10-701 Machine Learning: Assignment 2

Due on March 11, 2014 at 11:59

Barnabas Poczos, Aarti Singh

Instructions: Failure to follow these directions may result in loss of points.

- Your solutions for this assignment need to be in a pdf format and should be submitted to the blackboard and the webpage `http://barnabas-cmu-10701.appspot.com` for peer-reviewing.
- For the programming question, your code should be well-documented, so a TA can understand what is happening.
- We are NOT going to use Autolab in this assignment.
- DO NOT include any identification (your name, andrew id, or email) in both the content and filename of your submission.

Q1) Support Vector Machines (Pulkit)

SVM [20 points]

In class we discussed support vector classification, but did not cover the regression problem. In this question you have to derive the dual form of SV regression (SVR). Your training data is $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$.
Since the hinge loss that we used in class is only designed for classification we cannot use that for regression. A frequently used loss function for regression is the epsilon sensitive loss:

$$L_\epsilon(x, y, f) = |y - f(x)|_\epsilon = \max(0, |y - f(x)| - \epsilon)$$

Here x is the input, y is the output, and f is the function used for predicting the label.
Using this notation, the SVR cost function is defined as

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} L_\epsilon(x_i, y_i, f),$$

where $f(x) = w^T x$, and $C, \epsilon > 0$ are parameters.

1. Introduce appropriate slack variables, and rewrite this problem as a quadratic problem (i.e. quadratic objective with linear constraints). This form is called the Primal form of support vector regression. (3 points)

$\hat{\boldsymbol{w}} = \texttt{argmin}\ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N} max(0, |y_i - \boldsymbol{w}^T\boldsymbol{x_i}| - \epsilon)$

Let $\eta_i = max(0, |y_i - \boldsymbol{w}^T\boldsymbol{x_i}| - \epsilon)$

$\Rightarrow \eta_i \geq |y_i - \boldsymbol{w}^T\boldsymbol{x_i}| - \epsilon$

$\Rightarrow \eta_i \geq y_i - \boldsymbol{w}^T\boldsymbol{x_i} - \epsilon$ and $\eta_i \geq -y_i + \boldsymbol{w}^T\boldsymbol{x_i} - \epsilon$

Quadratic problem:

$$\hat{w} = \texttt{argmin}\ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}\eta_i \quad \text{s.t.}$$

$$(\boldsymbol{w}^T\boldsymbol{x_i} - y_i) + \eta_i + \epsilon \geq 0$$

$$\eta_i \geq 0$$

$$-(\boldsymbol{w}^T\boldsymbol{x_i} - y_i) + \eta_i + \epsilon \geq 0$$

2. Write down the Lagrangian function for the above primal form.(2 points).

We introduce Lagrange variables $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. Then,

$$L(\boldsymbol{w}, \boldsymbol{\eta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\boldsymbol{\eta}^T 1_n - \sum_{i=1}^{n}\alpha_i\big[(\boldsymbol{w}^T\boldsymbol{x_i} - y_i) + \eta_i + \epsilon\big] - \sum_{i=1}^{n}\beta_i\eta_i - \sum_{i=1}^{n}\gamma_i\big[-(\boldsymbol{w}^T\boldsymbol{x_i} - y_i) + \eta_i + \epsilon\big]$$

3. Using the Karush Kunh Tucker conditions derive the dual form. (5 points).

We need to compute

$$M(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = min_{\boldsymbol{w},\boldsymbol{\eta}} L(\boldsymbol{w},\boldsymbol{\eta},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})$$

First, $\dfrac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w},\boldsymbol{\eta},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = 0$

$\Rightarrow \hat{\boldsymbol{w}} - \sum_{i=1}^{N} \alpha_i \boldsymbol{x_i} + \sum_{i=1}^{N} \gamma_i \boldsymbol{x_i} = 0$

$\Rightarrow \hat{\boldsymbol{w}} = \sum_{i=1}^{N} (\alpha_i - \gamma_i) \boldsymbol{x_i}$

Next, $\dfrac{\partial}{\partial \boldsymbol{\eta}} L(\boldsymbol{w},\boldsymbol{\eta},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = 0$

$\Rightarrow C\boldsymbol{1_n} - \boldsymbol{\alpha} - \boldsymbol{\beta} - \boldsymbol{\gamma} = 0$

