Optimization Models

EECS 127 / EECS 227AT

Laurent El Ghaoui

EECS department
UC Berkeley

Fall 2018

LECTURE 8

Least-Squares and Variants

If others would but reflect on mathematical truths as deeply and continuously as I have, they would make my discoveries.

C.F. Gauss (1777 – 1855)

Outline

Least-squares

Goal: given $A \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$, find x such that $Ax \approx y$.

Least-squares approach: use Euclidean norm, and solve the optimization problem

$$\min_x \ \|Ax - y\|_2$$

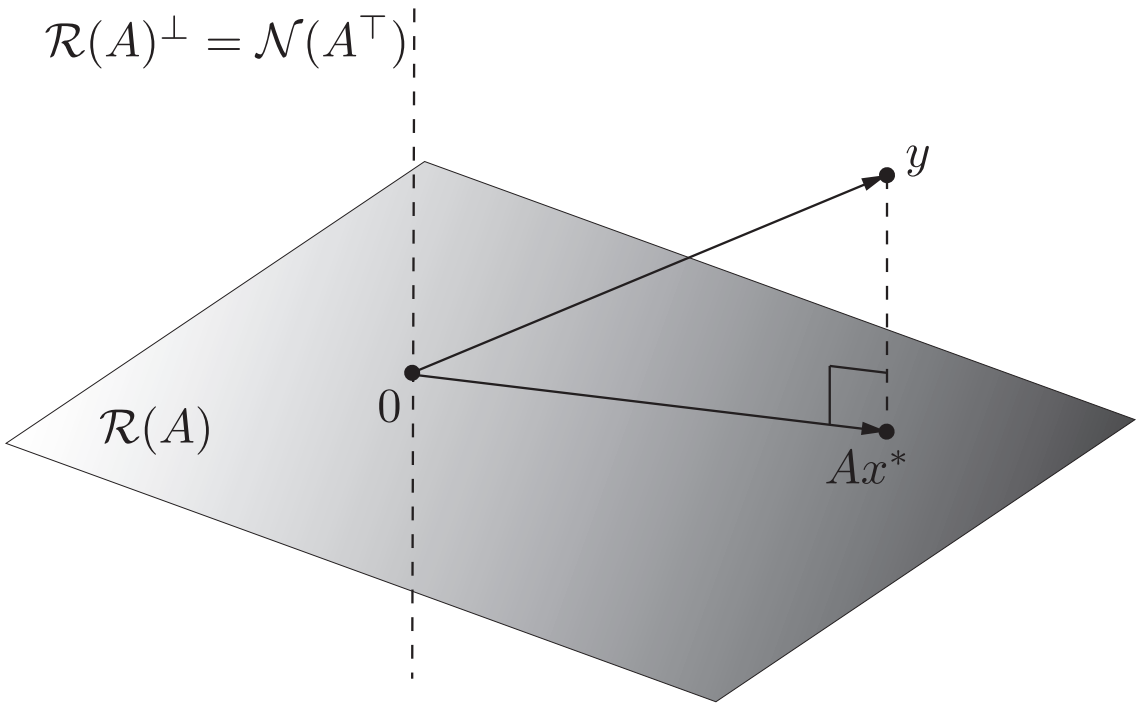
Since the objective function is always ≥ 0, we can solve the "ordinary least-squares" problem

$$\min_x \ \|Ax - y\|_2^2 = \sum_{i=1}^m r_i^2, \quad r \doteq Ax - y.$$

Geometric interpretation

As projection on a subspace

- Since vector Ax lies in $\mathcal{R}(A)$, the problem amounts to determining a point $\tilde{y} = Ax^*$ in $\mathcal{R}(A)$ at *minimum distance* form y.
- The Projection Theorem then tells us that this point is indeed the orthogonal projection of y onto the subspace $\mathcal{R}(A)$.



Interpretation

As fitting a linear model

We can also interpret the problem in terms of the rows $a_i^\top$, $i = 1, \ldots, m$, of A. The problem reads

$$\min_x : \sum_{i=1}^{m} (y_i - a_i^\top x)^2.$$

In this sense, we are trying to fit of each component y_i as a linear combination of the corresponding input a_i, with x as the coefficients of this linear combination.

