectors | Labels |
|---|---|
| Companies' corporate info | default/no default |
| Stock price data | price up/down |
| News data | price up/down |
| News data | sentiment (positive/negative) |
| Emails | presence of a keyword |
| Genetic measures | presence of disease |

# Linear classification

Using the training data set $\{x_i, y_i\}_{i=1}^n$, our goal is to find a classification rule $\hat{y} = f(x)$ allowing to predict the label $\hat{y}$ of a new data point $x$.

*Linear classification rule:* assumes $f$ is a combination of the sign function and a linear (in fact, affine) function:

$$\hat{y} = \mathbf{sign}(w^T x + b),$$

where $w \in \mathbf{R}^p$, $b \in \mathbf{R}$ are given.

The goal of a linear classification algorithm is to find $w, b$, using the training data.

# Separable data

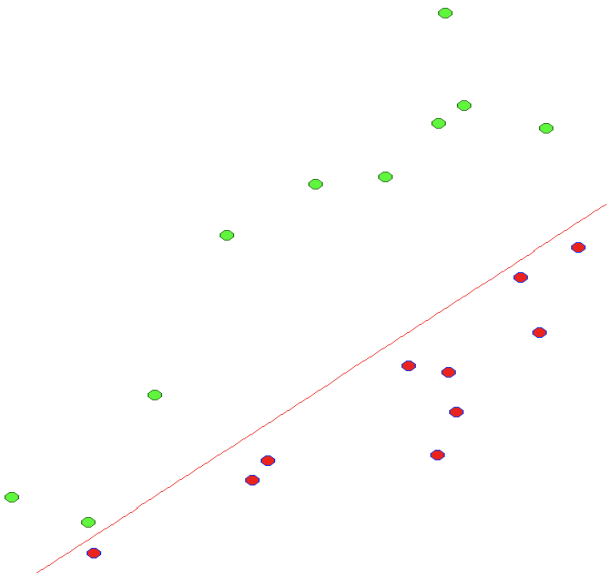
The data is linearly separable if there exist a linear classification rule that makes no error on the training set.

This is a set of linear inequalities constraints on $(w, b)$:

$$y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, n.$$

*Strict separability* corresponds the the same conditions, but with strict inequalities.

# Geometry

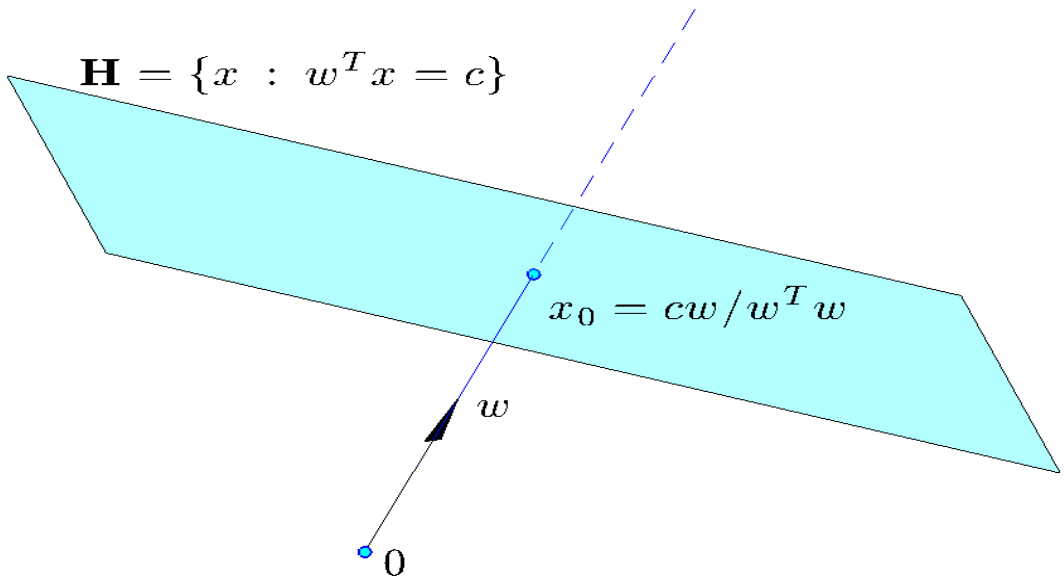


*Geometrically:* the hyperplane

$$\{x \ : \ w^T x + b = 0\}$$

perfectly separates the positive and negative data points.