$\Rightarrow C\boldsymbol{1_n} = \boldsymbol{\alpha} + \boldsymbol{\beta} + \boldsymbol{\gamma}$

$\Rightarrow \alpha_i \leq C, \;\; \gamma_i \leq C, \;\; \alpha_i + \gamma_i \leq C$

Due to the condition for the derivative w.r.t. $\boldsymbol{\eta}$ to be zerom, the terms with η cancel out. i.e.,

$$M(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{1}{2}||\hat{\boldsymbol{w}}||^2 - \sum_{i=1}^{n} \alpha_i \big[(\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \epsilon\big] - \sum_{i=1}^{n} \gamma_i \big[-(\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \epsilon\big]$$

Substituting the value of $\hat{\boldsymbol{w}}$ into this equation and after some simplification (not shown due to complexity), we get:

$$M(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\gamma})^T \boldsymbol{G}(\boldsymbol{\alpha}-\boldsymbol{\gamma}) + (\boldsymbol{\alpha}-\boldsymbol{\gamma})^T \boldsymbol{Y} - \epsilon(\boldsymbol{\alpha}+\boldsymbol{\gamma})^T \boldsymbol{1_n}$$

where $G_{ij} = \boldsymbol{x_i}^T \boldsymbol{x_j}$ is the Gram matrix, $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are the Lagrange variable vectors, and $\boldsymbol{Y}$ is just $y_1, \ldots, y_n$.
Hence, we get the dual:

$$\begin{aligned}&\texttt{max}\Big[-\tfrac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\gamma})^T \boldsymbol{G}(\boldsymbol{\alpha}-\boldsymbol{\gamma}) + (\boldsymbol{\alpha}-\boldsymbol{\gamma})^T \boldsymbol{Y} - \epsilon(\boldsymbol{\alpha}+\boldsymbol{\gamma})^T \boldsymbol{1_n}\Big]\\ &\boldsymbol{\alpha} \leq C\\ &\boldsymbol{\gamma} \leq C\\ &\boldsymbol{\gamma}+\boldsymbol{\alpha} \leq C\end{aligned}$$

4. Can we use quadratic optimization solvers to solve the dual problem? (1 point).

YES. The optimization function is quadratic in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$, i.e., the maximum degree of any term is 2. So, quadratic programmin can be used.

5. How would you define support vectors in this problem? (2 points).

The KKT conditions say that the product of the constraint and the Lagrange variable is zero, i.e., $\alpha_i((\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \eta_i + \epsilon) = 0$ and
$\gamma_i(-(\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \eta_i + \epsilon) = 0$

The pair (α_i, γ_i) associated with the i^{th} training point is non-zero only if $(\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \eta_i + \epsilon = 0$ or $-(\hat{\boldsymbol{w}}^T \boldsymbol{x_i} - y_i) + \eta_i + \epsilon = 0$. One of these holds if $|y_i - \hat{\boldsymbol{w}}^T| > \epsilon$.
Therefore, the support vectors are those $\boldsymbol{x_i}$ (from the training data) whose predicted value is very different from the actual value.

6. Write down the equation that can be used for predicting the label of an unseen sample X. (2 points)

The predicting function is:

$$f_{\hat{\boldsymbol{w}}}(\boldsymbol{x}) = \hat{\boldsymbol{w}}^T \boldsymbol{x} = \sum_{i=1}^{N} (\alpha_i - \gamma_i) \boldsymbol{x_i}^T \boldsymbol{x}$$

7. Is it possible to kernelize this algorithm? (1 points)

YES. The Gram matrix $\boldsymbol{G}$ will need to change so that $G_{i,j} = K(\boldsymbol{x_i}, \boldsymbol{x_j})$ in the learning algorithm for $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$. In the prediction function, $\boldsymbol{x_i}^T \boldsymbol{x}$ will need to be changed to $K(\boldsymbol{x_i}, \boldsymbol{x_j})$.

8. What happens if we change ϵ? (2 point)

9. What happens if we change C? (2 point)

Q2) Density Estimation (Prashant)

Kernel Density Estimation (10 points)

Let us explore kernel density estimation with a boxcar kernel. Given data, $X_1, \ldots, X_n$, the empirical pdf is defined as follows

$$\hat{p_h}(x) = \frac{1}{n} \frac{\sum_{j=1}^{n} I(|x - X_i| \leq h)}{2h} \tag{1}$$

Here, $|y|$ indicates absolute value and $I(y)$ is an indicator function that is 1 if the boolean argument evaluates to true and 0 otherwise.