This interpretation is useful in the context of prediction: once the solution x^* to the above is found, we can "predict" the output corresponding to a new vector $a \in \mathbb{R}^n$ via the prediction rule

$$\hat{y} = a^\top x^*.$$

Least-squares

Solution

- $y - Ax^* \in \mathcal{R}(A)^\perp = \mathcal{N}(A^\top)$, hence

$$A^\top (y - Ax^*) = 0$$

- Solutions x^* to the LS problem must satisfy the Normal Equations:

$$A^\top Ax = A^\top y$$

- This system *always* admits a solution.
- If A is full column rank (i.e., $\mathrm{rank}(A) = n$), then the solution is unique, and it is given by

$$x^* = (A^\top A)^{-1} A^\top y.$$

Set of solutions and the pseudoinverse

Corollary 1 (Set of solutions of LS problem)

The set of optimal solutions of the LS problem

$$p^* = \min_x \|Ax - y\|_2$$

can be expressed as

$$\mathcal{X}_{\text{opt}} = A^\dagger y + \mathcal{N}(A),$$

where $A^\dagger y$ is the minimum-norm point in the optimal set. The optimal value p^* is the norm of the projection of y onto orthogonal complement of $\mathcal{R}(A)$: for $x^* \in \mathcal{X}_{\text{opt}}$,

$$p^* = \|y - Ax^*\|_2 = \|(I_m - AA^\dagger)y\|_2 = \|P_{\mathcal{R}(A)^\perp} y\|_2,$$

where matrix $P_{\mathcal{R}(A)^\perp}$ is the projector onto $\mathcal{R}(A)^\perp$. If A is full column rank, then the solution is unique, and equal to

$$x^* = A^\dagger y = (A^\top A)^{-1} A^\top y.$$

Minimum-norm solutions to linear equations

- When matrix A has more columns than rows ($m < n$: underdetermined), and $y \in \mathcal{R}(A)$, we have that $\dim \mathcal{N}(A) \geq n - m > 0$, hence the system $y = Ax$ has infinite solutions and that the set of solutions is $\mathcal{S}_{\bar{x}} = \{x : x = \bar{x} + z,\ z \in \mathcal{N}(A)\}$, where $\bar{x}$ is any vector such that $A\bar{x} = y$.
- We single out from $\mathcal{S}_{\bar{x}}$ the one solution x^* with minimal Euclidean norm. That is, we solve

$$\min_{x:Ax=y} \|x\|_2,$$

  which is equivalent to $\min_{x \in \mathcal{S}_{\bar{x}}} \|x\|_2$.
- The solution x^* must be orthogonal to $\mathcal{N}(A)$ or, equivalently, $x^* \in \mathcal{R}(A^\top)$, which means that $x^* = A^\top \xi$, for some suitable ξ.
- Since x^* must solve the system of equations, it must be $Ax^* = y$, i.e., $AA^\top \xi = y$.
- If A is full row rank, $AA^\top$ is invertible and the unique ξ that solves the previous equation is $\xi = (AA^\top)^{-1} y$. This finally gives us the unique minimum-norm solution of the system:

$$x^* = A^\top (AA^\top)^{-1} y.$$

Solution of OLS via QR

Assume columns of $m \times n$ A are linearly independent, hence QR factorization $A = QR$ exists, with

- $m \times n$ matrix Q satisfies $Q^\top Q = I$;
- $n \times n$ matrix R invertible;
- then $A^\dagger = (A^\top A)^{-1} A^\top = R^{-1} Q^\top$.

Algorithm:

- compute QR factorization of $m \times n$ A: $A = QR$ ($2mn^2$ flops);
- form $z = Q^\top y$ ($2mn$ flops);
- solve triangular system $Rx = z$ via backward substitution.

Results:

- total complexity $2mn^2$ flops;
- identical to algorithm for solving $Ax = b$ for square invertible A;
- when A is tall ($m >> n$), gives least squares approximate solution to $Ax = b$.

