Shanni Lam
Math189R SP19
Homework 5
Thursday, June 25, 2020

Feel free to work with other students, but make sure you write up the homework and code on your own (no copying homework *or* code; no pair programming). Feel free to ask students or instructors for help debugging code or whatever else, though.

1 (Murphy 12.5 - Deriving the Residual Error for PCA) It may be helpful to reference section 12.2.2 of Murphy.

(a) Prove that

$$\left\| \mathbf{x}_i - \sum_{j=1}^{k} z_{ij}\mathbf{v}_j \right\|^2 = \mathbf{x}_i^\top \mathbf{x}_i - \sum_{j=1}^{k} \mathbf{v}_j^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{v}_j.$$

Hint: first consider the case when $k = 2$. Use the fact that $\mathbf{v}_i^\top \mathbf{v}_j$ is 1 if $i = j$ and 0 otherwise. Recall that $z_{ij} = \mathbf{x}_i^\top \mathbf{v}_j$.

(b) Now show that

$$J_k = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^\top \mathbf{x}_i - \sum_{j=1}^{k} \mathbf{v}_j^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{v}_j\right) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i - \sum_{j=1}^{k} \lambda_j.$$

Hint: recall that $\mathbf{v}_j^\top \mathbf{\Sigma} \mathbf{v}_j = \lambda_j \mathbf{v}_j^\top \mathbf{v}_j = \lambda_j$.

(c) If $k = d$ there is no truncation, so $J_d = 0$. Use this to show that the error from only using $k < d$ terms is given by

$$J_k = \sum_{j=k+1}^{d} \lambda_j.$$

Hint: partition the sum $\sum_{j=1}^{d} \lambda_j$ into $\sum_{j=1}^{k} \lambda_j$ and $\sum_{j=k+1}^{d} \lambda_j$.

a) $||\mathbf{x}_i - \sum_{j=1}^{k} z_{ij}\mathbf{v}_j||_2^2 = (\mathbf{x}_i - \sum_{j=1}^{k} z_{ij}\mathbf{v}_j)^T(\mathbf{x}_i - \sum_{j=1}^{k} z_{ij}\mathbf{v}_j)$
$= \mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k} z_{ij}\mathbf{v}_j^T\mathbf{x}_i - \mathbf{x}_i^T\sum_{j=1}^{k} z_{ij}\mathbf{v}_j + (\sum_{j=1}^{k} z_{ij}\mathbf{v}_j)^T(\sum_{j=1}^{k} z_{ij}\mathbf{v}_j)$ by applying the transpose and foiling
$= \mathbf{x}_i^T\mathbf{x}_i - 2\sum_{j=1}^{k} z_{ij}\mathbf{v}_j^T\mathbf{x}_i + (\sum_{j=1}^{k} z_{ij}\mathbf{v}_j)^T(\sum_{j=1}^{k} z_{ij}\mathbf{v}_j)$ by bringing $\mathbf{x}_i^T$ into the summation
$= \mathbf{x}_i^T\mathbf{x}_i - 2\sum_{j=1}^{k} z_{ij}\mathbf{v}_j^T\mathbf{x}_i + \sum_{j=1}^{K} \mathbf{v}_j^T z_{ij}^T z_{ij}\mathbf{v}_j$ by multiplying the last term
$= \mathbf{x}_i^T\mathbf{x}_i - 2\sum_{j=1}^{k} z_{ij}\mathbf{v}_j^T\mathbf{x}_i + \sum_{j=1}^{k} \mathbf{v}_j^T\mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j\mathbf{v}_j$ by plugging in $z_{ij} = \mathbf{x}_i^T\mathbf{v}_j$
$= \mathbf{x}_i^T\mathbf{x}_i - 2\sum_{j=1}^{k} z_{ij}\mathbf{v}_j^T\mathbf{x}_i + \sum_{j=1}^{k} \mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j\mathbf{v}_j^T\mathbf{v}_j$ by rearranging terms
$= \mathbf{x}_i^T\mathbf{x}_i - 2\sum_{j=1}^{k} \mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_k + \sum_{j=1}^{k} \mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j$ since $\mathbf{v}_i^T\mathbf{v}_j = 1$ if and only if $i = j$
$= \mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k} \mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j$, as desired.

b) By definition, $J_k = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k}\mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j)$
$= \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k}\mathbf{v}_j^T\frac{1}{n}(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T)\mathbf{v}_j$
$= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k}\mathbf{v}_j^T\sum\mathbf{v}_j$
$= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{k}\lambda_j$ since $\mathbf{v}_j^T\sum\mathbf{v}_j = \lambda_j$, as desired.

c) Since $J_d = 0$, which is when $k = d$, $\sum_{j=1}^{d}\lambda_j = \frac{1}{n}\mathbf{x}_i^T\mathbf{x}_i$. Then, we can partition the summation to break at the point when the upper bound $k = d$:
$J_k = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{d}\lambda_j + \sum_{j=k+1}^{d}\lambda_j = 0 + \sum_{j=k+1}^{d}\lambda_j = \sum_{j=k+1}^{d}\lambda_j$.

This states that the reconstruction error when using a PCA projection of your data is the sum of the eigenvalues you throw out.