# Linear algebra flashback: hyperplanes



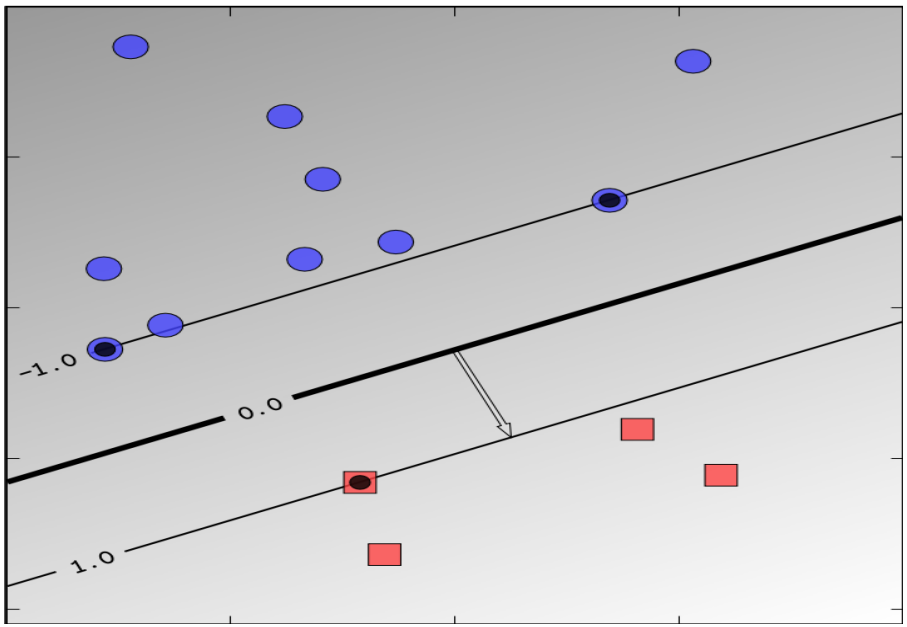

Geometrically, a hyperplane $\mathbf{H} = \{w \ : \ w^T x = c\}$ is a translation of the set of vectors orthogonal to $w$. The direction of the translation is determined by $w$, and the amount by $c/\|w\|_2$. Indeed, the projection of 0 onto $\mathbf{H}$ is $x_0 = cw/(w^T w)$.

# Geometry (cont'd)

Assuming strict separability, we can always rescale $(w, b)$ and work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n.$$

Amounts to make sure that negative (resp. positive) class contained in half-space $w^T x + b \leq -1$ (resp. $w^T x + b \geq 1$).




The distance between the two "$\pm 1$" boundaries turns out the be equal to $2/\|w\|_2$.

Thus the "margin" $\|w\|_2$ is a measure of how well the hyperplane separates the data apart.

# Non-separable data

Separability constraints are homogeneous, so WLOG we can work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n.$$

If the above is infeasible, we try to minimize the "slacks"

$$\min_{w,b,s} \sum_{i=1}^{n} s_i \;\; : \;\; s \geq 0, \quad y_i(w^T x_i + b) \geq 1 - s_i, \quad i = 1, \ldots, n.$$

