Homework #1

CSE 546: Machine Learning
Michael Ross

1 Gaussians

Recall that for any vector $u \in \mathbb{R}^n$ we have $||u||_2^2 = u^T u = \sum_{i=1}^n u_i^2$ and $||u||_1 = \sum_{i=1}^n |u_i|$. For a matrix $A \in \mathbb{R}^{n\times n}$ we denote $|A|$ as the determinant of A. A multivariate Gaussian with mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{n\times n}$ has a probability density function $p(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$ which we denote as $\mathcal{N}(\mu,\Sigma)$.

1. *[4 points]* Let

- $\mu_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$
- $\mu_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & -1.8 \\ -1.8 & 2 \end{bmatrix}$
- $\mu_3 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$ and $\Sigma_3 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

For each $i = 1,2,3$ on a separate plot:

a. Draw $n = 100$ points $X_{i,1},\dots,X_{i,n} \sim \mathcal{N}(\mu_i,\Sigma_i)$ and plot the points as a scatter plot with each point as a triangle marker (Hint: use `numpy.random.randn` to generate a mean-zero independent Gaussian vector, then use the properties of Gaussians to generate X).

b. Compute the sample mean and covariance matrices $\widehat{\mu}_i = \frac{1}{n}\sum_{j=1}^n X_{i,j}$ and $\widehat{\Sigma}_i = \frac{1}{n-1}\sum_{j=1}^n (X_{i,j} - \widehat{\mu}_i)^2$. Compute the eigenvectors of $\widehat{\Sigma}_i$. Plot the eigenvectors as line segments originating from $\widehat{\mu}_i$ and have magnitude equal to the square root of their corresponding eigenvalues.

c. If $(u_{i,1},\lambda_{i,1})$ and $(u_{i,2},\lambda_{i,2})$ are the eigenvector-eigenvalue pairs of the sample covariance matrix with $\lambda_{i,1} \geq \lambda_{i,2}$ and $||u_{i,1}||_2 = ||u_{i,2}||_2 = 1$, for $j = 1,\dots,n$ let $\widetilde{X}_{i,j} = \begin{bmatrix} \frac{1}{\sqrt{\lambda_{i,1}}} u_{i,1}^T (X_{i,j} - \widehat{\mu}_i) \\ \frac{1}{\sqrt{\lambda_{i,2}}} u_{i,2}^T (X_{i,j} - \widehat{\mu}_i) \end{bmatrix}$. Plot these new points as a scatter plot with each point as a circle marker.

For each plot, make sure the limits of the plot are square around the origin (e.g., $[-c,c]\times[-c,c]$ for some $c > 0$).

2 MLE and Bias Variance Tradeoff

Recall that for any vector $u \in \mathbb{R}^n$ we have $\|u\|_2^2 = u^T u = \sum_{i=1}^n u_i^2$ and $||u||_1 = \sum_{i=1}^n |u_i|$. Unless otherwise specified, if P is a probability distribution and $x_1,\dots,x_n \sim P$ then it can be assumed each each x_i is drawn iid from P.
2. *[1 points]* Let $x_1,\dots,x_n \sim \text{uniform}(0,\theta)$ for some θ. What is the Maximum likelihood estimate for θ?

Answer:

$\text{uniform}(o,\theta) = 1/\theta$ if $0 < x < \theta$ and 0 everywhere else

$\mathcal{L}(\theta|x) = \prod_{i=1}^{n} \frac{1}{\theta}$
$\log(\mathcal{L}(\theta|x)) = \sum_{i=1}^{n} \log(\frac{1}{\theta})$
$\log(\mathcal{L}(\theta|x)) = -n\log(\theta)$
$\frac{d}{d\theta}\log(\mathcal{L}(\theta|x)) = -\frac{n}{\theta}$
This shows that the log likelihood decreases with increasing θ so
$\widehat{\theta} = \max(x_i)$

3. *[2 points]* Let $(x_1, y_1), \dots, (x_n, y_n)$ be drawn at random from some population where each $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and let $\widehat{w} = \arg\min_w \sum_{i=1}^{n}(y_i - w^T x_i)^2$. Suppose we have some test data $(\widetilde{x}_1, \widetilde{y}_1), \dots, (\widetilde{x}_m, \widetilde{y}_m)$ drawn at random from the population as the training data. If $R_{tr}(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$ and $R_{te}(w) = \frac{1}{m}\sum_{i=1}^{m}(\widetilde{y}_i - w^T \widetilde{x}_i)^2$. Prove that

$$\mathbb{E}[R_{tr}(\widehat{w})] \leq \mathbb{E}[R_{te}(\widehat{w})]$$

where the expectations are over all that is random in each expression. Do not assume any model for y_i given x_i (e.g., linear plus Gaussian noise). [This is exercise 2.9 from HTF, originally from Andrew Ng.]

