Sparse Approximation and Variable Selection

Lorenzo Rosasco

9.520 Class 07

February 26, 2007

About this class

Goal To introduce the problem of variable selection, discuss its connection to sparse approximation and describe some of the methods designed to solve such problems

Why Selecting Variables?

- **interpretabilty of the model**: in many learning problems a main goal, besides good prediction , is to gain a better understanding of the problems, for example detecting the most discriminative information
- **data driven representation**: in place of tailoring an ad hoc representation (for example via a kernel) we can take a large, redundant set of measurements and then try to devise a data driven selection scheme
- **compression** it is often desirable to have parsimonious models, that is models requiring a (possibly very) small number of parameters to be described
- ...

A Useful Example

Biomarker Identification

Set up:

- n patients belonging to 2 groups (say two different diseases)
- p measurements for *each* patient quantifying the expression of p genes

Goal:

- learn a classification rule to predict occurrence of the disease for future patients
- detect which are the genes responsible for the disease

$p \gg n$ paradigm

typically n is in the order of tens and p of thousands....

Some Notation

Measurement matrix

Let X be the $n \times p$ measurements matrix.

$$X = \begin{pmatrix} x_1^1 & \dots & \dots & \dots & x_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^1 & \dots & \dots & \dots & x_n^p \end{pmatrix}$$

- n is the number of examples
- p is the number of variables
- we denote with X^i, $i = 1, \dots, p$ the columns of X

For each patient we have a response (output) $y \in R$ or $y = \pm 1$.
In particular we are given the responses for the training set

$$Y = (y_1, y_2, \dots, y_n)$$

Approaches to Variable Selection

So far we still have to define what are "*relevant*" variables. Different approaches are based on different way to specify what is relevant.

- Filters methods.
- Wrappers.
- Embedded methods.

We will focus on the latter class of methods.

(see "Introduction to variable and features selection" Guyon and Elisseeff '03)

Few Words on Filter Methods

The idea is to rank the variables X^i, $i = 1, \ldots, p$ according to some criteria $r(i)$

- **correlation score** (similar criteria for non linear correlation):

$$r(i) = \frac{cov(X^i, Y)}{\sqrt{Var(X^i)Var(Y)}}$$

- **single variable classifiers**, $r(i)$ is the test error of a classifier trained with only the i-th variable
- **information based criteria**

$$r(i) = \int_{X^i} \int_{Y} P(X^i, Y) \log \frac{P(X^i, Y)}{P(X^i)P(Y)}$$

Filter or Non Filter?

Cons

Filter methods usually look separately at each variable and are prone to provide us with redundant set of variables.
Moreover no explicit selection is embedded in the method and some thresholding procedure is needed.

Pros

Nonetheless they are often pretty fast and provide indications if some variables are by far more important than some others.

Wrapper methods

An algorithm of choice is used (as a *black box*) to evaluate the importance of a set of variables for the given task.

$\Rightarrow$ a strategy is needed to explore the variable space.

A brute force approach is infeasible even for relatively few features.

two main approaches

- **backward elimination**: start with all the variables and proceed iteratively to discard them...
- **forward selection**: start with no variables and incrementally add them...

...until some stopping criteria is satisfied. For example a fixed number of variables, or a certain validation error, is reached.

About Wrappers

- Such methods often require massive computation unless a favorable search strategy can be used.
- Using a learning algorithm as a black box make them often simple and problems independent...
- ...still we risk to forget what the learning machine does to avoid overfitting.

Embedded Methods

The selection procedure is **embedded** in the training phase.

An intuition

what happens to the generalization properties of empirical risk minimization as we subtract variables?

- if we keep all the variables we probably overfit
- if we take just a few variables we are likely to oversmooth (in the limit we have a single variable classifier!)

We are going to discuss this class of methods in detail.

