Chapter 3
Convex Optimization in Machine Learning

Mathematical Modeling

Le Hong Trang
Faculty of Computer Science and Engineering
HCMC University of Technology

Contents

Outline

1. **Least Squares Model**
   - Regression analysis and data fitting
   - Trend analysis
2. **Support Vector Machines**
   - Hard margin
   - Soft margin
   - Kernels
3. **Image Restoration**
4. **How about Deep Networks? – Nonlinear Optimization!**

Least Squares

$$\min \|Ax - b\|_2^2 = \sum_{i=1}^{k} (a_i^T x - b_i)^2,$$

where $A \in \mathbb{R}^{k \times n} (k \geq n), b \in \mathbb{R}^k$, a_i^T are rows of A, and $x \in \mathbb{R}^n$.

Solving least-squares problems

- Analytical solution: $x^* = (A^T A)^{-1} A^T b$.
- Reliable and efficient algorithms and software.
- Computation time proportional to $n^2 k$.
- A mature technology.

Using least-squares

- Basis for regression analysis, optimal control, and many parameter estimation and data fitting methods.
- Easy to recognize.
- A few standard techniques increase flexibility
  - *Including weights*

$$\min \sum_{i=1}^{k} w_i (a_i^T x - b_i)^2.$$

  - *Adding regularization terms*

$$\min \sum_{i=1}^{k} (a_i^T x - b_i)^2 + \rho \sum_{j=1}^{n} x_j^2.$$

Data fitting

General statement

In a fitting problem, we are given data

$$(u_1, y_1), (u_2, y_2), \dots, (u_m, y_m)$$

with $u_i \in D$ and $y_i \in \mathbb{R}$, and seek a function $f \in \mathcal{F}$ that matches this data as closely as possible. For example in least-squares fitting we consider the problem

$$\min \sum_{j=1}^{m} (f(u_i) - y_i)^2,$$

which is a simple least-squares problem in the variable x.

We can add a variety of constraints

- inequalities that must be satisfied by f at various points,
- constraints on the derivatives of f,
- monotonicity constraints,
- or moment constraints.

Data fitting

Polynomial fitting

Given data $u_1, u_1, \ldots, u_m \in \mathbb{R}$ and $v_1, v_2, \ldots, v_m \in \mathbb{R}$, we need to approximately fit a polynomial of the form

$$p(u) = x_1 + x_2 u + x_3 u^2 \ldots + x_n u^{n-1}$$

to the data.

For each x we form the vector of errors,

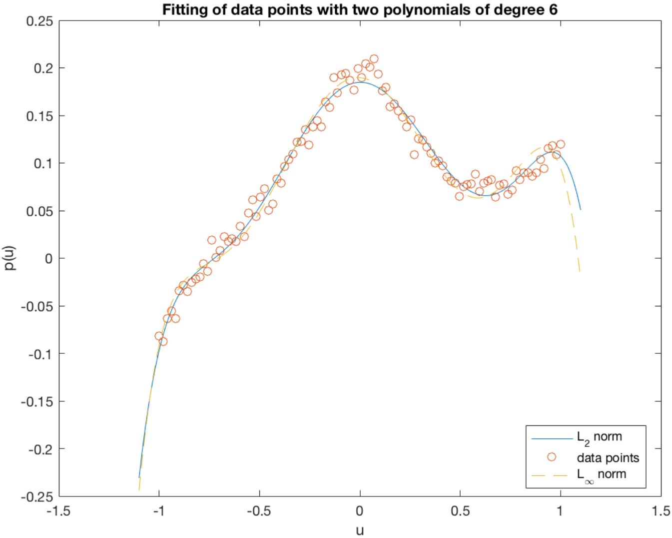
$$e = (p(u_1) - v_1, p(u_2) - v_2, \ldots, p(u_m) - v_m) .$$

To find the polynomial that minimizes the norm of the error, we solve the norm approximation problem

$$\min \|e\| = \|Ax - v\|,$$

where $x \in \mathbb{R}^n$, $A_{ij} = u_i^{j-1}$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$

Polynomial fitting



Trend filter: time series analysis

- The problem of estimating underlying trends in time series data arises in a variety of disciplines.
- The l_1 trend filtering method produces trend estimates x that are piecewise linear from the time series y.