1. Draw $\hat{p_h}(x)$ for all $x \in [-7, 7]$ where our data consists of the following observations $\{X_i\} = \{-7, -4, -3, -2, 1, 5\}$ and $h = 3$.

2. Let $F(x) = P(X \leq x)$ represent the true cdf over a random variable X. Show that

$$E(\hat{p_h}(x)) = \frac{F(x+h) - F(x-h)}{2h} \tag{2}$$

3. Determine if our kernel density estimator will be an unbiased estimator in the following scenarios

   1. $pdf(x)$ is uniform between 0 and 1. $\hat{f}(\frac{1}{2})$ is estimated using $h = \frac{1}{3}$
   2. $pdf(x) = 2x$ for $0 \leq x \leq 1$. $\hat{f}(\frac{1}{4})$ is estimated using $h = \frac{1}{5}$
   3. $pdf(x) = \frac{3}{2}x^2$ for $-1 \leq x \leq 1$. $\hat{f}(0)$ is estimated using $h = \frac{1}{5}$

1. Figure goes here

2.

$$\begin{aligned}\mathbb{E}(\hat{p_h}(x)) &= \frac{1}{n}\frac{\sum_{j=1}^{n}\mathbb{E}[I(|x-X_i|\le h)]}{2h}\\ &= \frac{1}{n}\frac{\sum_{j=1}^{n}\mathbb{P}(|x-X_i|\le h)}{2h}\\ &= \frac{1}{n}\frac{\sum_{j=1}^{n}F(x+h)-F(x-h)}{2h}\\ &= \frac{F(x+h)-F(x-h)}{2h}\end{aligned}$$

3. (a) Here, $f(x)=1, F(x)=x$. So, $\hat{f}(1/2)=\mathbb{E}[\hat{p}_{h=1/3}(1/2)]=\frac{F(1/2+1/3)-F(1/2-1/3)}{2*1/3}=\frac{2/3}{2/3}=1=f(1/2)$. So, the kernel density estimator $\hat{p}_{h=1/3}(1/2)$ is an unbiased estimator of $f(1/2)$.

   (b) Here, $f(x)=2x, F(x)=x^2$. So, $\hat{f}(1/4)=\mathbb{E}[\hat{p}_{h=1/5}(1/4)]=\frac{F(1/4+1/5)-F(1/4-1/5)}{2*1/5}=\frac{(1/2)*(2/5)}{2/5}=1/2=f(1/4)$. So, the kernel density estimator $\hat{p}_{h=1/5}(1/4)$ is an unbiased estimator of $f(1/4)$.

   (c) Here, $f(x)=\frac{3}{2}x^2, F(x)=\frac{x^3+1}{2}$. So, $\hat{f}(0)=\mathbb{E}[\hat{p}_{h=1/5}(0)]=\frac{F(0+1/5)-F(0-1/5)}{2*1/5}=\frac{2*(1/5)^3}{2/5}\neq f(0)$. So, the kernel density estimator $\hat{p}_{h=1/5}(0)$ is **not** an unbiased estimator of $f(0)$.

Histograms as MLE(10 points)

Recall that histograms can be used for non-parametric density estimation. Let $X_1,\dots,X_n$ be n data points drawn from a random variable X which takes values between 0 and 1. We further divide the range $[0,1]$ into N different, equal sized, bins, $B_1,\dots,B_N$ each with width h.
Define the empricial histogram p.d.f. is as follows

$$\hat{p_h}=\sum_{i=j}^{N}\frac{\hat{\theta}_j}{h}I(x\in B_j) \tag{3}$$

$$\hat{\theta}_j=\frac{1}{n}\sum_{i=1}^{n}I(X_i\in B_j) \tag{4}$$

Now assume that we model the true pdf of X as a piecewise constant function where the constants are the θ_j which is constant over the bins, B_j for j in $1,\dots,N$. Show that the MLE parameters under this model are identical to the histogram based approach for density estimation.