4. *[8 points]* Let random vector $X \in \mathbb{R}^d$ and random variable $Y \in \mathbb{R}$ have a joint distribution $P_{XY}(X, Y)$. Assume $\mathbb{E}[X] = 0$ and define $\Sigma = \mathrm{Cov}(X) = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T]$ with eigenvalues $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_d$ and orthonormal eigenvectors $v_1, \dots, v_d$ such that $\Sigma = \sum_{i=1}^{d} \alpha_i v_i v_i^T$. For $(X, Y) \sim P_{XY}$ assume that $Y = X^T w + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \sigma^2)$ such that $\mathbb{E}_{Y|X}[Y|X = x] = x^T w$. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ where each $(x_i, y_i) \sim P_{XY}$. For some $\lambda > 0$ let

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n}(y_i - x_i^T w)^2 + \lambda\|w\|_2^2$$

If $\mathbf{X} = [x_1, \dots, x_n]^T$, $\mathbf{y} = [y_1, \dots, y_n]^T$, $\boldsymbol{\epsilon} = [\epsilon_1, \dots, \epsilon_n]^T$ then it can be shown that

$$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}. \tag{1}$$

Note the notational difference between a random X of $(X, Y) \sim P_{XY}$ and the $n \times d$ matrix $\mathbf{X}$ where each row is drawn from P_X. Realizing that $\mathbf{X}^T\mathbf{X} = \sum_{i=1}^{n} x_i x_i^T$, by the law of large numbers we have $\frac{1}{n}\mathbf{X}^T\mathbf{X} \to \Sigma$ as $n \to \infty$. In your analysis assume n is large and make use of the approximation $\mathbf{X}^T\mathbf{X} = n\Sigma$. Justify all answers.

a. Show Equation (1).

Answer:

$\widehat{w} = \arg\min_w \sum_{i=1}^{n}(y_i - x_i^T w)^2 + \lambda\|w\|_2^2$

Switching to matrix notation:
$\widehat{w} = \arg\min_w \|\mathbf{X}w - \mathbf{y}\|_2^2 + \lambda\|w\|_2^2$

Set derivative to zero:
$\nabla_w(\|\mathbf{X}w - \mathbf{y}\|_2^2 + \lambda\|w\|_2^2) = 0$
$2\mathbf{X}^T(\mathbf{X}\widehat{w} - \mathbf{y}) + 2\lambda\widehat{w} = 0$
$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\widehat{w} + \lambda\widehat{w}$
$\mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X} + \lambda I)\widehat{w}$
$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$

b. Show that $\widehat{w}$ of Equation 1 can also be written as

$$\widehat{w} = w - \lambda(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$$

Answer:

$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$
$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T(\mathbf{X}w + \boldsymbol{\epsilon})$
$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{X}w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$
$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{X}w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda I w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda I w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$
$\widehat{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}(\mathbf{X}^T\mathbf{X} + \lambda I)w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda I w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$
$\widehat{w} = w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}$

c. For general $\widehat{f}_{\mathcal{D}}(x)$ and $\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$, we showed in class that the bias variance decomposition is stated as

$$\mathbb{E}_{XY,\mathcal{D}}[(Y - \widehat{f}_{\mathcal{D}}(X))^2] = \mathbb{E}_X\Big[\mathbb{E}_{Y|X,\mathcal{D}}\big[(Y - \widehat{f}_{\mathcal{D}}(X))^2|X = x\big]\Big]$$

where

$$\mathbb{E}_{Y|X,\mathcal{D}}\big[(Y - \widehat{f}_{\mathcal{D}}(X))^2|X = x\big] = \underbrace{\mathbb{E}_{Y|X}[(Y - \eta(x))^2|X = x]}_{\text{Irreducible error}} + \underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2}_{\text{Bias-squared}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]}_{\text{Variance}}.$$