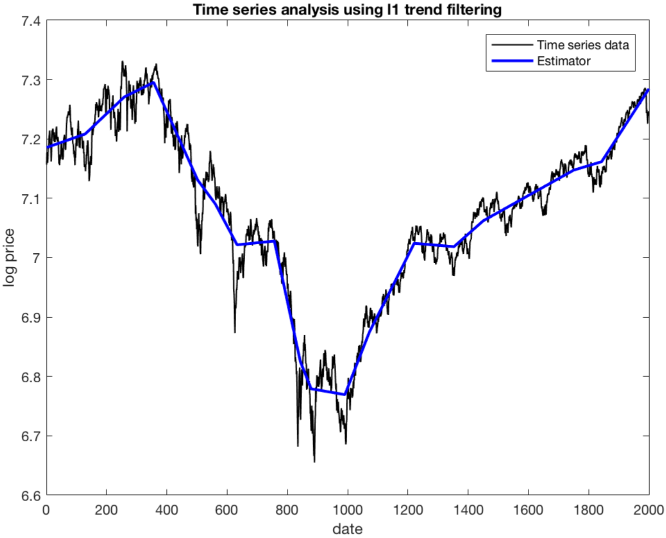
Optimization form

Given a time series data y, estimate x by solving

$$\min \frac{1}{2}\|y - x\|^2 + \lambda\|Dx\|_1,$$

where D is the second difference matrix, with rows $[0 \ldots -1\ 2\ \ -1 \ldots 0]$, $\lambda > 0$ is the regularization parameter.

Trend filter: time series analysis



Outline

Classification problem

A statement

A classification problem has two types of variables,

- $x_i \in \mathbb{R}^d$: the vector of observations (features) in the world,
- $y_i \in \mathbb{R}^d$: state (class) vector of the world.

The set of all samples in this world (or dataset) constructs the observation set $X \in \mathcal{R}^{n\times d}$ and state set $Y \in \mathbb{R}^{n\times d}$. The task of machine learning algorithm for classification is to find a mapping $X \rightarrow Y$ that achieves smallest number of misclassification.

Linear discriminant learning

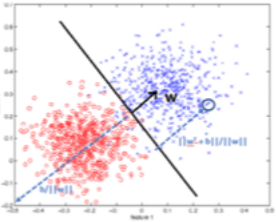
- *Propose a parametric family of decision boundaries, then pick the element in this family that produces the best classifier*.
- Take our decision boundary between two classes as a hyperplane, such that $w^T x + b = 0$,
  - w being the normal to the plane and b being the *bias term*.
- The decision function can be expressed as

$$f^*(x) = \begin{cases} 0, & \text{if } w^T x + b > 0, \\ 1, & \text{if } w^T x + b < 0. \end{cases}$$

The decision boundary

- Normal vector: w; distance to origin: $d = b/\|w\|$.
- distance to a data point x_i: $|w^T x_i + b|/\|w\|$.

Linear discriminant learning



The decision function

- Use labels $y \in \{-1, 1\}$ instead of $y \in \{0, 1\}$

$$f^*(x) = \begin{cases} -1, & \text{if } w^T x + b > 0, \\ 1, & \text{if } w^T x + b < 0, \end{cases}$$

then $f^*(x) = \text{sign}(w^T x + b)$.

A simple necessary and sufficient condition for a given training set $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, is *linearly separable*, i.e.,

$$y_i(w^T + b) > 0, \text{ for } i = 1, 2, \ldots, n. \qquad (1)$$

Kernel-based learning

Question arising when dealing with nonlinear problems

- Trying to fit a nonlinear model? – not easy!
- Idea: keep using the linear model
  - Map the problem a new (higher-dimensional) space (*called the feature space*) by doing a nonlinear transformation using suitably chosen basis functions;
  - Apply a linear model in *the feature space*.
  - Learning a linear boundary in a higher dimensional space will be equivalent to learning a nonlinear boundary in the current space.

$\rightarrow$ Known as the **kernel trick**.

