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)
2. Write down the Lagrangian function for the above primal form.(2 points)
3. Using the Karush Kunh Tucker conditions derive the dual form. (5 points)
4. Can we use quadratic optimization solvers to solve the dual problem? (1 point)

5. How would you define support vectors in this problem? (2 points)
6. Write down the equation that can be used for predicting the label of an unseen sample X. (2 points)
7. Is it possible to kernelize this algorithm? (1 points)
8. What happens if we change ϵ? (2 point)
9. What happens if we change C? (2 point)

1. In the above cost function, ϵ defines the region inside which errors are ignored. The loss function defined above is non-differentiable due to the absolute value in the loss function. We can introduce slack variables ξ and ξ^* to account for errors in points that lie outside the ϵ tube as follows. (These are similar to the slack variables used in classification.)

$$y_i - \langle w, x_i \rangle - \epsilon \leq \xi_i \tag{1}$$
$$\langle w, x_i \rangle - y_i - \epsilon \leq \xi_i^* \tag{2}$$
$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, n \tag{3}$$

Thus, we can rewrite the primal form as,

$$\min_{w \in \mathbb{R}^m, \xi \in \mathbb{R}^n, \xi^* \in \mathbb{R}^n} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

s.t. equations (1)-(3) are satisfied.

Rubric: 2 points for constraints. 1 point for the objective. Partial grade if there is a mistake using one of the slack variables etc.

2. Having the above constraints and objective, the Lagrangian function can be written as follows.

$$\begin{aligned} L = L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) := \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ & - \sum_{i=1}^{n}(\beta_i \xi_i + \beta_i^* \xi_i^*) \\ & - \sum_{i=1}^{n} \alpha_i(\epsilon + \xi_i - y_i + \langle w, x_i \rangle) \\ & - \sum_{i=1}^{n} \alpha_i^*(\epsilon + \xi_i^* + y_i - \langle w, x_i \rangle), \end{aligned} \tag{4}$$

where the Lagrange multipliers have to satisfy the positivity constraints,

$$\alpha_i, \alpha_i^*, \beta_i, \beta_i^* \geq 0, \quad (i = 1, \ldots, n).$$

Rubric: 2 points for the Lagrangian. Subtract 1 if the constraints are missing. Partial grade if minor typo in equation.

3. We need to solve the following min-max problem:

$$(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) = \min_{w,\xi,\xi^*} \max_{\alpha,\alpha^*,\beta,\beta^*} L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) \tag{5}$$

$$= \max_{\alpha,\alpha^*,\beta,\beta^*} \min_{w,\xi,\xi^*} L(w, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*) \tag{6}$$

[Similarly as we discussed in class for classification, the max and min can be switched because the so-called strong duality holds for quadratic problems.]

Taking the derivative of L w.r.t the primal variables (w, ξ_i and ξ_i^*), we get

$$\partial_w L = w - \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i = 0$$

$$\partial_{\xi_i} L = C - \alpha_i - \beta_i = 0$$

$$\partial_{\xi_i^*} L = C - \alpha_i^* - \beta_i^* = 0$$

From the last two equations we have that

$$0 \leq \beta_i = C - \alpha_i$$

$$0 \leq \beta_i^* = C - \alpha_i^*$$

Substituting the results back into the Lagrangian (4), we get

$$\begin{aligned}
L = \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\beta_i\xi_i + \beta_i^*\xi_i^*) - \sum_{i=1}^{n}\alpha_i(\epsilon + \xi_i - y_i + \langle w, x_i\rangle) - \sum_{i=1}^{n}\alpha_i^*(\epsilon + \xi_i^* + y_i - \langle w, x_i\rangle) \\
= & \frac{1}{2}\|\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i\|^2 + \sum_{i=1}^{n}\xi_i \underbrace{(C - \beta_i - \alpha_i)}_{0} + \sum_{i=1}^{n}\xi_i^* \underbrace{(C - \beta_i^* - \alpha_i^*)}_{0} - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \\
& + \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) \underbrace{\langle w, x_i\rangle}_{\langle \sum_{j=1}^{n}(\alpha_j - \alpha_j^*)x_j, x_i\rangle} \\
= & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*)
\end{aligned}$$

Therefore, the dual problem is

$$\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \tag{7}$$

$$s.t. \quad \alpha_i, \alpha_i^* \in [0, C] \tag{8}$$

[Note that if you use $\langle w, x\rangle + b$ instead of $\langle w, x\rangle$, then you have an extra constraint: $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$.]