Here, $\sum_{i=1}^{N} \theta_i * \frac{1}{N} = 1$. So, we can write $\theta_N = N - \sum_{i=1}^{N-1} \theta_i$. Let us call our data $X_1, \ldots, X_n$ by D. Further, let $n_j = \sum_{i=1}^{n} I(X_i \in B_j)$, i.e., n_j is the number of data points in the j^{th} bucket.
Then, the likelihood of our data D is:

$$L(D) = \prod_{i=1}^{N} \theta_i^{n_i}$$

To find θ_k, we differentiate $L(D)$ wrt θ_k and set the derivative to 0.

$$\begin{aligned} \frac{\partial}{\partial \theta_k} L(D) &= \frac{d}{d\theta_k}\Big[\big(\textstyle\prod_{i=1}^{N-1} \theta_i^{n_i}\big) * \theta_N^{n_N}\Big] \\ &= \big(\textstyle\prod_{i=1}^{N-1} \theta_i^{n_i}\big) * n_N * \theta_N^{n_N - 1} * \frac{\partial}{\partial \theta_k}\theta_N + \theta_N^{n_N} * \big(\textstyle\prod_{i=1}^{N-1} \theta_i^{n_i}\big) * \frac{n_k}{\theta_k} \\ &= \big(\textstyle\prod_{i=1}^{N-1} \theta_i^{n_i}\big) * n_N * \theta_N^{n_N - 1} * (-1) + \theta_N^{n_N} * \big(\textstyle\prod_{i=1}^{N-1} \theta_i^{n_i}\big) * \frac{n_k}{\theta_k} = 0 \end{aligned}$$

$\Rightarrow n_N * (-1) + \hat{\theta}_{N,\text{MLE}} * \dfrac{n_k}{\hat{\theta}_{k,\text{MLE}}} = 0$

$\Rightarrow \hat{\theta}_{k,\text{MLE}} = \dfrac{\hat{\theta}_{N,\text{MLE}} * n_k}{n_N}$

$\Rightarrow \hat{\theta}_{N,\text{MLE}} = \dfrac{n_N}{N}$

$\Rightarrow \hat{\theta}_{k,\text{MLE}} = \dfrac{n_k}{N}$

Therefore, the MLE estimate of θ_k is the fraction of data points in the k^{th} bin, which is equal to the empirical histogram pdf, i.e.,

$$\hat{\theta}_{k,MLE} = \hat{\theta}_k$$

Q3) K-Nearest Neighbors (Jit)

Spam Detection [10 points]

One of the earliest and most successful applications of machine learning has been in developing spam detection algorithms. For this problem, you are going to implement K-Nearest Neighbors algorithm on a dataset containing thousands of instances of spam and non-spam emails.

For this problem, you are going to submit your files to the predictive analytics site Kaggle. We have created a private Kaggle competition in which only users with a *cmu.edu* email can access. To access the competition, please go to the following link: https://kaggle.com/join/tyranitar

For this assignment, you will need to write code to complete the following functions:

- distance(email1, email2): Calculate some notion of distance or similarity between two e-mails. As an aside, keep in mind that higher similarities should correspond to smaller distances. (1 points)
- KNNClassify(testData, trainingData, trainingLabels): Classifies each email in the testing dataset using the kNN algorithm. (3 points)

There are many valid distance metrics you can implement, and we encourage you to explore how different distance metrics lead to different results. Furthermore, you are expected to try out different values of k to see which neighborhood leads to the best accuracy on the testing data. Please plot the misclassification rate of at least two different distance metrics and four different values of k. Using your results, comment on the behavior of the kNN algorithm with varying distance metric, and varying values of k. (6 points)

You will be submitting a .csv file to the Kaggle submission system (The format of the CSV file is on the Kaggle website. It needs to contain a header: EmailID, Prediction, and then for each row, it contains a ID number, and a label 0,1 similar to the training data labels file), and upload your code along it. We will be reviewing your code, so please document your code appropriately. For Python, you are allowed to use the NumPy package, but you cannot use machine learning packages that have implementations of kNN. For this assignment, we ask that you use either Python, Julia, or Matlab, so the TAs will be able to review your code without much difficulty. Please upload your plots and your analysis as part of your solutions that you will submit to the peer review site and Blackboard.