■

2 (ℓ_1-Regularization) Consider the ℓ_1 norm of a vector $\mathbf{x} \in \mathbb{R}^n$:

$$\|\mathbf{x}\|_1 = \sum_i |\mathbf{x}_i|.$$

Draw the norm-ball $B_k = \{\mathbf{x} : \|\mathbf{x}\|_1 \leq k\}$ for $k = 1$. On the same graph, draw the Euclidean norm-ball $A_k = \{\mathbf{x} : \|\mathbf{x}\|_2 \leq k\}$ for $k = 1$ behind the first plot. (Do not need to write any code, draw the graph by hand).

Show that the optimization problem

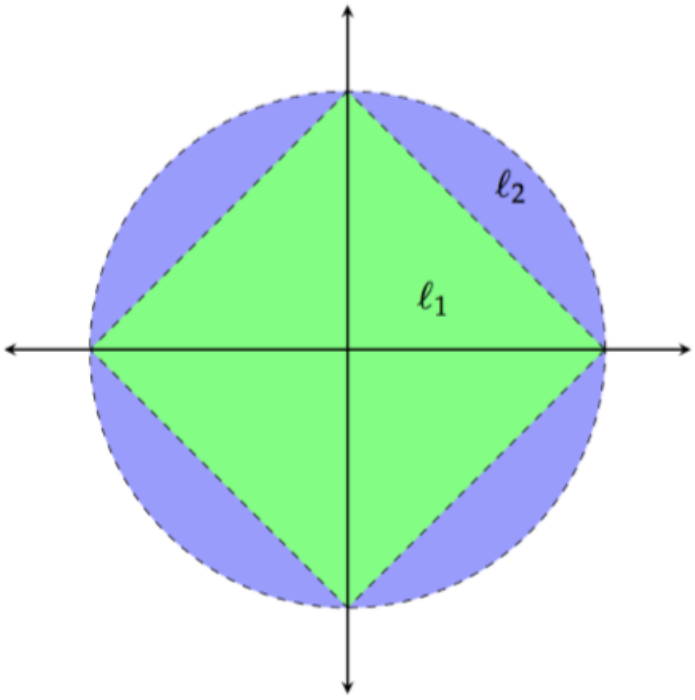
$$\begin{aligned} &\text{minimize: } f(\mathbf{x}) \\ &\quad \text{subj. to: } \|\mathbf{x}\|_p \leq k \end{aligned}$$

is equivalent to

$$\text{minimize: } f(\mathbf{x}) + \lambda\|\mathbf{x}\|_p$$

(hint: create the Lagrangian). With this knowledge, and the plots given above, argue why using ℓ_1 regularization (adding a $\lambda\|\mathbf{x}\|_1$ term to the objective) will give sparser solutions than using ℓ_2 regularization for suitably large λ.

a) Below is a picture of the norm balls. Note that the radius is k and for the ℓ_1 diamond, it includes points $(k, 0), (0, k), (-k, 0), (0, -k)$.



The optimization problem

$$\begin{aligned} &\text{minimize: } f(\mathbf{x}) \\ &\quad \text{subj. to: } \|\mathbf{x}\|_p \leq k \end{aligned}$$

is equivalent to $inf_x sup_{\lambda \geq 0} \mathcal{L}(\mathbf{x}, \lambda) = inf_x sup_{\lambda \geq 0} (f(\mathbf{x}) + \lambda(||\mathbf{x}||_p - k))$.

In its dual, we can flip the infimum and supremum.

$sup_{\lambda \geq 0} inf_x f(\mathbf{x}) + \lambda(||\mathbf{x}||_p - k) = \sup_{\lambda \geq 0} g(\lambda)$

The minimizing value of $f(\mathbf{x}) + \lambda(||\mathbf{x}||_p - k)$ over $\mathbf{x}$ is equivalent to the minimizing value of $f(\mathbf{x}) + \lambda||\mathbf{x}||_p$ because (after distributing) $-\lambda k$ doesn't depend on $\mathbf{x}$. Thus, optimizing $\mathbf{x}$ in the original optimization problem is the same as solving $Minimize : f(\mathbf{x}) + \lambda||\mathbf{x}||_p$ for some $\lambda \geq 0$, where the second term is the norm$p$*term*.

Looking at the plot and this result, consider ℓ_1 regularization as projecting the actual optimal solution of the optimization problem onto an ℓ_1 norm ball. Since the ℓ_1 ball has sharper edges, the probability of landing on an edge and not on the face of acceptable weight or coefficient values (where both elements of the vector are nonzero) is larger than in the ℓ_2 ball. Generalizing to higher dimensions, the ℓ_1 penalty is likelier to result in more weights or coefficients being zero compared to the ℓ_2 ball. This creates sparser solutions when using the ℓ_1 regularization.

■

Extra Credit (Lasso) Show that placing an equal zero-mean Laplace prior on each element of the weights $\boldsymbol{\theta}$ of a model is equivelent to ℓ_1 regularization in the Maximum-a-Posteriori estimate

$$\text{maximize: } \mathbb{P}(\boldsymbol{\theta}|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\mathcal{D})}.$$

Note the form of the Laplace distribution is

$$\text{Lap}(x|\mu, b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$$

where μ is the location parameter and $b > 0$ controls the variance. Draw (by hand) and compare the density $\text{Lap}(x|0,1)$ and the standard normal $\mathcal{N}(x|0,1)$ and suggest why this would lead to sparser solutions than a Gaussian prior on each elements of the weights (which correspond to ℓ_2 regularization).

■