CS 189 Introduction to Machine Learning
Fall 2018 EXAMREVIEW3

1 Connections between OLS, Ridge Regression, TLS, PCA, and CCA

We will review several topics we have learned so far: ordinary least-squares, ridge regression, total least squares, principle component analysis, and canonical correlation analysis. We emphasize their basic attributes, including the objective functions and the explicit form of their solutions. We will also discuss the connections and distinctions between these methods.

(a) What are the objective functions and closed-form solutions to OLS, ridge regression, and TLS? How do the probabilistic interpretations vary?

Solution:

The objective function for OLS is $\arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$, and the closed form solution is $(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$ if $\mathbf{X}^\top\mathbf{X}$ is full rank. If $\mathbf{X}^\top\mathbf{X}$ is not full rank, then there is not a unique solution; the least-norm solution uses the pseudo-inverse. The noise model is $\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$.

Ridge regression adds a regularization term: $\arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$. The closed form solution is $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$. The noise model is the same: $\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$. In Homework 2, we saw that the regularization dampens the effect of the noise on the training errors.

Total least squares assumes a different noise model, in which the measurements in $\mathbf{X}$ are also affected. The noise model is therefore $\mathbf{y} + \epsilon_y = (\mathbf{X} + \epsilon_x)\mathbf{w}$. The optimization problem is $\arg\min_{\epsilon_x, \epsilon_y} \|[\epsilon_x \quad \epsilon_y]\|_F^2$, subject to $(\mathbf{X} + \epsilon_x)\mathbf{w} = \mathbf{y} + \epsilon_y$. The closed form solution is $(\mathbf{X}^\top\mathbf{X} - \sigma_{d+1}^2\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$, where σ_{d+1} is the least singular value of $[\mathbf{X} \quad \mathbf{y}]$.

(b) Consider the matrix inversion in the solution to OLS, ridge regression, and TLS. How do the eigenvalues compare to those of the matrix $\mathbf{X}^\top\mathbf{X}$?

Solution: OLS inverts the matrix $\mathbf{X}^\top\mathbf{X}$, ridge inverts the regularized matrix $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ and TLS inverts $\mathbf{X}^\top\mathbf{X} - \sigma_{d+1}^2\mathbf{I}$ matrix.

We can write $\mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ with eigenvalues $\lambda_1, \ldots, \lambda_d$. Then in OLS, we invert these eigenvalues directly. In ridge regression, λ is added to each value before inverting, which ensures numerical stability. In TLS, σ_{d+1}^2 is subtracted from each. This may decrease the numerical stability, but it can be viewed as removing the components generated by noise.

$$\begin{aligned}\mathbb{E}\left[\mathbf{X}^\top\mathbf{X}\right] &= \mathbb{E}\left[(\mathbf{X}^* + \epsilon_x)^\top(\mathbf{X}^* + \epsilon_x)\right] \\ &= (\mathbf{X}^*)^\top\mathbf{X}^* + \mathbb{E}\left[\epsilon_x^\top\epsilon_x\right] \\ &= (\mathbf{X}^*)^\top\mathbf{X}^* + \sigma_{noise}^2\mathbf{I}\end{aligned} \tag{1}$$

In the above, we assume that the noise on the measured features $\mathbf{X}$ is independent across both features and samples, and has variance σ^2_{noise}. In this case, even the smallest eigenvalue will not be not zero. TLS tries to explicitly remove the noise by subtracting σ^2_{d+1}.

(c) Suppose you have a data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and output $\mathbf{y} \in \mathbb{R}^{n\times 1}$. Use PCA to compute the first k principal components of $[\mathbf{X} \quad \mathbf{y}]$. Describe how this solution would relate to performing TLS on the problem.