Example

Advertising purchases[1]

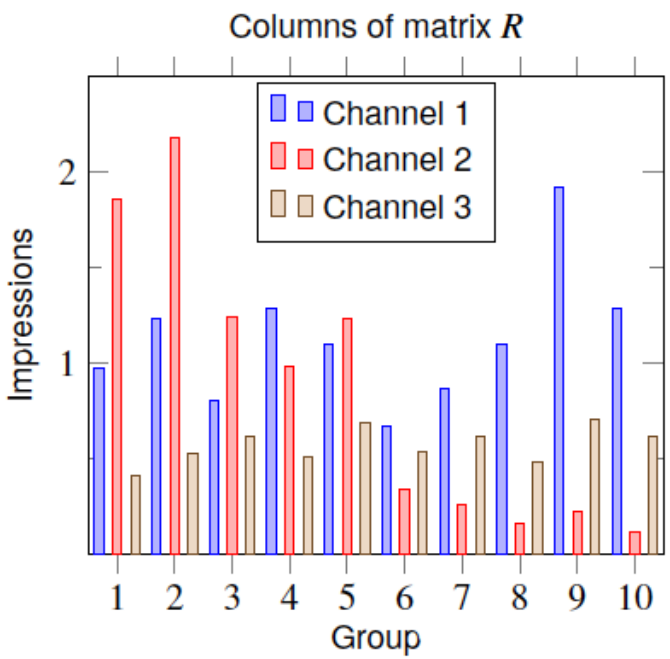
- m demographics groups we want to advertise to;
- v_{des} is m-vector of target views or impressions;
- n-vector s gives spending on n advertising channels;
- $m \times n$ matrix R gives demographic reach of channels, with R_{ij} the number of views per dollar spent (in 1000/\$);
- $v = Rs$ is m-vector of views across demographic groups;
- well use least squares spending: $\hat{v} = R^{\dagger} s$.

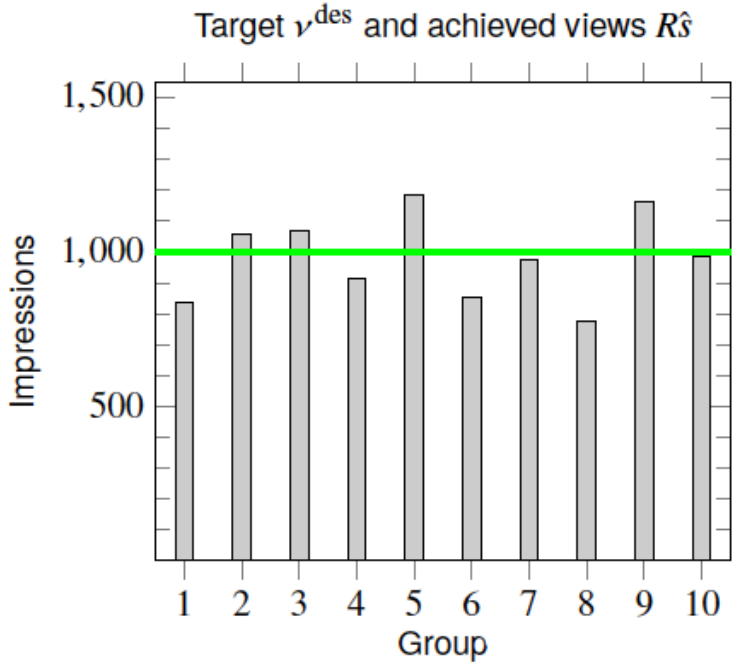
[1] from `http://web.stanford.edu/~boyd/vmls/vmls-slides.pdf`.

Example

Advertising purchases

- $m = 10$ groups, $n = 3$ channels
- target views vector $v_{\mathrm{des}} = 10^3\mathbf{1}$
- LS spending: $\hat{s} = (62, 100, 1443)$.





Variants of ordinary LS

Linear equality-constrained LS

- A generalization of the basic LS problem allows for the addition of linear equality constraints on the x variable, resulting in the constrained problem

$$\min_x \|Ax - y\|_2^2, \quad \text{s.t. } Cx = d,$$

  where $C \in \mathbb{R}^{p,n}$ and $d \in \mathbb{R}^p$.