In what follows, use our particular problem setting with $\widehat{f}_{\mathcal{D}}(x) = \widehat{w}^T x$.
Irreducible error: What is $\mathbb{E}_X\Big[\mathbb{E}_{Y|X}[(Y - \eta(x))^2|X = x]\Big]$?
Answer:

$\mathbb{E}_X\Big[\mathbb{E}_{Y|X}[(Y - \eta(x))^2|X = x]\Big] = \mathbb{E}_X\Big[\mathbb{E}_{Y|X}[(X^T w + \epsilon - X^T w)^2|X = x]\Big]$
$= \mathbb{E}_X\Big[\mathbb{E}_{Y|X}[(\epsilon)^2|X = x]\Big]$
$= \mathbb{E}_X\Big[\mathbb{E}_{Y|X}[(\epsilon)^2|X = x] - \mathbb{E}_{Y|X}[(\epsilon)|X = x]^2\Big]$ since $\mathbb{E}_{Y|X}[(\epsilon)|X = x] = 0$
$= \sigma^2$

d. Bias-squared: Use the approximation $\mathbf{X}^T\mathbf{X} = n\Sigma$ to show that

$$\mathbb{E}_X\big[(\eta(X) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(X)])^2\big] = \sum_{i=1}^{d} \frac{\lambda^2 (w_i^T v_i)^2 \alpha_i}{(n\alpha_i + \lambda)^2} \leq \max_{j=1,\dots,d} \frac{\lambda^2 \alpha_j \|w\|_2^2}{(n\alpha_j + \lambda)^2}$$

Answer:

$\mathbb{E}_X\big[(\eta(X) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(X)])^2\big] = \mathbb{E}_X\big[(X^T w - \mathbb{E}_{\mathcal{D}}[X^T\widehat{w}])^2\big]$
$= \mathbb{E}_X\big[(X^T w - \mathbb{E}_{\mathcal{D}}[X^T(w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon})])^2\big]$
$= \mathbb{E}_X\big[(X^T w - X^T w + \mathbb{E}_{\mathcal{D}}[X^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda w + X^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\boldsymbol{\epsilon}])^2\big]$
$= \mathbb{E}_X\big[\mathbb{E}_{\mathcal{D}}[X^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\lambda w]^2\big]$ since $\mathbf{X}^T$ and $\boldsymbol{\epsilon}$ are mean zero and independent
$= \mathbb{E}_X\big[\mathbb{E}_{\mathcal{D}}[X^T(n\Sigma + \lambda I)^{-1}\lambda w]^2\big]$
$= \mathbb{E}_X\big[\mathbb{E}_{\mathcal{D}}[X^T(n\sum_{i=1}^d \alpha_i v_i v_i^T + \lambda I)^{-1}\lambda w]^2\big]$

$\sum_{i=1}^d v_i v_i^T = I$ by definition

$= \mathbb{E}_X\big[\mathbb{E}_{\mathcal{D}}[X^T(\sum_{i=1}^d (n\alpha_i + \lambda) v_i v_i^T)^{-1}\lambda w]^2\big]$

Since the eigenvalues of A^{-1} are the reciprocals of the eigenvalues of A

$= \mathbb{E}_X\big[\mathbb{E}_{\mathcal{D}}[X^T\sum_{i=1}^d v_i v_i^T \lambda w/(n\alpha_i + \lambda)]^2\big]$

Drop the $\mathbb{E}_{\mathcal{D}}$ since everything is now independent of the D

$= \mathbb{E}_X \big[\sum_{i,j} \lambda^2 w^T v_i v_i^T X X^T v_j v_j^T w/((n\alpha_i + \lambda)(n\alpha_j + \lambda)) \big]$
$= \mathbb{E}_X \big[\sum_{i,j,k} \alpha_k \lambda^2 w^T v_i v_i^T v_k v_k^T v_j v_j^T w/((n\alpha_i + \lambda)(n\alpha_j + \lambda)) \big]$
$= \mathbb{E}_X \big[\sum_{i,j,k} \alpha_k \lambda^2 w^T v_i \delta_{i,k} \delta_{k,j} v_j^T w/((n\alpha_i + \lambda)(n\alpha_j + \lambda)) \big]$ due to orthonormality of eigenvectors
$= \mathbb{E}_X \big[\sum_{i=1}^d \alpha_i \lambda^2 w^T v_i v_i^T w/(n\alpha_i + \lambda)^2 \big]$
$= \sum_{i=1}^d \alpha_i \lambda^2 (w^T v_i)^2/(n\alpha_i + \lambda)^2$

e. Variance: Use the approximation $\mathbf{X}^T\mathbf{X} = n\Sigma$ to show that

$$\mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(X)] - \widehat{f}_{\mathcal{D}}(X))^2] \big] = \sum_{i=1}^d \frac{\sigma^2 \alpha_i^2 n}{(\alpha_i n + \lambda)^2} \leq \frac{d\sigma^2 \alpha_1^2 n}{(\alpha_1 n + \lambda)^2}$$