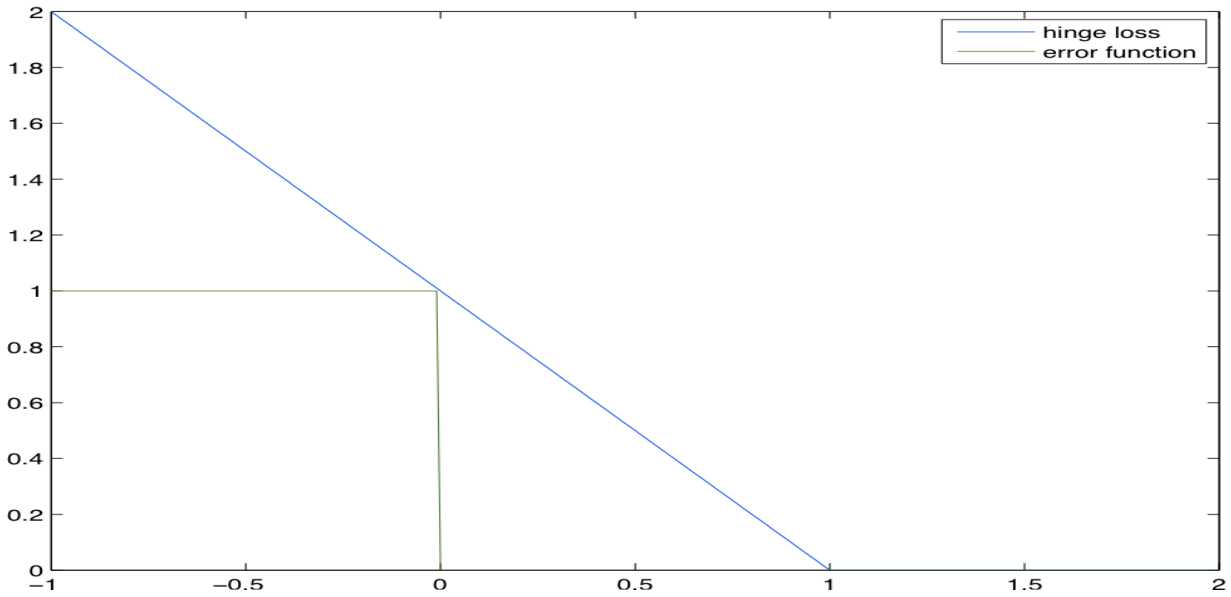
The above can be solved as a "linear programming" problem (in variables $w, b, s$).

# Hinge loss function

The previous LP can be interpreted as minimizing the hinge loss function

$$L(w, b) := \sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0).$$

This serves as an approximation to the number of errors made on the training set:



# Regularization

The solution might not be unique, so we add a regularization term $\|w\|_2^2$:

$$\min_{w,b} \ \frac{1}{2}\|w\|_2^2 + C \cdot L(w,b)$$

where $C > 0$ allows to trade-off the accuracy on the training set and the prediction error (more on why later). This makes the solution unique.

The above model is called the *Support Vector Machine*. It is a quadratic program (Q). It can be reliably solved using special fast algorithms that exploit its structure.

If $C$ is large, and data is separable, reduces to the maximal-margin problem

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \ : \ y_i(w^T x_i + b) \geq 1, \ \ i = 1, \ldots, n.$$

# CVX syntax

CVX allows to bypass the need to put the problem in standard format.
Here is a matlab snippet that solves an SVM problem, given $n \times m$ matrix $X$, $n$-vector $y$ and non-negative scalar $C$ exist in the workspace:

```
cvx_begin
variable w(n,1);
variable b(1);
minimize( C*sum(max(0,1-y.*(X'*w+b)))+ .5*w'*w)
cvx_end
```

# Robustness interpretation

Return to separable data. The set of constraints

$$y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, n,$$

has many possible solutions $(w, b)$.

We will select a solution based on the idea of robustness (to changes in data points).

# Maximally robust separating hyperplane

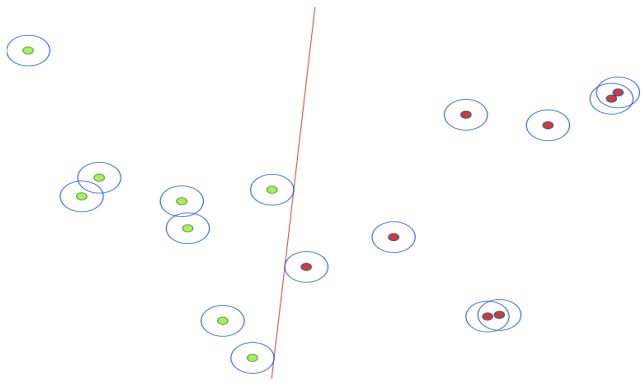
*Spherical uncertainty model:* assume that the data points are actually unknown, but bounded:

$$x_i \in \mathcal{S}_i := \{\hat{x}_i + u_i \ : \ \|u_i\|_2 \leq \rho\},$$

where $\hat{x}_i$'s are known, $\rho > 0$ is a given measure of uncertainty, and $u_i$ is unknown.