Hidely-Ho, K-Nearest Neighborinos [10 points]

Recall, the KNN algorithm on binary-labeled data works as follows

```
Data: Documents $\mathbf{X} = [X_1, X_2, ..., X_m]$, labels $\mathbf{Y} = [Y_1, Y_2, ..., Y_m]; Y_i \in \{-1, 1\}$, integer $k$, and query $x$
Result: label $y$
for i = 1 to m do
    Compute distance $d(x_i, x)$;
end
Compute set $I$ containing indices of the $k$ smallest distances $d(x_i, x)$ ;
Return majority label of $\{y_i$ where $i \in I\}$;
```

Algorithm 1: Algorithm: K-Nearest Neighbors Classification

While this algorithm is effective, computing the distances between all training points and the input query x

make take a long time depending on the size of the training points, the dimensionality of the data, and the distance function.

To simplify this, let's precompute $\mu_+ := \dfrac{1}{n_+} \sum\limits_{y_i=1} x_i$ and $\mu_- := \dfrac{1}{n_-} \sum\limits_{y_i=-1} x_i$, where n_+ is the number of positive examples, and n_- is the number of negative examples.

Using the fact that for vectors $\mathbf{u}, \mathbf{v} \in \mathbf{R}^n$

$$\|\mathbf{u} - \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - 2 < \mathbf{u}, \mathbf{v} >$$

Derive a new classifier $f(x)$ using only μ_+, μ_-, and x that compares each point to the average distance of each class, and express it in the form $f(x) = sgn(\sum\limits_{i=1}^{n} \alpha_i < x_i, x > +b)$ for some α_i and b, such that $f(x) = 1$ when the predicted label of the new point x is 1, and $f(x) = -1$ when the predicted label of the new point x is -1.

We have access to μ_+ and μ_-, which can be thought of as the center-of-mass of the positive and negative labels respectively. For a new x, we assign it the label whose center-of-mass is closer to it. Therefore,

$$f(x) = sgn(\|x - \mu_+\|^2 - \|x - \mu_-\|^2)$$

$$\begin{aligned} f(x) &= sgn(\|x\|^2 + \|\mu_+\|^2 - 2\langle x, \mu_+\rangle - \|x\|^2 - \|\mu_-\|^2 + 2\langle x, \mu_-\rangle) \\ f(x) &= sgn(2\langle x, \mu_- - \mu_+\rangle + \|\mu_+\|^2 - \|\mu_-\|^2) \\ f(x) &= sgn(\langle x, \tfrac{2}{n_-} \textstyle\sum_{y_i=-1} x_i - \tfrac{2}{n_+} \sum_{y_i=1} x_i\rangle + \|\mu_+\|^2 - \|\mu_-\|^2) \end{aligned}$$

So,

$$b = \|\mu_+\|^2 - \|\mu_-\|^2$$

and

$$\alpha_i = \frac{2}{n_-}, y_i = -1$$

$$\alpha_i = \frac{-2}{n_+}, y_i = 1$$

Q4) Kernel Regression (Pengtao)

Kernel Regression (10 points)

Consider local linear regression where the predicted output value of x is $\hat{f}(x) = \hat{\alpha} + \hat{\beta}x$, where

$$\hat{\alpha}, \hat{\beta} = \text{argmin}_{\alpha,\beta} \sum_{i=1}^{n} w_i(x)(y_i - \alpha - \beta x_i)^2 \tag{5}$$

where $w_i(x) = K(\frac{x-x_i}{h}) / \sum_{i=1}^{n} K(\frac{x-x_i}{h})$.

1. Show that the objective function can be re-written as

$$(\boldsymbol{y} - \boldsymbol{Ba})^\top \Omega(x)(\boldsymbol{y} - \boldsymbol{Ba})$$

where $\boldsymbol{y} = [y_1\ y_2\ \dots y_n]^\top$, $\boldsymbol{a} = [\alpha\ \beta]^\top$, $\boldsymbol{B} = [1\ x_1;\ 1\ x_2;\ \dots;\ 1\ x_n]$ and $\Omega(x)$ is a diagonal matrix with diagonal $[w_1(x)\ w_2(x)\ \dots\ w_n(x)]$.

2. Show that $\hat{f}(x)$ is a linear combination of $\{y_i\}_{i=1}^n$, namely $\hat{f}(x)$ can be written as $\hat{f}(x) = \sum_{i=1}^n \ell_i(x) y_i = \boldsymbol{\ell}(x)^\top \boldsymbol{y}$, where $\ell_i(x)$ is some quantity defined over x and $\boldsymbol{\ell}(x) = [\ell_1(x)\ \ell_2(x)\ \dots\ \ell_n(x)]^\top$.