Answer:

$\mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(X)] - \widehat{f}_{\mathcal{D}}(X))^2] \big] = \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[x^T \widehat{w}] - x^T \widehat{w})^2] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[x^T(w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon})] - x^T(w - (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w + (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}))^2] \big]$

Since X^T and w is are independent of the data
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(x^T w + \mathbb{E}_{\mathcal{D}}[-x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w + x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}] - x^T w + x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w - x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}))^2] \big]$

Since $\mathbf{X}^T$ and ϵ are independent and mean zero
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[-x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w] + x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \lambda w - x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}))^2] \big]$

Substitution from last part
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(-X^T \sum_{i=1}^d v_i v_i^T \lambda w/(n\alpha_i + \lambda) + X^T \sum_{i=1}^d v_i v_i^T \lambda w/(n\alpha_i + \lambda) - x^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}))^2] \big]$

$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(-X^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \boldsymbol{\epsilon}))^2] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[(-X^T \sum_{i=1}^d v_i v_i^T \mathbf{X}^T \boldsymbol{\epsilon}/(n\alpha_i + \lambda))^2] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j} X^T v_i v_i^T \mathbf{X}^T \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T \mathbf{X} v_j v_j^T X/((n\alpha_i + \lambda)(n\alpha_j + \lambda))] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j,k} X^T v_i v_i^T \mathbf{X}^T \sigma^2 v_k v_k^T \mathbf{X} v_j v_j^T X/((n\alpha_i + \lambda)(n\alpha_j + \lambda))] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j,k} \sigma^2 X^T v_i v_i^T \mathbf{X}^T \mathbf{X} v_k v_k^T v_j v_j^T X/((n\alpha_i + \lambda)(n\alpha_j + \lambda))] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j,k,l} \sigma^2 X^T v_i v_i^T n\alpha_l v_l v_l^T v_k v_k^T v_j v_j^T X/((n\alpha_i + \lambda)(n\alpha_j + \lambda))] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j,k,l} n\alpha_l \sigma^2 X^T v_i \delta_{i,l} \delta_{l,k} \delta_{k,j} v_j^T X/((n\alpha_i + \lambda)(n\alpha_j + \lambda))] \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_i n\alpha_i \sigma^2 X^T v_i v_i^T X/(n\alpha_i + \lambda)^2 \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_i n\alpha_i \sigma^2 v_i^T X X^T v_i/(n\alpha_i + \lambda)^2 \big]$
$= \mathbb{E}_X \big[\mathbb{E}_{\mathcal{D}}[\sum_{i,j} n\alpha_i \sigma^2 v_i^T \alpha_j v_j v_j^T v_i/(n\alpha_i + \lambda)^2 \big]$
$= \sum_i n\alpha_i^2 \sigma^2/(n\alpha_i + \lambda)^2$

f. Assume $\Sigma = \alpha_1 I$ for some $\alpha_1 > 0$. Show that for the approximation $\mathbf{X}^T\mathbf{X} = n\Sigma$ we have

$$\mathbb{E}_{XY,\mathcal{D}}[(Y - \widehat{f}_{\mathcal{D}}(X))^2] = \sigma^2 + \frac{\lambda^2 \alpha_1 \|w\|_2^2}{(\alpha_1 n + \lambda)^2} + \frac{d\sigma^2 \alpha_1^2 n}{(\alpha_1 n + \lambda)^2}$$

What is the $\lambda^\star$ that minimizes this expression? In a sentence each describe how varying each parameter (e.g., $\|w\|_2$, d, σ^2) affects the size of $\lambda^\star$ and if this makes intuitive sense. Plug this $\lambda^\star$ back into the expression and comment on how this result compares to the $\lambda = 0$ solution. It may be helpful to use $\frac{1}{2}(a + b) \leq \max\{a, b\} \leq a + b$ for any $a, b > 0$ to simplify the expression.

