Optimization of Communication Systems
EE 669

Chathuranga Weeraddana

June 2022

Lecture 11: Applications I [1]

[1] The content in this lecture is based on § 13,§ 15, § 16 of [Calafiore-2014].

Learning from Data

Some Notation

- $X \in \mathbb{R}^{n,m}$
- $X = [x_1\ x_2\ \cdots\ x_m]$, where $x_i \in \mathbb{R}^n$
- x_i is the ith data example
- particular dimension j of x_i is related to a feature, where $j \in \{1, 2, \ldots, n\}$

Learning from Data: Supervised Learning

Supervised Learning

- task: build a model $\hat{y}$ of an unknown function $y : \mathbb{R}^n \to \mathbb{R}$

Supervised Learning

- given set of examples and their corresponding outputs
- i.e., we have $(x_i, y_i) \in \mathbb{R}^{n+1}$ for $i = 1, 2, \ldots, m$
- we use these examples to learn a model $y_w : \mathbb{R}^n \to \mathbb{R}$, where $w \in \mathbb{R}^p$ is a model parameter

Supervised Learning

- then for an arbitrary input $x \in \mathbb{R}^n$ we have the prediction $\hat{y} \in \mathbb{R}$, where $\hat{y} = y_w(x)$

Examples: Least-Squares Prediction via Polynomial Models

Linear Models

- linear models

$$y_w(x) = w_1 + xw_2 + e(x) = w^{\mathsf{T}}\phi(x) + e(x) \qquad (1)$$

- the optimization problem to determine $w \in \mathbb{R}^2$ (a least-squares approach):

$$\begin{array}{ll} \text{minimize} & ||\Phi w - y||_2^2 \\ \text{subject to} & w \in \mathbb{R}^2 \end{array}$$

- an unconstrained optimization problem

- what about polynomial models? How does (1) change?

Regularized Least-Squares

- regularized least-squares

$$\begin{array}{ll} \text{minimize} & ||\Phi w - y||_2^2 + \lambda ||w||_2^2 \\ \text{subject to} & w \in \mathbb{R}^2 \end{array}$$

- selection of model order k and regularization parameter λ

LASSO

- LASSO [2]

$$\begin{array}{ll} \text{minimize} & ||\Phi w - y||_2^2 + \lambda ||w||_1 \\ \text{subject to} & w \in \mathbb{R}^2 \end{array}$$

- sparsity

[2] See lecture slides set 1, pages 6 to 14.

Examples: Binary Classification

Binary Classification

- recall a previous case: predict a real-valued variable y
- an affine function $x \to w^{\mathsf{T}}\phi(x) = w^{\mathsf{T}}(1, x)$ was used
- there are cases where y is binary, e.g., $y_i \in \{-1, 1\}$, $i = 1, 2, \cdots, m$

Binary Classification

- when predicting a binary label y_i, a simple modification is:

$$\hat{y} = \mathrm{sgn}(x^{\mathsf{T}} w + b)$$

- try to find w such that cost [3] of errors on the training set is minimized

- to this end, define the margins μ_i associated with the training example i:

$$\mu_i = y_i(x_i^{\mathsf{T}} w + b)$$

[3] A cost (or a loss) associated with a training example is incurred when there is a classification error.

Binary Classification

- the cost of errors on the training set is written in terms of margins
- the cost (or loss) for the ith training example is

$$l_i(\mu_i) = l_i\left(y_i(x_i^{\mathsf{T}} w + b)\right) \qquad i = 1, 2, \cdots, m$$

- a classification error in ith example $\iff$ the margin $\mu_i < 0$
- so the function $l_i : \mathbb{R} \to \mathbb{R}$ should be
  - positive and decreasing for negative arguments
  - zero or small for positive arguments

Binary Classification

- minimize the average loss together with a regularization term
- thus, an optimization problem to compute w and b is as follows:

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} l_i\left(y_i(x_i^{\mathsf{T}}w+b)\right) + r(w) \\ \text{subject to} & (w,b) \in \mathbb{R}^{n+1} \end{array}$$

where $r : \mathbb{R}^n \to \mathbb{R}$ is a regularization function

Binary Classification

- commonly used loss functions, $l_i : \mathbb{R} \to \mathbb{R}$
  - $l_i(\mu) = \max(0, 1 - \mu)$: hinge loss
  - $l_i(\mu) = \exp(-\mu)$: exponential loss
  - $l_i(\mu) = \log\left(1 + \exp(-\mu)\right)$: logistic loss
- commonly used regularizers, $r : \mathbb{R}^n \to \mathbb{R}$
  - $r(w) = ||w||_2^2$: ℓ_2 squared
  - $r(w) = ||w||_1$: ℓ_1

Support Vector Machine (SVM)

- l_i for all i: hinge loss
- regularizer: quadratic
- thus, the corresponding optimization problem is:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{m}\sum_{i=1}^{m}\max\Big(0, 1 - y_i(x_i^{\mathsf{T}}w + b)\Big) + \lambda||w||_2^2 \\ \text{subject to} \quad & (w, b) \in \mathbb{R}^{n+1} \end{aligned}$$

Logistic Regression

- l_i for all i: logistic loss loss
- no regularizers
- thus, the corresponding optimization problem is:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{m} \log\left(1+\exp\left(-y_i(x_i^\mathsf{T} w + b)\right)\right) \\ \text{subject to} & (w,b) \in \mathbb{R}^{n+1} \end{array}$$