*Robust counterpart:* we now ask that the separating hyperplane separates the spheres (and not just the points):

$$\forall x_i \in \mathcal{S}_i \ : \ y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, n.$$



For separable data we can try to separate spheres around the given points. We'll grow the spheres' radius until sphere separation becomes impossible.

# Robust classification

We obtain the equivalent condition

$$y_i(w^T \hat{x}_i + b) \geq \rho \|w\|_2, \quad i = 1, \ldots, n.$$

Now we seek $(w, b)$ which maximize $\rho$ subject to the above.

By homogeneity we can always set $\rho\|w\|_2 = 1$ , so that problem reduces to

$$\min_w \ \|w\|_2 \ : \ y_i(w^T \hat{x}_i + b) \geq 1, \quad i = 1, \ldots, n.$$

This is exactly the same problem as the SVM in separable case.

# Dual problem

Denote by $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) the set of points $x_i$ with $y_i = +1$ (resp. $-1$).

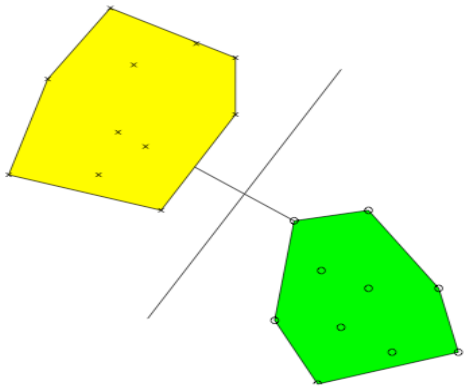
It can be shown that the dual of the SVM problem can be expressed as:

$$\min_{x_+, x_-} \|x_+ - x_-\|_2 \ : \ x_+ \in \mathbf{Co}\mathcal{C}^+, \ \ x_- \in \mathbf{Co}\mathcal{C}^-,$$

where $\mathbf{Co}\mathcal{C}$ denotes convex hull of set $\mathcal{C}$, that is:

$$\mathbf{Co}\mathcal{C} = \left\{ \sum_{i=1}^{q} \lambda_i x_i \ : \ x_i \in \mathcal{C}, \ \ \lambda \geq 0, \ \ \sum_{i=1}^{q} \lambda_i = 1 \right\}.$$

# Geometry



Dual problem amounts to find the smallest distance between the two classes, each represented by the convex hull of its points. The optimal hyperplane sits at the middle of the line segment joining the two closest points.

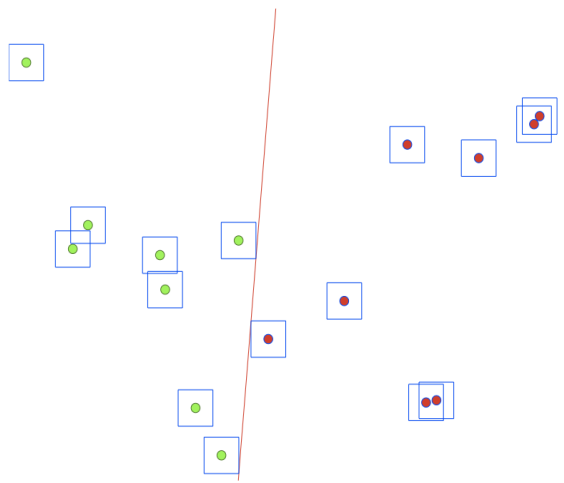
# Separating boxes instead of spheres

We can use a box uncertainty model:

$$x_i \in \mathcal{B}_i := \{\hat{x}_i + u_i \ : \ \|u_i\|_\infty \leq \rho\}\,.$$

This leads to

$$\min_w \ \|w\|_1 \ : \ y_i(w^T \hat{x}_i + b) \geq 1, \quad i = 1, \ldots, n.$$



Classifiers found that way tend to be sparse. In 2D, the boundary line tends to be vertical or horizontal.