Function Approximation

Suppose the output is a linear combination of the variables

$$f(x) = \sum_{i=1}^{p} \beta_i x_i = \langle \beta, x \rangle$$

each coefficient β_i can be seen as a weight on the i-th variable. For a given training set we have to solve (approximatively)

$$f(x_i) = y_i \quad i = 1, \ldots, n$$

recall that that data are sampled and noisy.

Solving a BIG linear system

In vector notation we can can write the problem as a linear system of equation

$$Y = X\beta$$

Though existence is ensured (recall $p \gg n$) solution is *not* unique. Moreover we can expect unstable behavior w.r.t. noisy data.

In other words the problem is *ill-posed*

Sparse Approximation

We can consider a more general framework.
Given a function (a signal) $f : X \to \mathbb{R}$ we look for a *sparse* approximation on an overcomplete dictionary.

Overcomplete Dictionary

Let

$$\mathcal{D} = \{\phi_i : X \to \mathbb{R}; i = 1, \dots\} \subset \mathcal{H}$$

be a set of functions, namely the *dictionary*. The functions ϕ_i are called *atoms*.
Overcomplete means that we have many ways to write

$$f = \sum_i \beta_i \phi_i$$

where $f \in \mathcal{H}$.

Examples of Overcomplete Dictionary

- In the last years a large number of overcomplete dictionary have been proposed: stationary wavelets, wavelet packets, cosine packets, chirplets, and warplets ...
- The simplest example we can think of, is a union of orthonormal basis $(e_i), (g_i), ...$ in some space $\mathcal{H}$.
  It is clear that every function in $\mathcal{H}$ can be written in many ways using the elements of $\mathcal{D}$.

Finite dimensional problem

Typically one has to solve a (noisy) discretization of the sparse approximation problem: the signal f is replaced by a vector $\mathbf{f}$.

Discretization via Sampling

- **sampling**: a possible discretization is obtained considering a sample $x_1, \ldots, x_n$ and

$$\mathbf{f} = (f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n$$

- **noise**: the signal is often noisy so that f one has a noisy vector

$$Y = (y_1, \ldots, y_n) \in \mathbb{R}^n$$

where

$$y_j = f(x_j) + noise; \quad j = 1, \ldots, n$$

Truncated Dictionary

In such cases it often suffices to consider a truncated dictionary $\phi_1 \dots, \phi_p$ and write

$$f(x_j) = \sum_{i=1}^{p} \beta_i \phi_i(x_j) \quad j = 1, \dots, n$$

The measurements matrix is now

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \dots & \dots & \dots & \phi_p(x_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1(x_n) & \dots & \dots & \dots & \phi_p(x_n) \end{pmatrix}$$

Few Comments

- It is easy to check that if the dictionary is simply an orthonormal basis $\phi_i = e_i$, then

$$e_i(x) = x_i$$

  and the measurement matrix Φ is simply the matrix X we previously defined;
- in principle if we sample $(x_1, \ldots, x_n)$ and n can increase the size of the truncated dictionary should depend on n, that is $p = p(n)$;
- if $\mathbf{f}$ is a discrete and finite signal (e.g. an image) the idea is simply to take $p \gg n$.

In any case one should be aware of the possible effects of the discretization step.

Ill-Posed Problem

Ill-posed problem

Using again vector notation, we have to solve (approximatively) the ill-posed problem

$$Y = \Phi\beta$$

Even in the noiseless case there is not a unique solution and in general for noisy data we need some regularization procedure.

$\Rightarrow$ In particular we would like to have **sparsity enhancing regularization**.

Generalized and Tikhonov Regularized Solutions

To restore well posedeness:

- if **there is no** noise then take the generalized solution

$$\min_{\beta \in \mathbb{R}^p} \left(\sum_{i=1}^{p} \beta_i^2 \right)^{1/2} \quad \text{s.t} \;\; \mathbf{f} = \Phi\beta$$

  (called method of frames (MOF) in signal processing)

- if **there is** noise use Tikhonov regularization

$$\min_{\beta \in \mathbb{R}^p} \{ \frac{1}{n} \sum_{j=1}^{n} V(y_j, \langle \beta, \Phi(x_j) \rangle) + \lambda \sum_{i=1}^{p} \beta_i^2 \}$$

$\Rightarrow$ in general all the β_i will be different from zero.