Solution: To write down the principle components, we first write the SVD decomposition of the matrix $[\mathbf{X} \quad \mathbf{y}] = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where the singular values are ordered by magnitude. Then the principle components are given by $\mathbf{v}_1, \ldots, \mathbf{v}_k$, the first k columns of $\mathbf{V}$. Remember that projecting the data onto these components is equivalent to finding the best rank-k approximation to the original matrix $[\mathbf{X} \quad \mathbf{y}]$.

On the other hand, recall that TLS minimizes $\| \begin{bmatrix}\boldsymbol{\epsilon}_X & \boldsymbol{\epsilon}_y\end{bmatrix} \|_F^2$, subject to the constraint of:

$$\begin{bmatrix}\mathbf{X}+\boldsymbol{\epsilon}_X & \mathbf{y}+\boldsymbol{\epsilon}_y\end{bmatrix} \begin{bmatrix}\mathbf{w}^\top & -1\end{bmatrix}^\top = \mathbf{0}\,.$$

Therefore, we find the minimum perturbation of $[\mathbf{X} \quad \mathbf{y}]$ that has at least one zero eigenvalue. On the homework, we see that this is equivalent to setting $\begin{bmatrix}\mathbf{X}+\boldsymbol{\epsilon}_X & \mathbf{y}+\boldsymbol{\epsilon}_y\end{bmatrix}$ to be the best rank-d approximation to the original matrix $[\mathbf{X} \quad \mathbf{y}]$. Therefore, performing TLS is like using PCA and projecting onto the first d principle components.

(d) Suppose you have a problem of fitting $\mathbf{y}$ from $\mathbf{X} \in \mathbb{R}^{n\times d}$. d is large in this case, and you want to reduce the dimensionality before doing the fitting. You have two choices for the dimensionality reduction: the first one being do PCA on $\mathbf{X}$ and the second one being use CCA between $\mathbf{X}$ and $\mathbf{y}$ to reduce dimension of $\mathbf{X}$ with $\mathbf{X}\mathbf{U}_p$, where $\mathbf{U}_p \in \mathbb{R}^{d\times p}$ is the p-dimensional projection matrix solved using CCA. Which of them is better in general?

Solution: In general, CCA will be better. PCA captures the largest variation directions in $\mathbf{X}$, however, that is not necessarily the most predictive direction for $\mathbf{y}$. One could come up with a problem that the least variation direction determines the labels $\mathbf{y}$. In that case, PCA will completely fail. CCA on the other hand, captures the correlation between the projected $\mathbf{X}$ and $\mathbf{y}$ and thus does not suffer from this problem.

2 Linear regression (16 points)

In this problem, we study the loss function for ridge regression:

$$\frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2, \tag{2}$$

where X a data matrix in $\mathbb{R}^{n\times d}$ and y is the response in $\mathbb{R}^n$. Let the regularization weight $\lambda > 0$. Recall that the closed-form solution to ridge regression is

$$\hat{w} = (X^T X + \lambda I_d)^{-1} X^T y. \tag{3}$$

where $I_d \in \mathbb{R}^{d\times d}$ is an identity matrix of dimension d.

(a) (8 points) Augment the matrix X with d additional rows $ce_1^T, ce_2^T, \ldots, ce_d^T$ to get the matrix $X' \in \mathbb{R}^{n+d \times d}$, where c is a given constant and e_i^T is a unit vector whose ith element is 1 and the rest of the elements are zero, and augment y with d zeros to get $y' \in \mathbb{R}^{n+d}$:

$$X' = \begin{bmatrix} X \\ ce_1^T \\ ce_2^T \\ \vdots \\ ce_d^T \end{bmatrix} \qquad y' = \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Write out the closed form solution for w' for the ordinary least squares problem on X', y'.

Derive the concrete value of c that corresponds to the ridge regression problem (2) in terms of λ. Conclude that the ridge regression estimate for w can be obtained by doing ordinary least squares on an augmented dataset.