Kernel-based learning

- If we introduce a mapping $\phi : X \to Z$ such that $dim(Z) > dim(X)$, we can achieve linear separability for some $k = dim(Z)$.
- Instead of applying this mapping to each data point as $\phi(x)$, we can leverage the dot-product form

$$y_i(\sum_{j=1}^{n} \alpha_j y_j x_j^T x_i + b) \tag{2}$$

$$y_i(\sum_{j=1}^{n} \alpha_j y_j \phi(x_j)^T \phi(x_i) + b). \tag{3}$$

Then we apply the kernel trick, $K(x, z) = \phi(x)^T \phi(z)$.

- Some formulas for $K(x, z)$
  1. $K(x, z) = x^T z$, linear kernel.
  2. $K(x, z) = e^{\frac{\|x-z\|^2}{\delta}}$, Gaussian kernel.
  3. $K(x, z) = (1 + x^T z)^k$, polynomial kernel.

Support Vector Machines (SVM)

- SVMs go one step further from the necessary and sufficient condition given by (1) and works on the *margin* defined by the distance from the boundary to the closest point,

$$\gamma = \min_i \frac{|w^T x_i + b|}{\|w\|}.$$

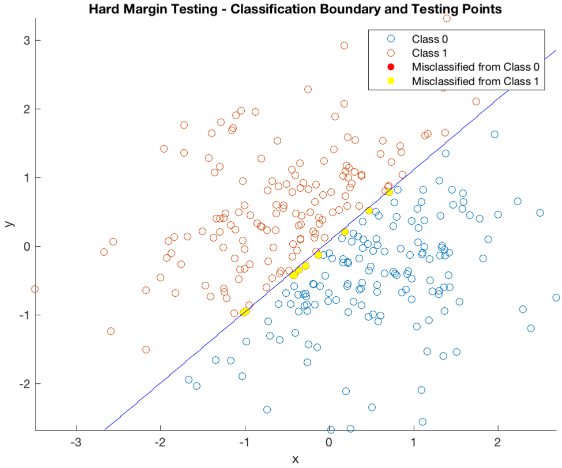
- Maximizing γ is ill-defined, since γ does not change if both w and b are scaled by a factor λ. We then need some sort of normalization,
  - this can be done by arbitrarily selecting some normalization on w, e.g. $\|w\| = 1$
- The SVM algorithm works with a more convenient normalization and makes $|w^T x + b| = 1$ for the closest point, i.e. $\min_i |w^T x_i + b|$ under $\gamma = \frac{1}{\|w\|}$.

SVM model

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \forall i. \end{aligned}$$

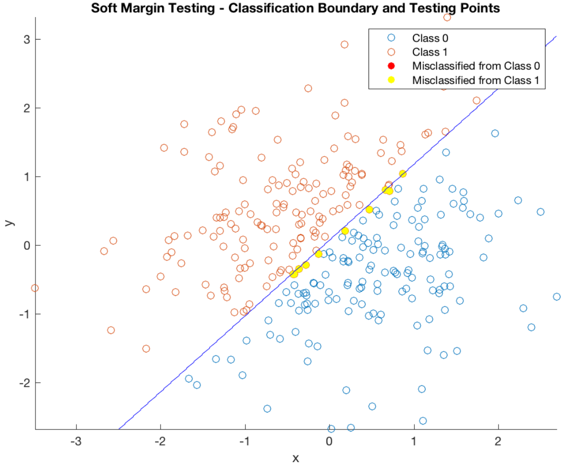
SVM – hard margin

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2$$
$$\textbf{s.t.} \quad y_i(w^T x_i + b) \geq 1, \forall i.$$



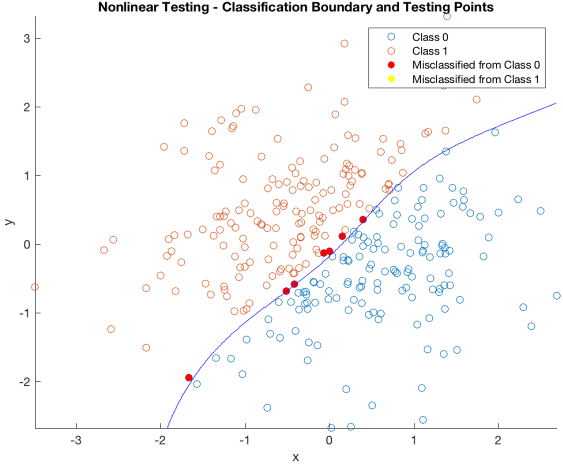
SVM – soft margin

$$
\begin{aligned}
\min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 \\
\textbf{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i, \\
& \xi_i \geq 0, \forall i.
\end{aligned}
$$



SVM – nonlinear classification with kernels

Using Gaussian kernel: $K(x,z) = e^{\frac{\|x-z\|^2}{\delta}}$.