Sparsity

Define the "zero"-norm (not a real norm) as

$$\|\beta\|_0 = \#\{i = 1, \ldots, p \mid \beta_i \neq 0\}$$

It is a measure of how "complex" is f and of how many variables are important.

Is it a good way to define sparsity?

L0 Regularization

If we have the *prior* information that only a few variables are meaningful we can look for

$$\begin{aligned} \min_{\beta \in \mathbb{R}^p} \quad & \|\beta\|_0 \\ s.t. \quad & \mathbf{f} = \Phi\beta \end{aligned}$$

or, since the data are noisy, we would like to consider

$$\min_{\beta \in \mathbb{R}^P} \{ \frac{1}{n} \sum_{j=1}^{n} V(y_j, \langle \beta, \Phi(x_j) \rangle) + \lambda \|\beta\|_0 \}$$

$\Rightarrow$ This is as difficult as trying all possible subsets of variables.

Can we find meaningful approximations?

Approximate solutions

Two main approaches

There exist approximations for various choices of loss function. Most of them fall in either one of the following two approaches

1. Convex relaxation (L1 regularization, L0-SVM...)
2. Greedy schemes (boosting algorithms, projection pursuit...)

We mostly discuss the first class of methods.

One Slide on Greedy Approaches

Very similar techniques have been proposed by different communities with different names:

- statistics - forward stagewise regression,
- approximation theory - greedy algorithms,
- learning - boosting methods,
- signal processing - projection pursuit methods.

The various algorithms are often based on the iteration of the following steps. After some initialization:

1. selection an element of the dictionary,
2. update of the solution.

These schemes proceed incrementally and are not based on a global optimization procedure.

Convex Relaxation

A natural approximation to L0 regularization is given by:

$$\frac{1}{n}\sum_{j=1}^{n} V(y_j, \langle \beta, \Phi(x_j) \rangle) + \lambda \|\beta\|_1$$

where $\|\beta\|_1 = \sum_{i=1}^{p} |\beta_i|$.

If we choose the square loss

$$\frac{1}{n}\sum_{j=1}^{n} (y_j - \langle \beta, \Phi(x_j) \rangle)^2 = \|Y - \Phi\beta\|_n^2$$

Such a scheme is related to **Basis Pursuit** and the **Lasso** algorithms.

Lq regularization?

One might wonder why L1 penalty should ensure sparseness and what happens using more general penalty of the form