Learning from Data: Unsupervised Learning

Unsupervised Learning

- in unsupervised learning, the data examples $x_i \in \mathbb{R}^n$ $i = 1, 2, \cdots, m$ is do not come with assigned labels $y_i \in \mathbb{R}$
- i.e., the responses $y_i \in \mathbb{R}$ $i = 1, 2, \cdots, m$, no longer present

Unsupervised Learning

- task: learn some information or structure about the data

Examples: Principal Component Analysis (PCA)

Basic PCA

- let $\tilde{x} = (1/m) \sum_{i-1}^{m} x_i$; the sample average of the data examples
- center the data points, i.e., let $\tilde{X} = [\tilde{x}_1 \ \tilde{x}_2 \ \cdots \ \tilde{x}_m]$ where $\tilde{x}_i = (x_i - \tilde{x})$
- we seek for a normalized direction $z \in \mathbb{R}^n$ in data space
  - such that the sample variance of the 'projections' of the data points on the line determined by z is maximized

Basic PCA

- the projection of $\tilde{x}_i$ on $\mathrm{span}(z)$ is $\alpha_i z$, where $\alpha_i = \tilde{x}_i^\mathsf{T} z$
- sample variance of $\{\alpha_i\}$ is

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}\alpha_i^2 &= \frac{1}{m}\sum_{i=1}^{m} z\,(\tilde{x}_i\tilde{x}_i)^\mathsf{T}\,z \\ &= \frac{1}{m}\;z\left(\sum_{i=1}^{m}\tilde{x}_i\tilde{x}_i^\mathsf{T}\right)z \\ &= \frac{1}{m}\;z\;\tilde{X}\tilde{X}^\mathsf{T}z\end{aligned}$$

Basic PCA

- thus, the optimization problem to compute z is as follows:

$$\begin{aligned} \text{maximize} \quad & z\ \tilde{X}\tilde{X}^{\mathsf{T}} z \\ \text{subject to} \quad & ||z||_2 = 1 \end{aligned}$$

- the solution of the above can be found via SVD

Sparse PCA

- thus, the optimization problem to compute z is as follows:

$$\begin{array}{ll} \text{maximize} & z\,\tilde{X}\tilde{X}^{\mathsf{T}}z \\ \text{subject to} & ||z||_2 = 1 \\ & \mathrm{card}(z) \leq k \end{array}$$

- the dimension (i.e., n) of data is small $\rightarrow$ solution by exhaustively solving basic PCAs

- larger dimensions $\rightarrow$ exhaustive approach breaks $\rightarrow$ other approaches are be adopted

Non-negative Matrix Factorization

- seek a low rank approximation for a given non-negative data matrix $X \in \mathbb{R}^{n \times n}$
- in particular, compute two $n \times k$ non-negative matrices denoted P and Q such that

$$X \approx PQ^{\mathsf{T}}$$

where $k \ll \min(n, m)$

Non-negative Matrix Factorization

- the optimization problem to compute P, Q is as follows:

$$\begin{array}{ll} \text{minimize} & ||\tilde{X} - PQ^{\mathsf{T}}||_F \\ \text{subject to} & P, Q \in \mathbb{R}^{n \times k} \\ & P, Q \geq 0 \end{array}$$

- problem is no convex $\rightarrow$ hard to solve $\rightarrow$ heuristics are adopted

- interpret the idea of $X \approx PQ^{\mathsf{T}}$
  - each document is a particular 'blend' of k basic 'documents'

Robust PCA

- express a data matrix X as a sum of a low-rank + a sparse
- i.e.,

$$X = Y + Z$$

  where Y is low rank and Z is sparse

- the optimization problem to compute $Y, Z \in \mathbb{R}^{n \times m}$ is:

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(Y) + \lambda \operatorname{card}(Z) \\ \text{subject to} & Y, Z \in \mathbb{R}^{n \times m} \\ & Y + Z = X \end{array}$$

Robust PCA

- useful when the measured data is believed to have essentially
  - a low-rank structure
  - there are some ‘spikes’ or outliers
- e.g., a video with light moving objects

Robust PCA

- the problem is not convex
- a commonly used convex approximation is as follows:

$$\begin{array}{ll} \text{minimize} & ||Y||_* + \lambda \, ||Z||_1 \\ \text{subject to} & Y, Z \in \mathbb{R}^{n \times m} \\ & Y + Z = X \end{array}$$

where $|| \cdot ||_*$ is the nuclear norm [4] of Y

[4] See the appendix, pp: 27-29.

Appendix

Nuclear Norm

- let $Y \in \mathbb{R}^{n \times m}$, an arbitrary matrix [5]
- compute SVD of Y, i.e.,

$$Y = \sum_{i=1}^{n} \sigma_i u_i v_i^{\mathsf{T}}$$

[5] Assume $n \leq m$ with out loss of generality.

Nuclear Norm

- $\mathrm{rank}(Y) =$ number of nonzero singular values
- define the vector $s(Y) = (\sigma_1(Y), \sigma_2(Y), \ldots, \sigma_n(Y)) \in \mathbb{R}^n$
- thus, we have that

$$\mathrm{rank}(Y) = \mathrm{card}(s(Y))$$

Nuclear Norm

- $\mathrm{rank}(Y)$ is a nonconvex function, so is $\mathrm{card}(s(Y))$
- as a convex proxy, a commonly used approximation to 'rank' is ℓ_1 norm of $s(Y)$
- since all singular values are positive, we have

$$||s(Y)||_1 = \sum_{i=1}^{n} \sigma_1(Y) \triangleq ||Y||_*$$

- the above norm $||\cdot||_*$ is called the nuclear norm of Y
- thus, $||Y||_* \approx \mathrm{rank}(Y)$ is a commonly used approximation