Outline

1. Least Squares Model
   - Regression analysis and data fitting
   - Trend analysis
2. Support Vector Machines
   - Hard margin
   - Soft margin
   - Kernels
3. **Image Restoration**
4. How about Deep Networks? – Nonlinear Optimization!

Image restoration/reconstruction

- original

- optical blur

- motion blur

- spatial quantization (discrete pixels)

- additive intensity noise

Image restoration

- A general noisy (including noise, blur, inpainting) image $\mathbf{y}$, of an original image $\mathbf{x}$, can be modeled

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{n},$$

  where
  - $\mathbf{B}$: a direct operator represented under matrix form,
  - $\mathbf{n}$: noise.

- As common, images are adopted by vector notation
  - The pixels on an $M \times N$ image are stored as an (NM)-vector.
  - If the number of elements of $\mathbf{x}$ is n then $\mathbf{x} \in \mathbb{R}^n$, while $\mathbf{y} \in \mathbb{R}^m$ (m and n can be different).

Remark

The problem of estimating $\mathbf{x}$ from $\mathbf{y}$ is *ill-posed*. This inverse problem can then be solved by adopting a some sort of regularization.

Image restoration – models

Unconstrained formulation

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{\mathbf{1}}{\mathbf{2}} \|\mathbf{Bx} - \mathbf{y}\|_{\mathbf{2}}^{\mathbf{2}} + \tau\phi(\mathbf{x}),$$

where

- $\phi : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the regularizer.
- τ is the regularization parameter.

Lagrangian (constrained form)

$$\begin{aligned} \min_{\mathbf{x}} \quad & \phi(\mathbf{x}) \\ \text{s.t.} \quad & \|\mathbf{Bx} - \mathbf{y}\| \leq \epsilon. \end{aligned}$$

Regularization functions

- $\phi(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1} |x_i|$: basis pursuit denoising (BPD).
- $\phi(\mathbf{x}) = \varphi(\mathbf{Dx})$: total-variational regularization.

Image restoration: inpainting and noise by unconstrained model

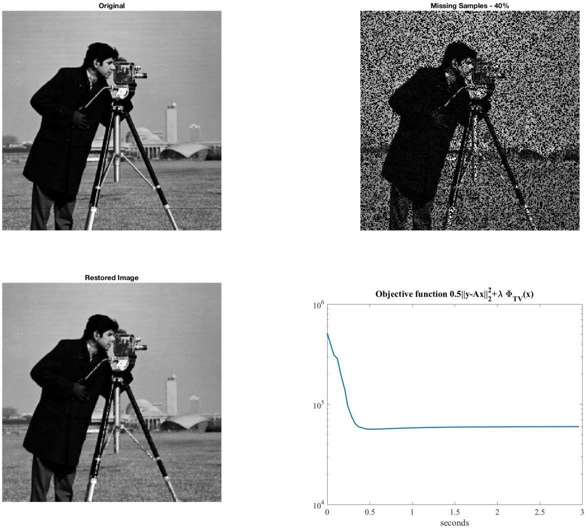


Image restoration: blurred by unconstrained model

Original

Blurred and noisy

Estimated

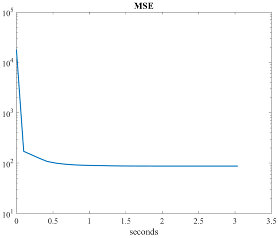


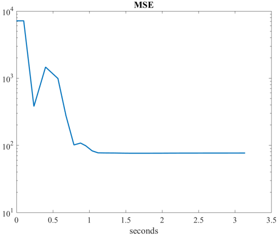
Image restoration: blurred and noise by constrained model

Original

Blurred and noisy

Constrained model result

MSE



Outline

1. **Least Squares Model**
   - Regression analysis and data fitting
   - Trend analysis
2. **Support Vector Machines**
   - Hard margin
   - Soft margin
   - Kernels
3. **Image Restoration**
4. **How about Deep Networks? – Nonlinear Optimization!**