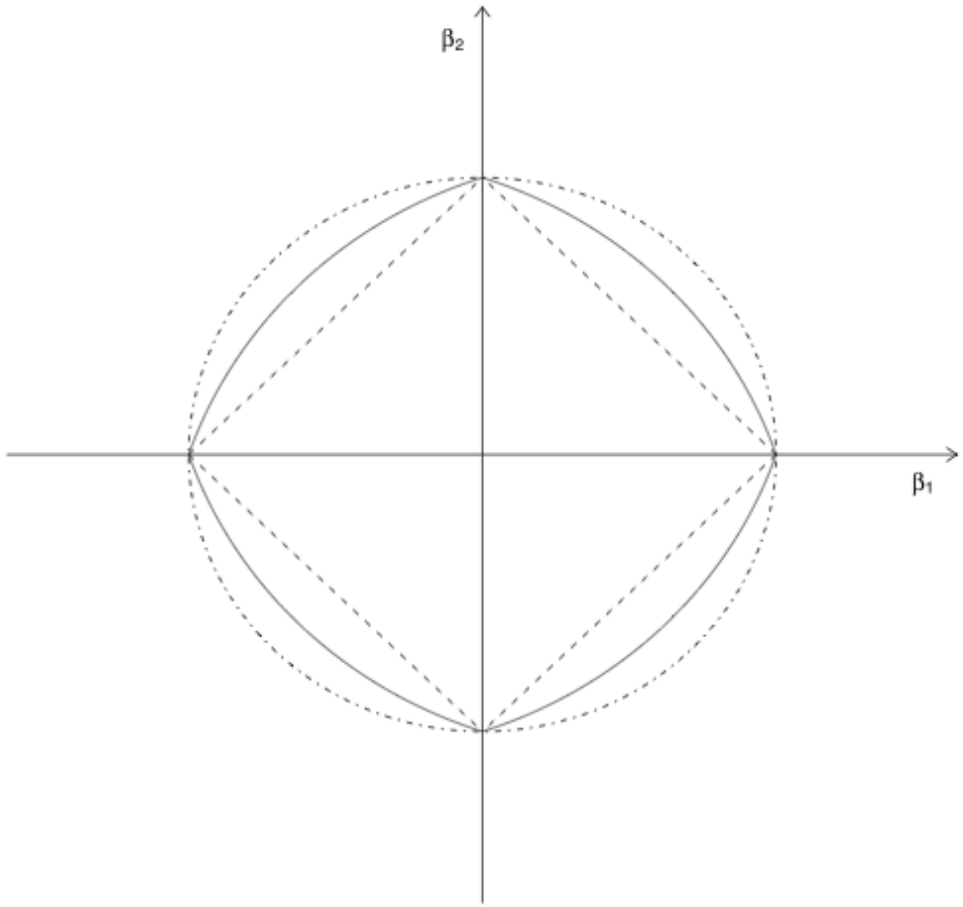
$$\|\beta\|_q = (\sum_{i=1}^{p} |\beta|^q)^{1/q}$$

(related to bridge regression in statistics).
It can be proved that:

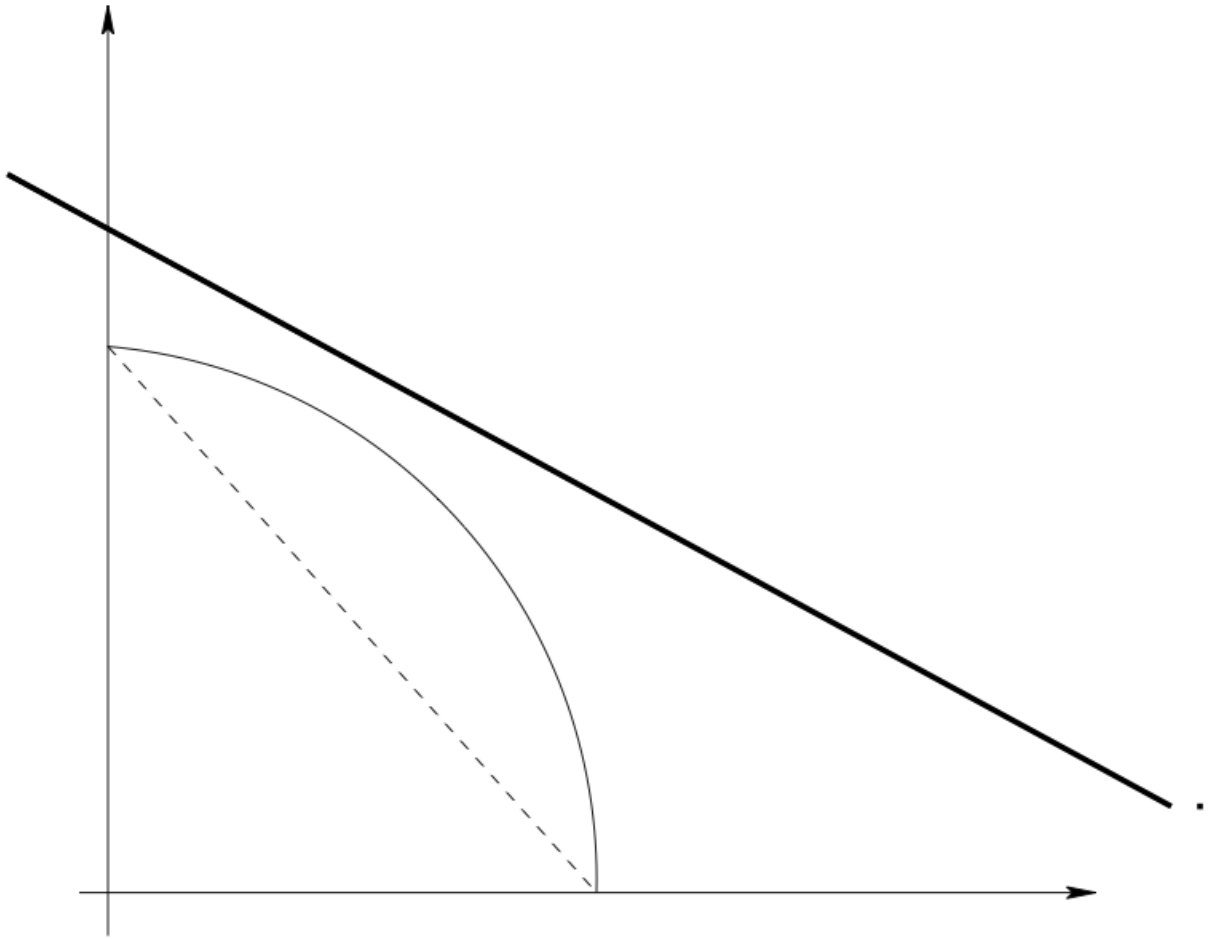
- $\lim_{q\to 0} \|\beta\|_q \to \|\beta\|_0$,
- for $0 < q < 1$ the norm is **not** a convex map,
- for $q = 1$ the norm **is** a convex map and is **strictly** convex for $q > 1$.