Solution: Augmenting the matrix X with d additional rows $ce_1^T, ce_2^T, \ldots, ce_d^T$, we get a new matrix X' such that

$$X' = \begin{bmatrix} X \\ ce_1^T \\ ce_2^T \\ \vdots \\ ce_d^T \end{bmatrix} \tag{4}$$

Augmenting y with d zeros, we get a new vector y' such that

$$y' = \begin{bmatrix} y \\ 0_d \end{bmatrix} \tag{5}$$

where 0_d is an all-zero vector in $\mathbb{R}^d$. Solving the least square problem with X', y', we have

$$\begin{aligned} w' &= (X'^T X')^{-1} X' y' \\ &= \left(\begin{bmatrix} X^T & ce_1 & ce_2 & \ldots & ce_d \end{bmatrix} \begin{bmatrix} X \\ ce_1^T \\ ce_2^T \\ \vdots \\ ce_d^T \end{bmatrix}\right)^{-1} \begin{bmatrix} X^T & ce_1 & ce_2 & \ldots & ce_d \end{bmatrix} \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= (X^T X + c^2 \sum_{i=1}^{d} e_i e_i^T)^{-1} X^T y \\ &= (X^T X + c^2 I_d)^{-1} X^T y. \end{aligned}$$

The closed form solution of the ridge regression is

$$w' = (X^T X + \lambda I_d)^{-1} X^T y. \tag{6}$$

Therefore, we have $\lambda = c^2$. So $c = \sqrt{\lambda}$ or $c = -\sqrt{\lambda}$.

(b) (8 points) **Prove that applying gradient descent on the ridge regression loss function with a suitable fixed step size results in geometric convergence.** If $X^\top X$ has maximum and minimum eigenvalues M and m, **what fixed step size should we choose as a function of M, m, and ridge weight λ?**

Hint: Reduce the problem to ordinary least squares. Recall that applying gradient descent on the least squares problem

$$\min_x f(x) = \min_x \frac{1}{2}\|Ax - b\|_2^2$$

results in geometric convergence when $A^T A$ is positive definite and using a constant step size $\gamma = \frac{2}{\lambda_{\min}(A^T A) + \lambda_{\max}(A^T A)}$. Geometric convergence means that $f(x_k) - f(x^*) \leq c' Q^k$ for some $0 \leq Q < 1$ and for some $c' > 0$. You may use this result without proof.

Solution: Let m, M denote the smallest and largest eigenvalues of $X^T X$, respectively. We can formulate the problem as

$$\min_x \frac{1}{2}\|A'x - b'\|_2^2$$

where

$$A' = \begin{bmatrix} X \\ \lambda I_d \end{bmatrix} \text{ and } b' = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

We observe that $A'^T A' = X^T X + \lambda I_d \succeq m + \lambda > 0$ (positive definite). That is, we reduce the given problem to the quadratic problem above. The smallest and largest eigenvalues of $X^T X + \lambda I_d$ are $m + \lambda, M + \lambda$, respectively. Then, by the given result, geometric convergence is attained for fixed step size $\gamma = \frac{2}{m + M + 2\lambda}$.

3 Classification and Separability

In this problem, we will explore how adding the label as a feature affects separability of a dataset.

(a) Consider a dataset $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, $y \in \{-1, 1\}$ which is *not* linearly separable.

Now for each point in the dataset, consider augmenting each feature vector with the label, so that we get a new feature vector $z_i \in \mathbb{R}^{d+1}$, where:

$$z_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Show that using the augment feature vectors, the dataset is linearly separable.

Solution:

The hyperplane defined by $w = e_{d+1}$ separates the points.

(b) How will we expect the linear decision boundary defined above to perform when applied to a new unseen set of features $\mathbf{X}^{new} \in \mathbb{R}^{n_2 \times d}$?

Solution: Well first of all, it's unclear how we would even apply w from part (a) to the new dataset, since there are no labels for the new data (the dimensions don't match). Additionally, the classifier from part (a) put 0 weight on all features other than the labels, so we can't use it to discriminate the labels of new data based on the features given in $\mathbf{X}^{new}$.