# CVX syntax

Here is a matlab snippet that solves an SVM problem with $l_1$ penalty, given $n \times m$ matrix $X$, $n$-vector $y$ and non-negative scalar $C$ exist in the workspace:

```
cvx_begin
variable w(n,1);
variable b(1);
minimize( C*sum(max(0,1-y.*(X'*w+b)))+ norm(w,1))
cvx_end
```

# Logistic model

We model the probability of a label $Y$ to be equal $y \in \{-1, 1\}$, given a data point $x \in \mathbf{R}^n$, as:

$$P(Y = y \mid x) = \frac{1}{1 + \exp(-y(w^T x + b))}.$$

This amounts to modeling the *log-odds ratio* as a linear function of $X$:

$$\log \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} = w^T x + b.$$

- The decision boundary $P(Y = 1 \mid x) = P(Y = -1 \mid x)$ is the hyperplane with equation $w^T x + b = 0$.
- The region $P(Y = 1 \mid x) \geq P(Y = -1 \mid x)$ (*i.e.*, $w^T x + b \geq 0$) corresponds to points with predicted label $\hat{y} = +1$.

# Maximum-likelihood

The likelihood function is

$$l(w, b) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(w^T x_i + b)}}.$$

Now maximize the log-likelihood:

$$\max_{w,b} \; L(w, b) := -\sum_{i=1}^{m} \log(1 + e^{-y_i(w^T x_i + b)})$$

In practice, we may consider adding a regularization term

$$\max_{w,b} \; L(w, b) + \lambda r(w),$$

with $r(w) = \|w\|_2^2$ or $r(x) = \|w\|_1$.

# Learning problems seen so far

Least-squares linear regression, SVMs, and logistic regression problems can all be expressed as minimization problems:

$$\min_{w} \; f(w, b)$$

with

- $f(w, b) = \|X^T w - y\|_2^2$ (square loss);
- $f(w, b) = \sum_{i=1}^{m} \max(0, 1 - y_i(w^T x_i + b))$ (hinge loss);
- $f(w, b) = -\sum_{i=1}^{m} \log(1 + \exp(-y_i(w^T x_i + b)))$ (logistic loss).

The regularized version involves an additional penalty term.

# 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 $y_{t-1}, y_{t-2}$

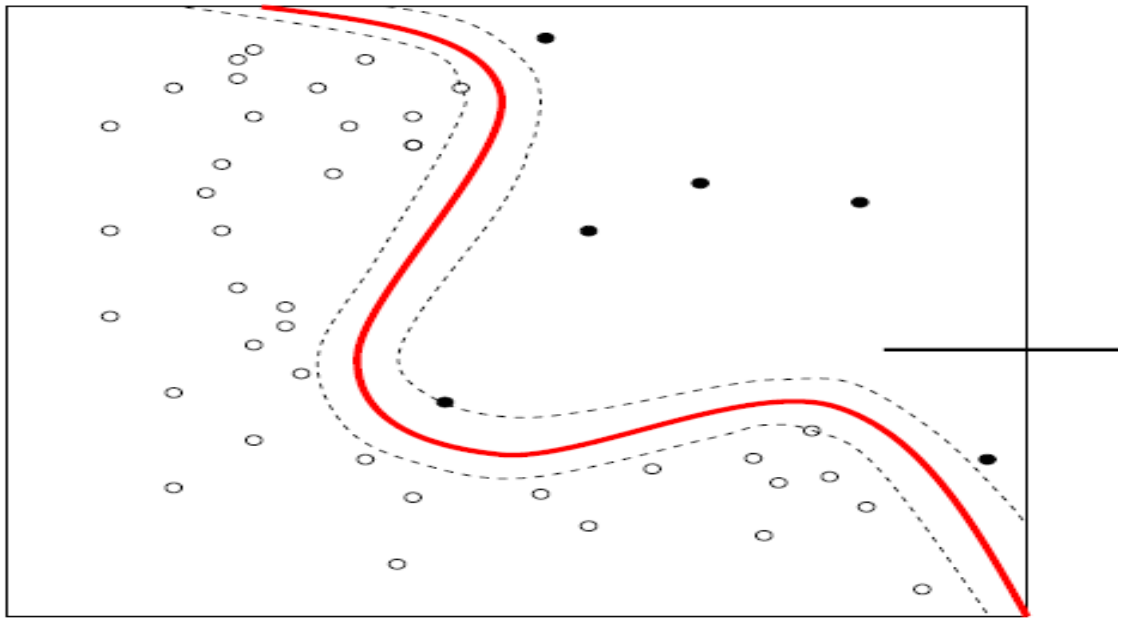
$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2} + w_4 y_{t-1}^2 + w_5 y_{t-1} y_{t-2} + w_6 y_{t-2}^2.$$