- This problem can be converted into a standard LS one, by "eliminating" the equality constraints, via a standard procedure. Suppose the problem is feasible, and let $\bar{x}$ be such that $C\bar{x} = d$.
- All feasible points are expressed as $x = \bar{x} + Nz$, where N contains by columns a basis for $\mathcal{N}(C)$, and z is a new variable.
- Problem becomes unconstrained in variable z:

$$\min_z \|\bar{A}z - \bar{y}\|_2^2,$$

  where $\bar{A} \doteq AN$, $\bar{y} \doteq y - A\bar{x}$.

Variants of the least-squares problem

Weighted LS

- The standard LS objective is a sum of squared equation residuals

$$\|Ax - y\|_2^2 = \sum_{i=1}^{m} r_i^2, \quad r_i = a_i^\top x - y_i.$$

- In some cases, the equation residuals may not be given the same importance, and this relative importance can be modeled by introducing *weights* into the LS objective, that is $f_0(x) = \sum_{i=1}^m w_i^2 r_i^2$, where $w_i \geq 0$ are the given weights. This objective is rewritten as

$$f_0(x) = \|W(Ax - y)\|_2^2 = \|A_w x - y_w\|_2^2,$$

  where

$$W = \operatorname{diag}(w_1, \ldots, w_m), \quad A_w \doteq WA,\ y_w = Wy.$$

- The weighted LS problem still has the structure of a standard LS problem, with row-weighted matrix A_w and vector y_w.

Variants of the least-squares problem

ℓ_2-regularized LS

- Regularized LS refer to a class of problems of the form

$$\min_x \ \|Ax - y\|_2^2 + \phi(x),$$

  where a "regularization," or *penalty*, term $\phi(x)$ is added to the usual LS objective.

- In the most usual cases, ϕ is proportional either to the ℓ_1 or to the ℓ_2 norm of x. The ℓ_1-regularized case gives rise to the LASSO problem, which is discussed in more detail later. The ℓ_2-regularized case is instead discussed next:

$$\min_x \ \|Ax - y\|_2^2 + \lambda\|x\|_2^2, \quad \lambda \geq 0$$

Variants of the least-squares problem

ℓ_2-regularized LS

$$\min_x \ \|Ax - y\|_2^2 + \lambda \|x\|_2^2, \quad \lambda \geq 0$$

- Recalling that the squared Euclidean norm of a block-partitioned vector is equal to the sum of the squared norms of the blocks, i.e.,

$$\left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\|_2^2 = \|a\|_2^2 + \|b\|_2^2$$

  we see that the regularized LS problem can be rewritten in the format of a standard LS problem as follows

$$\|Ax - y\|_2^2 + \lambda \|x\|_2^2 = \|\tilde{A}x - \tilde{y}\|_2^2,$$

  where

$$\tilde{A} \doteq \begin{bmatrix} A \\ \sqrt{\lambda} I_n \end{bmatrix}, \quad \tilde{y} \doteq \begin{bmatrix} y \\ 0_n \end{bmatrix}.$$

- $\lambda \geq 0$ is a *tradeoff parameter.* Interpretation in terms of tradeoff between *output tracking accuracy* and *input effort*.

How to choose the regularization parameter?

Choosing a good value of λ is crucial. To each value corresponds a different prediction rule (that is, a model), where a new point a is given a predicted output $\hat{y} = a^\top x^*(\lambda)$, with

$$x^*(\lambda) := \arg\min_x \ \|Ax - y\|_2^2 + \lambda \|x\|_2^2.$$

For a given λ:

- split original data into a training set and a test set (typical split: 80% / 20%);
- build (train) model on training data set; then check the models predictions on the test data set;
- if they are similar, we can guess the model will work well on *unseen* data, a desirable "generalization" property.

Example

Auto-regressive models

- Auto-Regressive (AR) models try to describe a time series $y(k)$, $k = 0, 1, \ldots$, according to the model

$$y(k) = w_1 y(k-1) + \cdots + w_n y(k-n) + e(k),$$

where $e(k)$ is an error term, assumed to have zero mean.