4 Kernel PCA (24 points)

In lectures, discussion, and homework, we learned how to use PCA to do dimensionality reduction by projecting the data to a subspace that captures most of the variability. This works well for data that is roughly Gaussian shaped, but many real-world high dimensional datasets have underlying low-dimensional structure that is not well captured by linear subspaces. However, when we lift the raw data into a higher-dimensional feature space by means of a nonlinear transformation, the underlying low-dimensional structure once again can manifest as an approximate subspace. Linear dimensionality reduction can then proceed. As we have seen in class so far, kernels are an alternate way to deal with these kinds of nonlinear patterns without having to explicitly deal with the augmented feature space. This problem asks you to discover how to apply the "kernel trick" to PCA.

Let $\mathbf{X} \in \mathbb{R}^{n \times \ell}$ be the data matrix, where n is the number of samples and ℓ is the dimension of the raw data. Namely, the data matrix contains the data points $\mathbf{x}_j \in \mathbb{R}^{\ell}$ as rows

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times \ell}. \tag{7}$$

(a) (5 pts) **Compute $\mathbf{X}\mathbf{X}^\top$ in terms of the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n \times n}, \mathbf{\Sigma} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{V} \in \mathbb{R}^{\ell \times \ell}$.** Notice that $\mathbf{X}\mathbf{X}^\top$ is the matrix of pairwise Euclidean inner products for the data points. **How would you get $\mathbf{U}$ if you only had access to $\mathbf{X}\mathbf{X}^\top$?** You may assume that the data is centered: That is assume $\frac{1}{N}\sum_{i=1}^{N} x_i = 0$

Solution: By plugging in the compact SVD decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and using $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ we get

$$\mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\top.$$

Similarly with $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$ we get

$$\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top.$$

Notice from the last line that $\mathbf{U}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$ with eigenvalues $\sigma_1^2, \sigma_2^2, \ldots, \sigma_\ell^2$ where $\sigma_1, \sigma_2, \ldots, \sigma_d$ are the singular values of $\mathbf{X}$ and can therefore be computed by performing an eigendecomposition of $\mathbf{X}\mathbf{X}^\top$.

(b) (7 pts) Given a new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, one central use of PCA is to compute the projection of $\mathbf{x}_{test}$ onto the subspace spanned by the k top singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.

Express the scalar projection $z_j = \mathbf{v}_j^\top \mathbf{x}_{test}$ onto the j-th principal component as a function of the inner products

$$\mathbf{X}\mathbf{x}_{test} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_{test} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_{test} \rangle \end{pmatrix}. \tag{8}$$

Assume that all diagonal entries of $\mathbf{\Sigma}$ are nonzero and non-increasing, that is $\sigma_1 \geq \sigma_2 \geq \cdots > 0$.

Hint: Express $\mathbf{V}^\top$ in terms of the singular values $\mathbf{\Sigma}$, the left singular vectors $\mathbf{U}$ and the data matrix $\mathbf{X}$. If you want to use the compact form of the SVD, feel free to do so.

Solution: By multiplying the compact SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ on both sides with $\mathbf{U}^\top$, we get $\mathbf{U}^\top\mathbf{X} = \mathbf{\Sigma}\mathbf{V}^\top$ and multiplying both sides of the new equation with $\mathbf{\Sigma}^{-1}$, we obtain

$$\mathbf{V}^\top = \mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{X}.$$

Therefore we get

$$z_j = \mathbf{v}_j^\top \mathbf{x}_{test} = \frac{1}{\sigma_j}\mathbf{u}_j^\top \mathbf{X}\mathbf{x}_{test}$$

.

(c) (12 pts) How would you define kernelized PCA for a general kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ (to replace the Euclidean inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$)? You may assume that the data remains centered after the feature map ϕ induced by the kernel function k. That is assume $\frac{1}{N}\sum_{i=1}^N \phi(x_i) = 0$

Describe this in terms of a procedure which takes as inputs the training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^\ell$ and the new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, and outputs the analog of the previous part's z_j coordinate in the kernelized PCA setting. You should include how to compute $\mathbf{U}$ from the data, as well as how to compute the analog of $\mathbf{X}\mathbf{x}_{test}$ from the previous part.