1. We can write:
   $(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a}) = [y_1 - \alpha - \beta x_1, \dots, y_n - \alpha - \beta x_n]^\top$
   $\Rightarrow (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a})^\top \Omega(x) = [(y_1 - \alpha - \beta x_1) w_1(x), \dots, (y_n - \alpha - \beta x_n) w_n(x)]$
   $\Rightarrow (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a})^\top \Omega(x)(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a}) = \sum_{i=1}^n w_i(x)(y_i - \alpha - \beta x_i)^2$

2. We can now minimize the objective function in its matrix form.

$$\frac{\partial}{\partial \boldsymbol{a}} (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a})^\top \Omega(x)(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a}) = 0$$

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{a}} (\boldsymbol{y}^\top \Omega(x) - \boldsymbol{a}^\top \boldsymbol{B}^\top \Omega(x))(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{a}) = 0$$

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{a}} (\boldsymbol{y}^\top \Omega(x)\boldsymbol{y} - \boldsymbol{y}^\top \Omega(x)\boldsymbol{B}\boldsymbol{a} - \boldsymbol{a}^\top \boldsymbol{B}^\top \Omega(x)\boldsymbol{y} + \boldsymbol{a}^\top \boldsymbol{B}^\top \Omega(x)\boldsymbol{B}\boldsymbol{a}) = 0$$

$$\Rightarrow \boldsymbol{B}^\top \Omega(x)\boldsymbol{y} = \boldsymbol{B}^\top \Omega(x)\boldsymbol{B}\boldsymbol{a}$$

$$\Rightarrow \boldsymbol{a} = (\boldsymbol{B}^\top \Omega(x)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \Omega(x)\boldsymbol{y}$$

This equation says that $\boldsymbol{a}$ can be obtained by multiplying each y_i by a function of x_i and x (this holds because $(\boldsymbol{B}^\top \Omega(x)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \Omega(x)$ depends only on x_i and x). We can write the action of the matrix $(\boldsymbol{B}^\top \Omega(x)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \Omega(x)$ in terms of functions $f_i(x)$ and $g_i(x)$.

$$\boldsymbol{a} = \begin{bmatrix} \sum_{i=1}^n f_i(x) y_i \\ \sum_{i=1}^n g_i(x) y_i \end{bmatrix}$$

Next, $\hat{f}(x) = [1, x]\boldsymbol{a} = \sum_{i=1}^n f_i(x) y_i + \sum_{i=1}^n x * g_i(x) y_i$. Therefore,

$$l_i(x) = f_i(x) + x * g_i(x).$$

Kernelized Ridge Regression (10 points)

The nonparametric kernel regression does a local fit around the test point. Lets now investigate the use of kernels for regression in another way. Similar to the kernel trick for SVMs, we can apply the kernel trick in regression as follows. (However, note that this produces a global fit to the data.)
Consider the ridge regression problem where we have a set of data points $\{\mathbf{x}_i\}_{i=1}^N$ and corresponding response values $\{y_i\}_{i=1}^N$. To achieve better performance, we use a feature mapping function Φ to map the original d-dimensional feature vector $\mathbf{x}$ to a new D-dimensional feature vector $\hat{\mathbf{x}} = \Phi(\mathbf{x})$, where $D \gg d$. Let $\boldsymbol{\Phi} \in \mathbb{R}^{N \times D}$ denote the design matrix, whose ith row contains the new feature vector $\hat{\mathbf{x}}_i^\intercal$ of the ith data point. Let $\mathbf{y}$ denote the response value vector whose ith component is the response value y_i of the ith point.
For ridge regression, the objective function is $J(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}||^2 + \frac{\lambda}{2}||\boldsymbol{\beta}||^2$, where λ is the tradeoff parameter. Let $\boldsymbol{\beta}^*$ denote the solution to the ridge regression. In the following steps, we are going to derive the

kernel ridge regression.

(a) First, show that $\boldsymbol{\beta}^*$ is in the space spanned by rows in $\boldsymbol{\Phi}$, i.e., $\boldsymbol{\beta}^*$ can be written in the form $\boldsymbol{\beta}^* = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}^*$, where $\boldsymbol{\alpha}^* \in \mathbb{R}^{N\times 1}$. (Hint: $\boldsymbol{\beta}$ can be decomposed into $\boldsymbol{\beta} = \boldsymbol{\beta}_{||} + \boldsymbol{\beta}_{\perp}$, where $\boldsymbol{\beta}_{\perp}$ is orthogonal to the span of rows in $\boldsymbol{\Phi}$ and $\boldsymbol{\beta}_{||}$ lies in the span of rows in $\boldsymbol{\Phi}$.)