How about sparsity?

Geometrical View



Graphical Solution in in 2D



Back to L1 regularization

We focus on the square loss so that we now have to solve

$$\min_{\beta \in \mathbb{R}^p} \|Y - \beta\Phi\|^2 + \lambda \|\beta\|_1 .$$

- Though the problem is no longer hopeless there is no straightforward way to solve it.
- The functional is convex but not *strictly* convex, so that the solution is not unique.
- One possible approach relies on linear or quadratic programming techniques.
- Using convex analysis tools we can get a constructive representation theorem.

An Iterative Thresholding Algorithm

If we let β^λ be a solution of the convex relaxation problem, it can be proved that the following iterated algorithm converges to β^λ as the number of iteration increases.

Set $\beta_0^\lambda = 0$ and let

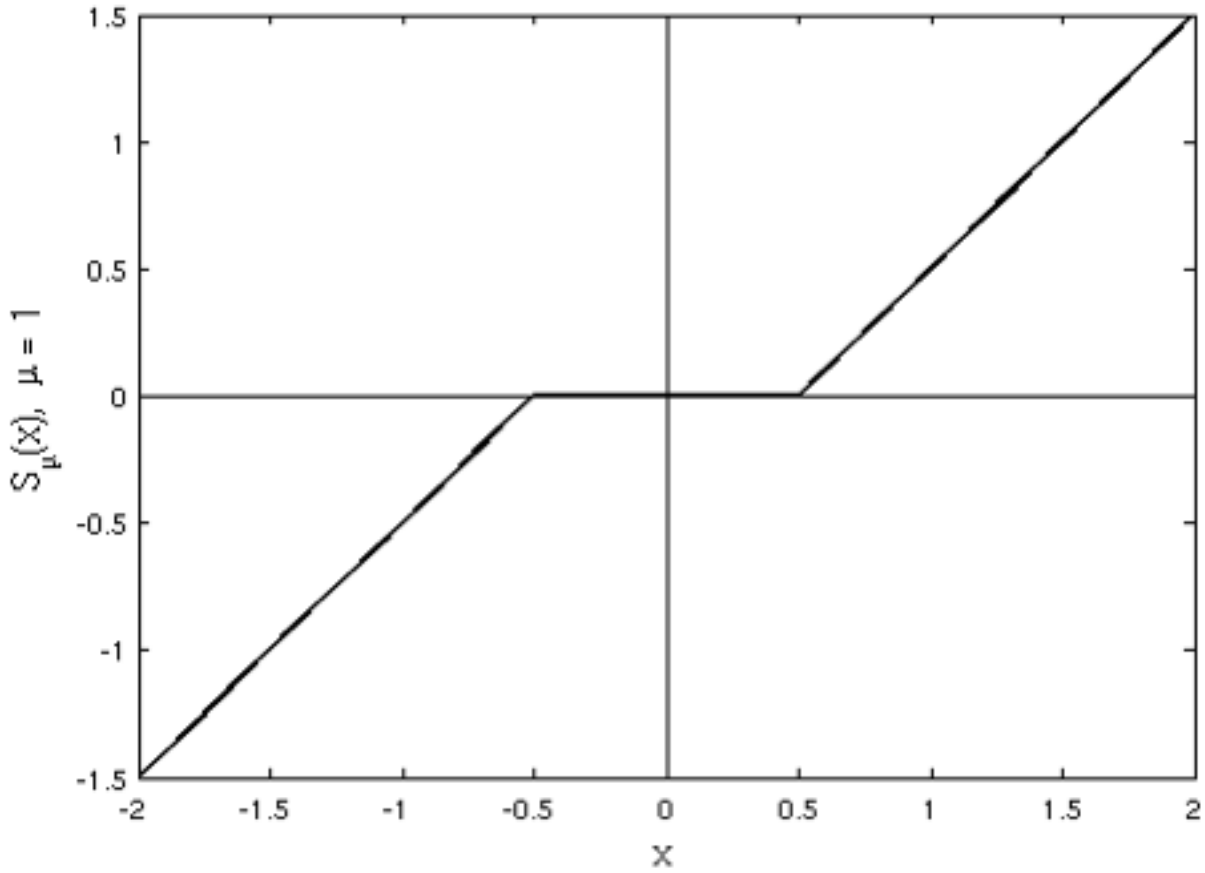
$$\beta_t^\lambda = S_\lambda[\beta_{t-1}^\lambda + \tau \Phi^T (Y - \Phi \beta_{t-1}^\lambda)]$$

where τ is a normalization constant ensuring $\|\Phi\| \leq 1$ and the map S_λ is defined component-wise as

$$S_\lambda(\beta_i) = \begin{cases} \beta_i + \lambda & \text{if } \beta_i < -\lambda/2 \\ 0 & \text{if } |\beta_i| \leq \lambda/2 \\ \beta_i - \lambda & \text{if } \beta_i < \lambda/2 \end{cases}$$

(see Daubechies et al.'05)

Thresholding



Algorithmics Aspects

Set $\beta_0^\lambda = 0$
for t=1:tmax

$$\beta_t^\lambda = S_\lambda[\beta_{t-1}^\lambda + \tau\Phi^T(Y - \Phi\beta_{t-1}^\lambda)]$$

- The algorithm we just described is very easy to implement but can be quite heavy from a computational point of view.
- The number of iteration t can be stopped when a certain precision is reached.
- The complexity of the algorithm is $O(tp^2)$ for each value of the regularization parameter.
- The regularization parameter controls the degree of sparsity of the solution.

Some Final Remarks

- **Computational Burden**: algorithms for feature selection are often computationally expansive. This should be no surprise since the problem we are trying to solve is more difficult then a simple regression/classification task.
- **About Uniqueness**: the solution of in L1 regularization is not unique. Note however that the various solution have the **same prediction properties** but **different selection properties**.
- **Correlated Variables**: If we have a group of correlated variables the algorithm is going to select just one of them. This can be bad for interpretability but maybe good for compression.
- **Connections**: the same approaches is used MANY different problem: compressed sensing, inverse problems, linear coding...