Answer:

$\mathbb{E}_{Y|X,\mathcal{D}}\big[(Y-\widehat{f}_{\mathcal{D}}(X))^2|X=x\big] = \mathbb{E}_{Y|X}[(Y-\eta(x))^2|X=x]+(\eta(x)-\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2+\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)]-\widehat{f}_{\mathcal{D}}(x))^2]$
$= \sigma^2 + \sum_{i=1}^d \alpha_i \lambda^2 (w^T v_i)^2/(n\alpha_i+\lambda)^2 + \sum_{i=1}^d n\alpha_i^2\sigma^2/(n\alpha_i+\lambda)^2$

If $\Sigma = \alpha_1 I$ then v_i are vectors with 1 in the i-th element and zeroes everywhere else.
$= \sigma^2 + \frac{\alpha_1\lambda^2}{(n\alpha_1+\lambda)^2}\sum_{i=1}^d (w_i)^2 + dn\alpha_1^2\sigma^2/(n\alpha_1+\lambda)^2$
$= \sigma^2 + \alpha_1\lambda^2\|w\|_2^2/(n\alpha_1+\lambda)^2 + dn\alpha_1^2\sigma^2/(n\alpha_1+\lambda)^2$

g. Assume that $\alpha_1 > \alpha_2 = \alpha_3 = \cdots = \alpha_d$ and furthermore, that $w/\|w\|_2 = v_1$. Show that for the approximation $\mathbf{X}^T\mathbf{X} = n\Sigma$ we have

$$\mathbb{E}_{XY,\mathcal{D}}[(Y-\widehat{f}_{\mathcal{D}}(X))^2] = \sigma^2 + \frac{\lambda^2\alpha_1\|w\|_2^2}{(\alpha_1 n+\lambda)^2} + \frac{\sigma^2 n\alpha_1^2}{(\alpha_1 n+\lambda)^2} + \frac{\sigma^2 n\alpha_2^2(d-1)}{(\alpha_2 n+\lambda)^2}$$

It can be shown that $\lambda^\star = \frac{\sigma^2+\sigma^2(d-1)\alpha_1/\alpha_2}{\|w\|_2^2}$ approximately minimizes this expression. In a sentence each describe if this makes intuitive sense, comparing to the solution of the last problem.

h. As λ increases, how does the bias and variance terms behave?

3 Programming: Ridge Regression on MNIST

5. *[10 points]* In this problem we will implement a least squares classifier for the MNIST data set. The task is to classify handwritten images of numbers between 0 to 9.

You are **NOT** allowed to use any of the prebuilt classifiers in `sklearn`. Feel free to use any method from `numpy` or `scipy`. Remember: if you are inverting a matrix in your code, you are probably doing something wrong (Hint: look at `scipy.linalg.solve`).

Get the data from `https://pypi.python.org/pypi/python-mnist`.
Load the data as follows:

```
from mnist import MNIST

def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0
```

You can visualize a single example by reshaping it to its original 28×28 image shape.

a. In this problem we will choose a linear classifier to minimize the least squares objective:

$$\widehat{W} = \text{argmin}_{W\in\mathbb{R}^{d\times k}} \sum_{i=0}^{n} \|W^T x_i - y_i\|_2^2 + \lambda\|W\|_F^2$$

We adopt the notation where we have n data points in our training objective and each data point $x_i \in \mathbb{R}^d$. k denotes the number of classes which is in this case equal to 10. Note that $\|W\|_F$ corresponds to the Frobenius norm of W, i.e. $\|\text{vec}(W)\|_2^2$.

Derive a closed form for $\widehat{W}$.

Answer:

$\|W\|_F^2 = Tr(W^TW)$
$\nabla_W \|W\|_F^2 = 2W$

$\widehat{W} = \text{argmin}_{W\in\mathbb{R}^{d\times k}} \sum_{i=0}^{n} \|W^Tx_i - y_i\|_2^2 + \lambda\|W\|_F^2$