We write $\boldsymbol{\beta} = \boldsymbol{\beta}_{\perp} + \boldsymbol{\beta}_{||}$. So,

$$\begin{aligned} \Rightarrow \quad J(\boldsymbol{\beta}) \quad &= \quad \tfrac{1}{2}||\mathbf{y} - \boldsymbol{\Phi}(\boldsymbol{\beta}_{||} + \boldsymbol{\beta}_{\perp})||^2 + \tfrac{\lambda}{2}||\boldsymbol{\beta}_{||} + \boldsymbol{\beta}_{\perp}||^2 \\ \boldsymbol{\Phi}\boldsymbol{\beta}_{\perp} = 0 & \\ \Rightarrow \quad J(\boldsymbol{\beta}) \quad &= \quad \tfrac{1}{2}||\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}_{||}||^2 + \tfrac{\lambda}{2}\left(||\boldsymbol{\beta}_{||}||^2 + ||\boldsymbol{\beta}_{\perp}||^2\right) \end{aligned}$$

We can minimize $J(\boldsymbol{\beta})$ over $\boldsymbol{\beta}_{||}$ independently of $\boldsymbol{\beta}_{\perp}$. This minimum is realized with $\boldsymbol{\beta}_{\perp} = 0$. So, the component of $\boldsymbol{\beta}$ perpendicular to $\boldsymbol{\Phi}$ is 0. Therefore, $\boldsymbol{\beta}$ is in the column space of $\boldsymbol{\Phi}$.

1. *Victory was near.*

(b) In (a), we have proved that $\boldsymbol{\beta}^* = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}^*$. In this step, show that $\boldsymbol{\alpha}^* = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \mathbf{I})^{-1}\mathbf{y}$.

From our analysis of linear regression, we know that:

$$\begin{aligned} &\boldsymbol{\beta}^* = (\lambda \mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \mathbf{y} \\ &\Rightarrow (\lambda \mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})\beta^* = \boldsymbol{\Phi}^\top \mathbf{y} \\ &\Rightarrow \lambda\boldsymbol{\beta}^* = \boldsymbol{\Phi}^\top(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}^*) \\ &\Rightarrow \boldsymbol{\beta}^* = \boldsymbol{\Phi}^\top \frac{1}{\lambda}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}^*) \\ &\Rightarrow \boldsymbol{\beta}^* = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}^* \\ &\Rightarrow \boldsymbol{\alpha}^* = \frac{1}{\lambda}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}^*) \\ &\Rightarrow \boldsymbol{\alpha}^* = \frac{1}{\lambda}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\alpha}^*) \\ &\Rightarrow (\lambda\boldsymbol{\alpha}^* + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\alpha}^*) = \mathbf{y} \\ &\Rightarrow \boldsymbol{\alpha}^* = (\lambda \mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\mathbf{y} \end{aligned}$$

1. *but the power of the ring could not be undone...*

(c) In practice, it is very hard to design the feature mapping function Φ. Even if we can design it, computing the inner product $\Phi(\mathbf{x})^\top\Phi(\mathbf{x}')$ between two points can be costly. Instead, we can use a kernel function $k(\mathbf{x}, \mathbf{x}')$ to implicitly compute the inner product of high dimensional features, i.e., $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top\Phi(\mathbf{x}')$. Basically, kernelization means using the kernel function to replace inner products. In this step, given the kernel function $k(\mathbf{x}, \mathbf{x}')$, where x and x' are the original feature vectors of dimension d, try to kernelize the ridge regression. You need to consider both training and testing. (Hint: In training phase, you need to replace all inner produces in $\boldsymbol{\alpha}^*$ with kernel function. In testing phase, giving a new test point $\mathbf{x}$, you need to compute $\Phi(\mathbf{x})^\intercal \boldsymbol{\beta}^*$. Replace all inner products in $\Phi(\mathbf{x})^\intercal \boldsymbol{\beta}^*$ with kernel function.)