- If we observe the outputs (regressors)

$$\varphi(k)^\top \doteq [y(k-1)\ y(k-2)\ \cdots\ y(k-n)]$$

and we know the model parameters $w^\top \doteq [w_1\ a_2\ \cdots\ w_n]$, we can *predict* the output value at time k, as

$$\hat{y}(k) = \varphi(k)^\top w.$$

- The *prediction error* is

$$\epsilon(k) = y(k) - \hat{y}(k) = y(k) - \varphi(k)^\top w.$$

AR models

- **Idea:** Use observed data $\varphi(1), \ldots, \varphi(N)$ to estimate a value $\hat{w}$ of the parameter a which minimizes the prediction errors in LS sense.
- That is, we solve

$$\min_{w} \sum_{k=1}^{N} (y(k) - \varphi(k)^{\top} w)^2$$

- This is an OLS problem

$$\min_{a} \|y - \Phi w\|_2^2,$$

with

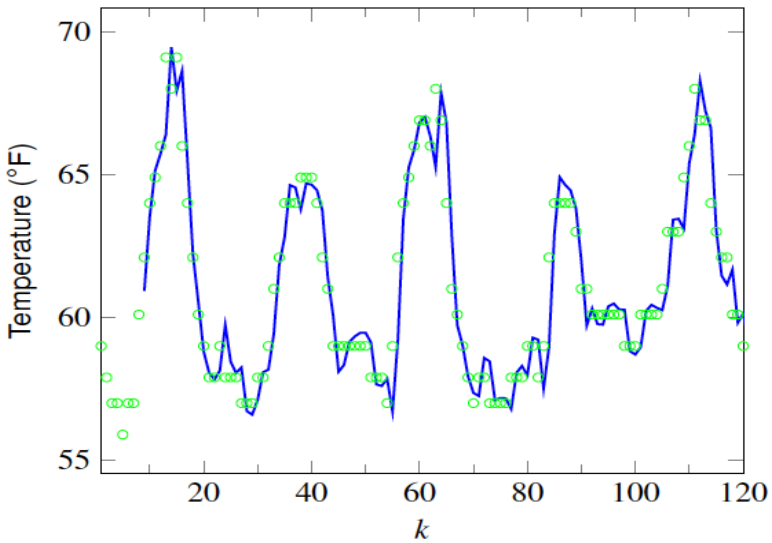
$$y = [y(1) \ \cdots y(N)]^{\top}, \quad \Phi = \begin{bmatrix} \varphi(1)^{\top} \\ \vdots \\ \varphi(N)^{\top} \end{bmatrix}.$$

- Ridge regression is obtained by adding a ℓ_2 regularization parameter:

$$\min_{w} \ \|y - \Phi w\|_2^2 + \lambda \|w\|_2^2.$$

Example

- hourly temperature at LAX in May 2016, length 744;
- average is 61.76^0F, standard deviation 3.05^oF;
- predictor $\hat{y}_{t+1} = y_t$ gives RMS error 1.16^oF;
- AR model with $M = 8$ gives RMS error 0.98^oF.



Solid line shows one-hour ahead predictions from AR model, first 5 days.