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 $\phi(x)$.

# Nonlinear classification



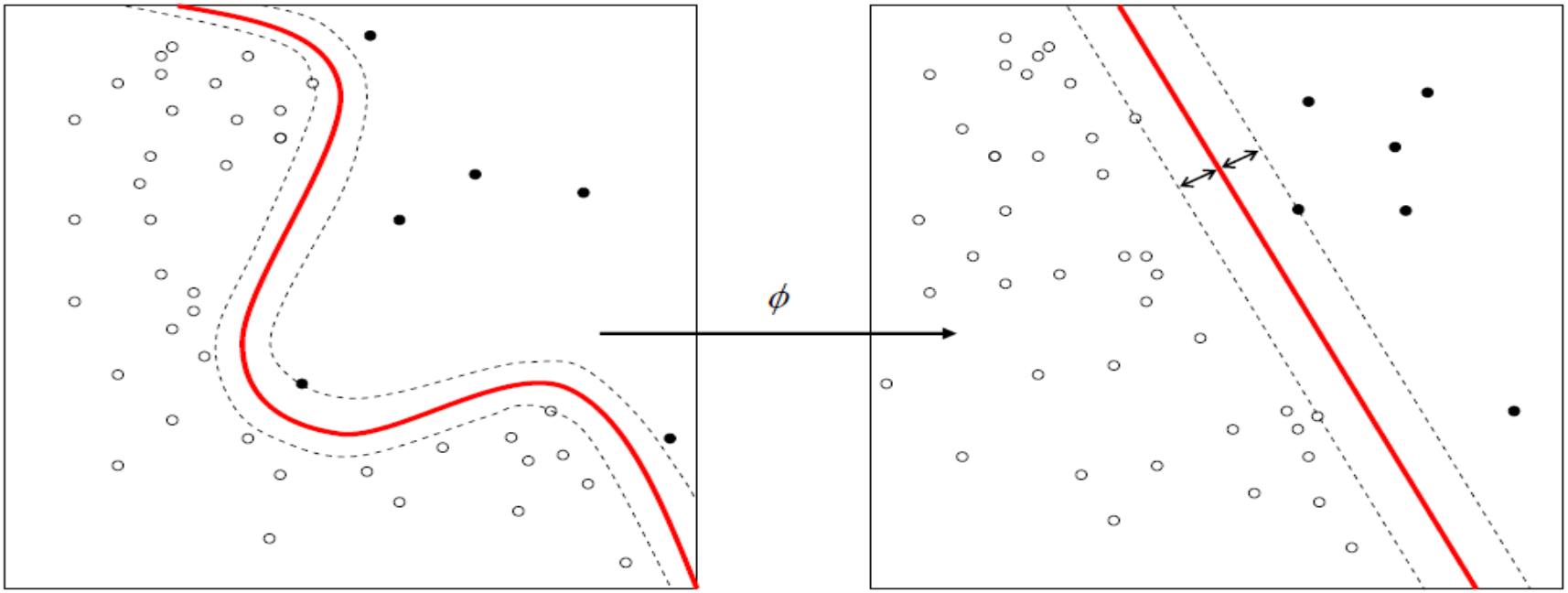
Non-linear (*e.g.*, quadratic) decision boundary

$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + b = 0.$$

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?

# Generic form of learning 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.
- $y \in \mathbf{R}^m$ 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 $x \in \mathbf{R}^n$ is a new data point.

Note: from now on, we consider $l_2$ penalties only!