Switching to matrix notation:
$\widehat{W} = \text{argmin}_{W\in\mathbb{R}^{d\times k}} \|W^T\mathbf{X} - \mathbf{Y}\|_2^2 + \lambda\|W\|_F^2$
$\nabla_W(\|W^T\mathbf{X} - \mathbf{Y}\|_2^2 + \lambda\|W\|_F^2) = 0$
$2\mathbf{X}^T(\widehat{W}^T\mathbf{X} - \mathbf{Y}) + 2\lambda\widehat{W} = 0$
$\widehat{W} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y}$

b. As as first step we need to choose the vectors $y_i \in \mathbb{R}^k$ by converting the original labels (which are in $\{0,\ldots,9\}$) to vectors. We will use the one-hot encoding of the labels, i.e. the original label $j \in \{0,\ldots,9\}$ is mapped to the standard basis vector e_j. To classify a point x_i we will use the rule $\arg\max_{j=0,\ldots,9} \widehat{W}^Tx_i$.

c. Code up a function called `train` that returns $\widehat{W}$ that takes as input $X \in \mathbb{R}^{n\times d}$, $y \in \{0,1\}^{n\times k}$, and $\lambda > 0$. Code up a function called `predict` that takes as input $W \in \mathbb{R}^{d\times k}$, $X' \in \mathbb{R}^{m\times d}$ and returns an m-length vector with the ith entry equal to $\arg\max_{j=0,\ldots,9} W^Tx_i'$ where x_i' is a column vector representing the ith example from X'.

Train $\widehat{W}$ on the MNIST training data with $\lambda = 10^{-4}$ and make label predictions on the test data. What is the training and testing classification accuracy (they should both be about 85%)?

Answer:

Training Accuracy: 85.195%
Testing Accuracy: 85.34%

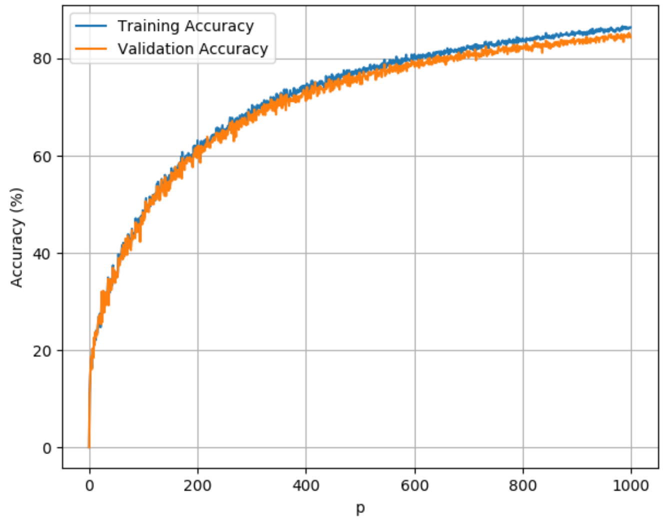
d. We just fit a classifier that was linear in the pixel intensities to the MNIST data. For classification of digits the raw pixel values are very, very bad features: it's pretty hard to separate digits with linear functions in pixel space. The standard solution to the this is to come up with some transform $h : \mathbb{R}^d \to \mathbb{R}^p$ of the original pixel values such that the transformed points are (more easily) linearly separable. In this problem, you'll use the feature transform:

$$h(x) = \cos(Gx + b).$$

where $G \in \mathbb{R}^{p\times d}$, $b \in \mathbb{R}^p$, and the cosine function is applied elementwise. We'll choose G to be a *random* matrix, with each entry sampled i.i.d. with mean $\mu = 0$ and variance $\sigma^2 = 0.1$, and b to be a random vector sampled i.i.d. from the uniform distribution on $[0, 2\pi]$. The big question is: *how do we choose p?* Cross-validation, of course!

Randomly partition your training set into proportions 80/20 to use as a new training set and validation set, respectively. Using the `train` function you wrote above, train a $\widehat{W}^p$ for different values of p and plot the classification training error and validation error on a single plot with p on the x-axis. Be careful, your computer may run out of memory and slow to a crawl if p is too large ($p \leq 6000$ should fit into 4 GB of memory). You can use the same value of λ as above but feel free to study the effect of using different values of λ and σ^2 for fun.

Answer:



e. Instead of reporting just the classification test error, which is an unbiased estimate of the *true* error, we would like to report a *confidence interval* around the test error that contains the true error. For any $\delta \in (0,1)$, it follows from Hoeffding's inequality that if X_i for all $i = 1, \dots, m$ are i.i.d. random variables with $X_i \in [a,b]$ and $\mathbb{E}[X_i] = \mu$, then with probability at least $1 - \delta$

$$\mathbb{P}\left(\left| \left(\frac{1}{m} \sum_{i=1}^{m} X_i \right) - \mu \right| \geq \sqrt{\frac{\log(2/\delta)}{2m}} \right) \leq \delta$$

We will use the above equation to construct a confidence interval around our true classification error since the test error is just the average of indicator variables taking values in 0 or 1 corresponding to the ith test example being classified correctly or not, respectively, where an error happens with probability μ, the *true* classification error.