Invoking the SVD or computing eigenvalues/eigenvectors is fine in your procedure, as long as it is clear what matrix is having its SVD or eigenvalues/eigenvectors computed. The kernel $k(\cdot, \cdot)$ can be used as a black-box function in your procedure as long as it is clear what arguments it is being given.

Solution: For kernelizing PCA, we replace inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with $k(\mathbf{x}_i, \mathbf{x}_j)$ and $\langle \mathbf{x}_i, \mathbf{x}_{test} \rangle$ with $k(\mathbf{x}_i, \mathbf{x}_{test})$, the procedure is then:

(a) Obtain the vectors $\mathbf{u}_j$ as eigenvectors from the eigendecomposition of the kernelized counterpart of the Gram matrix: $\mathbf{K} \in \mathbb{R}^{n \times n}$ with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The eigenvalues should be sorted in decreasing order. They are all non-negative real numbers because of the properties of kernels — the K matrix must be positive semi-definite.

(b) Kernelize the inner products $z_j = \frac{1}{\sigma_j}\mathbf{u}_j^\top \mathbf{X}\mathbf{x}_{test}$ from the previous part by using:

$$z_j = \frac{1}{\sigma_j}\mathbf{u}_j^\top \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_{test}) \\ k(\mathbf{x}_2, \mathbf{x}_{test}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_{test}) \end{pmatrix}, \tag{9}$$

where the σ_j are the square roots of the eigenvalues for the martix K above generated by using the kernel on all pairs of training points. Because these are non-negative real numbers, the square root is well defined.

5 Gradient Descent and Convexity

Here is the **Smoothed version of the Hinge loss function** (with parameter t):

$$f(y) = \begin{cases} \frac{1}{2} - ty & \text{if } ty \leq 0, \\ \frac{1}{2}(1-ty)^2 & \text{if } 0 < ty < 1, \\ 0 & \text{if } 1 \leq ty \end{cases}$$

Given $x_1, x_2, \ldots, x_n$ data points and $y_1, y_2, \ldots, y_n$ labels. we define the optimization problems as follows:

$$\min_w \frac{1}{n}\sum_{i=1}^{n} f(w^\top x_i - y_i)$$

Define: $L(w) = \frac{1}{n}\sum_{i=1}^{n} f(w^\top x_i - y_i)$

- Is $L(w)$ convex?
- Write out the Gradient Descent update equation.
- Write out the Stochastic Gradient Descent update equation.

Solution:

- Define $g(y) = \frac{\partial f}{\partial y}$.

$$g(y) = \begin{cases} -t & \text{if } ty \leq 0, \\ -t(1-ty) & \text{if } 0 < ty < 1, \\ 0 & \text{if } 1 \leq ty \end{cases}$$

Taking the second derivative,

$$\frac{\partial g}{\partial y} = \begin{cases} 0 & \text{if } ty \leq 0, \\ t^2 & \text{if } 0 < ty < 1, \\ 0 & \text{if } 1 \leq ty \end{cases}$$

This implies that the second derivative is always greater than or equal to 0, so the function f is convex. Since the function L is the sum of convect functions, it follows that $L(w)$ is convex.

- Using chain rule

$$\frac{\partial f(w^\top x - y)}{\partial w} = g(w^\top x - y)\frac{\partial (w^\top x - y)}{w} = g(w^\top x - y)x$$

Hence the gradient descent update rule is

$$w = w - \alpha \frac{1}{N} \sum_{i=1}^{N} g(w^\top x_i - y_i)x_i$$

- Pick an index i uniofrmly at random from $\{1, ..., n\}$

The stochastic gradient descent update rule is

$$w = w - \alpha g(w^\top x_i - y_i)x_i$$