Kernel least-squares

Motivation: Nonlinear auto-regressive regression

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

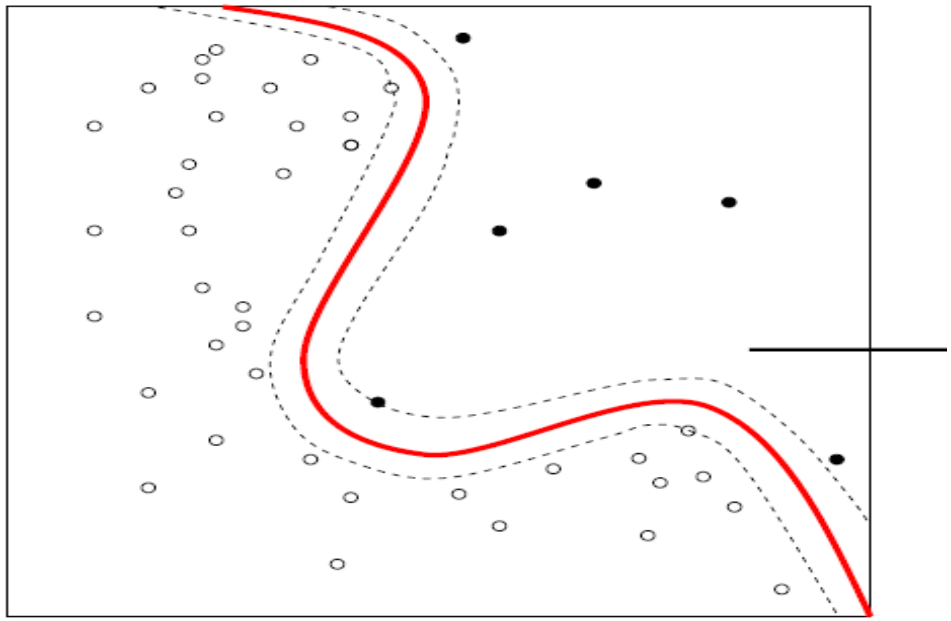
$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2} + w_4 y_{t-1}^2 + w_5 y_{t-1} y_{t-2} + w_6 y_{t-2}^2.$$

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

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

Prediction rule is $\hat{y}_{T+1} = w^\top \phi(x_{T+1})$.

Nonlinear classification



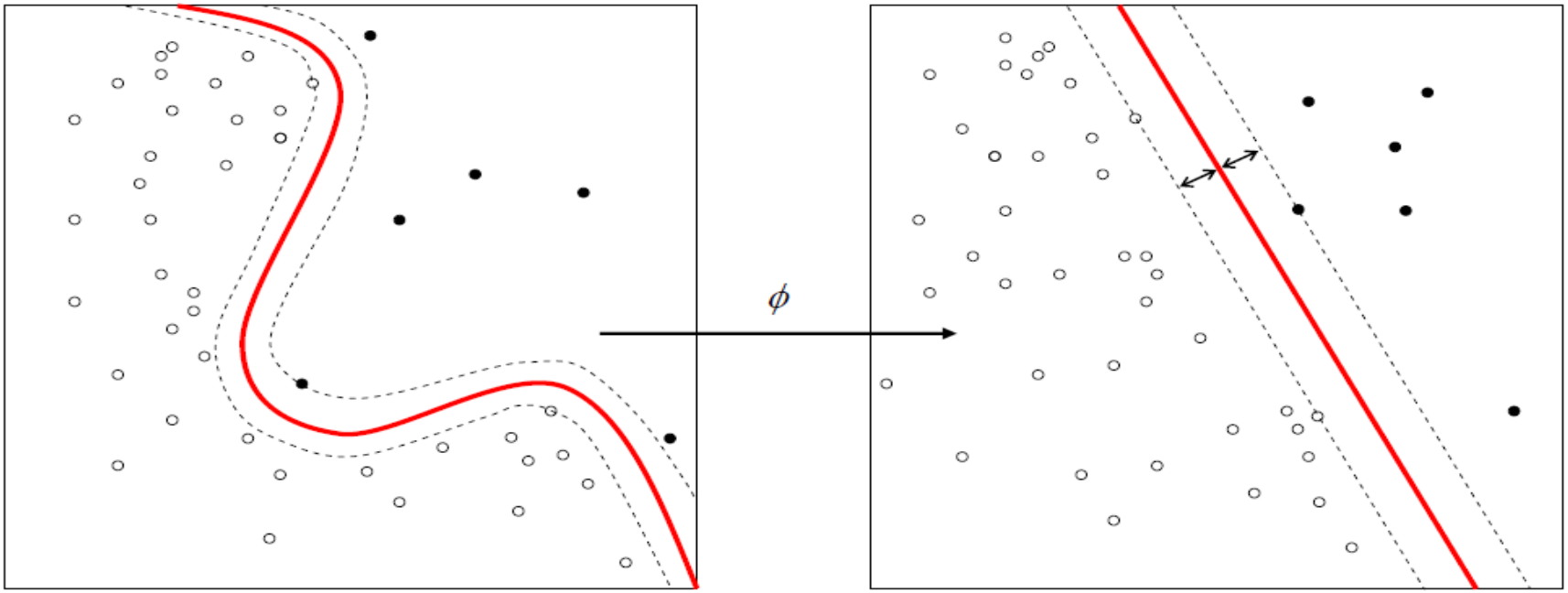
Non-linear (*e.g.*, quadratic) decision boundary

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

Writes $w^\top \phi(x) + b = 0$, 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?