# 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)

$$L(z, y) = -\sum_{i=1}^{m} \log(1 + e^{-y_i z_i}).$$

# 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 $v \in \mathbf{R}^m$.

# Proof

Any $w \in \mathbf{R}^n$ 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 $\mathcal{N}(X^T)$).

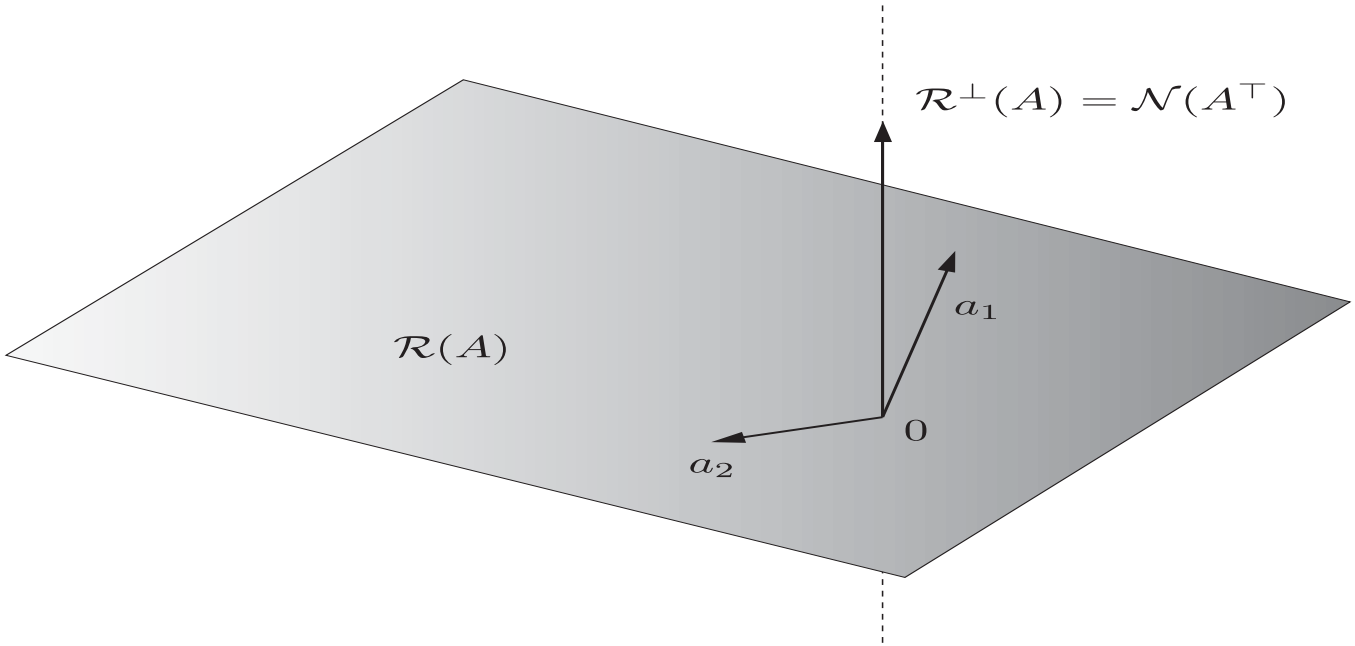



Figure shows the case $X = A = (a_1, a_2)$.

# 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 $v \in \mathbf{R}^m$.

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

$$\min_v \; L(Kv) + \lambda v^T K v.$$

# 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 $n >> m$, 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 $\phi$.

# 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 $x, z \in \mathbf{R}^2$, 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 $x \in \mathbf{R}^n$, up to degree $d$, then for any two vectors $x, z \in \mathbf{R}^n$,

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

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:

$$k(x, z) = \exp\left(-\frac{\|x - z\|_2^2}{2\sigma^2}\right),$$

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.

# References


S. Boyd and M. Grant.

The CVX optimization package, 2010.

Chih-Chung Chang and Chih-Jen Lin.

SVM-LIB: A library for SVM classification.

S. Sra, S.J. Wright, and S. Nowozin.

*Optimization for Machine Learning*.
MIT Press, 2011.