Let $\widehat{p}$ be the value of p that approximately minimizes the validation error on the plot you just made and use $\widehat{W}^{\widehat{p}}$ to compute the classification test accuracy, which we will denote as E_{test}. Use Hoeffding's inequality, above, to compute a confidence interval that contains $\mathbb{E}[E_{test}]$ (i.e., the *true* error) with probability at least 0.95 (i.e., $\delta = 0.05$). Report E_{test} and the confidence interval.

```
import numpy as np
import matplotlib.pyplot as plt
from mnist import MNIST
import random
import time

from numpy.core.multiarray import ndarray

X_train = []
X_test = []
labels_train = []
labels_test = []


def load_dataset():
    global X_train, X_test, labels_test, labels_train
    mndata = MNIST('./python-mnist/data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
```

```
    X_train = X_train/255.0
    X_test = X_test/255.0


def one_hot(inpt):
    output = np.zeros((inpt.size, 10))
    for i in range(len(inpt)):
        vec = np.zeros(10)
        for j in range(10):
            vec[j] = int(int(inpt[i]) == j)
            output[i] = vec
    return output


def train(X, y, lamb):
    w = np.linalg.solve(np.dot(np.transpose(X), X)+lamb*np.identity(X.shape[1]), np.dot(
    return w


def predict(w, x):
    y=np.dot(np.transpose(w), np.transpose(x))
    return np.argmax(y, axis=0)

def generateTransform(inpt, p, sigma):
    G = sigma*np.random.randn(inpt.shape[1], p)
    b = np.random.uniform(0, 2*np.pi, (p, 1))

    return G, b


def transform(inpt, G, b):
    out = np.transpose(np.cos(np.dot(np.transpose(G), np.transpose(inpt))+b))
    return out


def split(x, y, ylist, frac):
    index = random.sample(range(x.shape[0]), int(x.shape[0]*frac))
    xmajor = x[index]
    xminor = np.delete(x, index, 0)
    ymajor = y[index]
    yminor = np.delete(y, index, 0)
    listmajor = ylist[index]
    listminor = np.delete(ylist, index, 0)

    return xmajor, xminor, ymajor, yminor, listmajor, listminor


load_dataset()

y_train=one_hot(labels_train)
w = train(X_train, y_train, 10**-4)

print("No transformation")
print("Training Accuracy: "+str(sum(predict(w, X_train) == labels_train)/len(X_train)*10
print("Testing Accuracy: "+str(sum(predict(w, X_test) == labels_test)/len(X_test)*100)+'
```

```python
pmax=1*10**3
trainErr = np.zeros((pmax, 1))
valErr = np.zeros((pmax, 1))
for p in range(1, pmax):
    start=time.time()
    G, b= generateTransform(X_train, p, np.sqrt(0.1))
    transX = transform(X_train, G, b)
    trainX, valX, trainY, valY, trainList, valList = split(transX, y_train, labels_train
    w= train(trainX, trainY, 10**-4)
    trainErr[p] = sum(predict(w, trainX) == trainList) / len(trainX)*100
    valErr[p] = sum(predict(w, valX) == valList) / len(valX)*100
    end=time.time()
    print(str(round(p/pmax*100, 3))+"% Done")
    print(str(round((end-start)*(pmax-p),2))+" s left")


pOpt = np.argmax(valErr)
G, b = generateTransform(X_train, p, 0.01)
transX = transform(X_train, G, b)
trainX, valX, trainY, valY, trainList, valList = split(transX, y_train, labels_train, 0
w = train(trainX, trainY, 10**-4)

print("With transformation")
print("Training Accuracy: "+str(sum(predict(w, trainX) == trainList)/len(trainX)*100)+"%
print("Testing Accuracy: "+str(sum(predict(w, transform(X_test, G, b)) == labels_test)/l

plt.plot(trainErr)
plt.plot(valErr)
plt.grid()
plt.ylabel("Accuracy (%)")
plt.xlabel("p")
plt.legend(("Training Accuracy", "Validation Accuracy"))
plt.show()
```