During training To compute $\boldsymbol{\alpha}^\intercal$ we need to compute $G = \boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal$. $G_{i,j} = \boldsymbol{\Phi}(\mathbf{x_i})^\intercal \boldsymbol{\Phi}(\mathbf{x_j}) = K(x_i, x_j)$. Hence, training can be done using kernel functions.

During testing: The value assigned to a new point x by the regression function is $\boldsymbol{\Phi}(x)^\intercal \boldsymbol{\beta}^* = \boldsymbol{\Phi}(x)^\intercal(\boldsymbol{\Phi}^\intercal \boldsymbol{\alpha}^*) = (\boldsymbol{\Phi}(x)^\intercal \boldsymbol{\Phi}^\intercal)\boldsymbol{\alpha}^*$.
$(\boldsymbol{\Phi}(x)^\intercal \boldsymbol{\Phi}^\intercal)$ is a $1 \times n$ matrix (where n is the number of training examples). It's i^{th} element is $\boldsymbol{\Phi}(x)^\intercal \boldsymbol{\Phi}(x_i) = K(x, x_i)$. Therefore, the regression function can be computed using kernel functions.

Q5) Model Selection (Dani)

In this homework, you will perform model selection on a sentiment analysis dataset of music reviews [1]. The dataset consists of reviews from Amazon.com for musics. The ratings have been converted to a binary label, indicating a negative review or a positive review. We will use lasso logistic regression [2] for this problem. The lasso logistic regression objective function to minimize during training is:

$$\mathcal{L}(\boldsymbol{\beta}) = \log(1 + \exp(-y\boldsymbol{\beta}^\top \boldsymbol{x})) + \lambda \|\boldsymbol{\beta}\|_1$$

In lasso logistic regression, we penalize the loss function by an L_1 norm of the feature coefficients. As you have learned in class, penalization with an L_1 norm tends to produce solutions where some coefficients are exactly 0. This makes it attractive for high-dimensional data such as text, because in most cases most words can typically be ignored. Furthermore, since we are often left with only a few nonzero coefficients, the lasso solution is often easy to interpret. The goal of model selection here is to choose λ since each setting of λ implies a different model size (number of non-zero coefficients).
You do not need to implement lasso logistic regression. You can download an implementation from `https://github.com/redpony/creg`, and the dataset can be found here: `http://www.cs.cmu.edu/~dyogatam/10701/`. There are three feature files and three response (label) files (all response files end with `.res`). They are already in the format required by the implementation you will use. The files are:

- Training data: `music.train` and `music.train.res`
- Development data: `music.dev` and `music.dev.res`
- Test data: `music.test` and `music.test.res`

Important note: the code outputs **accuracy**, whereas you need to plot **classification error** here. You can simply transform accuracy to error by using $1 -$ accuracy.

Error on development (validation) data [8 points]

In the first part of the problem, we will use error on a development data to choose λ. Run the model with $\lambda = \{10^{-8}, 10^{-7}, 10^{-6}, \ldots, 10^{-1}, 1, 10, 100\}$.

1. Plot the error on training data and development data as a function of $\log \lambda$.
2. Plot the model size (number of nonzero coefficients) on development data as a function of $\log \lambda$.
3. Choose λ that gives the lowest error on development data. Run it on the test data and report the test error.

Briefly discuss all the results.

Model Complexity and Bias-Variance Tradeoff [3 points]

Give a high-level explanation on the relation between λ and the bias and variance of parameter estimates $\hat{\boldsymbol{\beta}}$. Does larger λ correspond to higher or lower bias? What about the variance? Does larger λ lead to a more complex or a less complex model?

Resolving a tie [2 points]

If there are more than one λ that minmizes the error on the development data, which one will you pick? Explain your choice.

Random search [5 points]

An alternative way to search λ is by randomly sampling its value from an interval.

1. Sample eleven random values log uniformly from an interval $[10^{-8}, 100]$ for λ and train a lasso logistic regression model. Plot the error on development data as a function of $\log \lambda$.
2. Choose λ that gives the lowest error on development data. Run it on the test data and report the test error.

Random vs. grid search [2 points]

Which one do you think is a better method for searching values to try for λ? Why?

References

[1] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In Proc. of ACL, 1997.

[2] Robert Tibshirani. Regression shrinkage and selection via the lasso, Journal of Royal Statistical Society B, 58(1):267:288, 1996.