Regularized learning problem

Regularized LS:

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

where

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

Prediction/classification rule: depends only on $w^\top x$, where $x \in \mathbb{R}^p$ is a new data point.

Key result

For the generic problem:

$$\min_{w} \; L(X^\top w, y) + \lambda \|w\|_2^2$$

where L is *any* loss function, the optimal w lies in the span of the data points $(x_1, \ldots, x_n)$:

$$w = Xv$$

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

Proof

Fundamental theorem of linear algebra

For any matrix $X \in \mathbb{R}^{p \times n}$: every $w \in \mathbb{R}^p$ can be written as the sum of two *orthogonal* vectors, one in the range of X and the other orthogonal to it:

$$w = Xv + r$$

where $v \in \mathbb{R}^n$, and $X^\top r = 0$ (that is, r is in the nullspace $\mathcal{N}(X^\top)$).

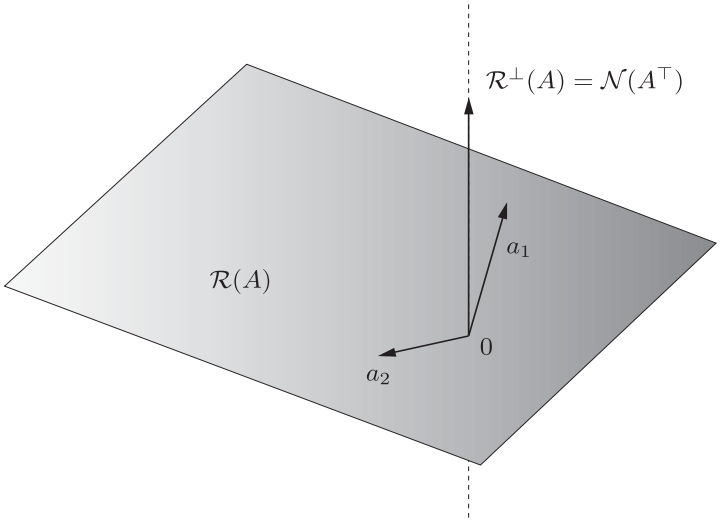


Figure shows the case $X = A = (a_1, a_2)$.

Consequence of key result

For the generic problem: (here L is any "loss" function)

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

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

Hence training problem depends only on the $n \times n$ (PSD) matrix $K := X^\top X$:

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

Kernel matrix

The training problem depends only on the "kernel matrix" $K = X^\top X$

$$K_{ij} = x_i^\top x_j, \;\; 1 \leq i, j \leq n.$$

That is, 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 training data points $x_1, \ldots, x_n$:

$$w^\top x = v^\top X^\top x = v^\top k, \;\; k := X^\top x = (x^\top x_1, \ldots, x^\top x_n).$$

Computational advantage: Once K is formed (this takes $O(n^2 p)$), then the training problem has only n variables. When $p >> n$, 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 ϕ a non-linear mapping.

Example: in classification with quadratic decision boundary, we use

$$\phi(x) := (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2).$$

This leads to the modified kernel matrix

$$K_{ij} = \phi(x_i)^\top \phi(x_j), \quad 1 \leq i, j \leq n.$$

The kernel function

The *kernel function* associated with mapping ϕ is

$$k(x, z) = \phi(x)^\top \phi(z).$$

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 ϕ.

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

$$\phi(x)^\top \phi(z) = (1 + x^\top z)^2.$$

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 $x, z \in \mathbb{R}^n$,

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

Computational effort grows linearly in n.

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

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

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 ϕ to infinite-dimensional feature space.

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