Rubric: 2 points for the derivatives of L, 1 point for (7), 1 point for (8), 1 point for explaining the details well. Partial grade for minor mistakes in derivation.

4. The problem has a quadratic objective with linear constraints, therefore it can be solved by a Quadratic Programming solver.

   Rubric: 1 point for correct answer. No partial credit.

5. The KKT complementary slackness conditions are as follows. In the optimal solutions of (5) and (6) we have that

$$\alpha_i(\epsilon + \xi_i - y_i + \langle w, x_i \rangle) = 0 \tag{9}$$
$$\alpha_i^*(\epsilon + \xi_i^* + y_i - \langle w, x_i \rangle) = 0 \tag{10}$$
$$\beta_i \xi_i = 0 \tag{11}$$
$$\beta_i^* \xi_i^* = 0 \tag{12}$$

   for all $i = 1, \ldots, n$.

   Equation (9) implies that if $\alpha_i > 0$, then $(\epsilon + \xi_i - y_i + \langle w, x_i \rangle) = 0$.

   Now, if $\xi_i = 0$, then it implies that x_i is on the border of the ϵ-tube, therefore x_i is a margin support vector. If $\xi_i > 0$, then it means we are outside of the ϵ-tube. These x_i vectors are the non-margin support vectors. Similar reasoning holds for ξ_i^* and α_i^*.

   Rubric: 1 point for margin support vectors. 1 point for non-margin support vectors.

6. Since for prediction we use $f(x) = \langle w, x \rangle$, and $w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i$, therefore

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle$$

   Rubric: 1 point for the correct prediction, 1 point for reasoning.

7. Yes, we can write the above equation in the kernel form.

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) k(x_i, x)$$

   Rubric: 1 point for the correct answer.

8. ϵ plays the opposite role of C. The smaller the value of ϵ, the harder SVM tries to fit smaller errors around the learnt SVM function, and leads to a more complex model. Smaller ϵ also leads to a less sparse solution (more support vectors).
   Small ϵ - More complex model. Low Bias, High variance.
   Large ϵ - Less complex model. High Bias, Low Variance.

   Rubric: 1 point for reasoning. 1 point for mentioning the relationship with Bias/Variance

9. C plays a similar role as it did during classification. It is a measure of how strongly we penalize errors. It should be tuned for bias vs variance with model selection. The higher the value of C, the larger the tendency of SVM to penalize errors and overfit the data. The lower the value of C, the larger its tendency to ignore errors and underfit the data.
   Large C - More complex model. Low Bias, High variance.
   Small C - Less complex model. High Bias, Low Variance.

   Rubric: 1 point for reasoning. 1 point for mentioning the relationship with Bias/Variance

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

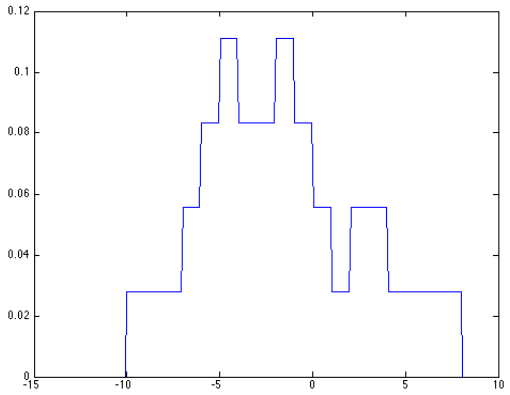
$$\hat{p_h}(x) = \frac{1}{n} \frac{\sum_{j=1}^{n} I(|x - X_i| \leq h)}{2h} \tag{13}$$

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

$$E(\hat{p_h}(x)) = \frac{F(x + h) - F(x - h)}{2h} \tag{14}$$

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



1)
Award 3 points for a correct plot. Award 2 points if the shape of the plot is similar.
2)

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

Here note that $E(I(|x - X_j|)$ is the same for all j. Therefore I can use just X instead of X_j

$$= \frac{1}{2h} E(I(|x - X| \leq h))$$
$$= \frac{1}{2h} P(|x - X| \leq h)$$
$$= \frac{1}{2h} P(-h \leq x - X \leq h)$$
$$= \frac{F(x+h) - F(x-h)}{2h}$$

Award 4 points for this proof, partial points if they get some steps right.
3) Note that $pdf(x) = \frac{dF(x)}{dx}$. From part two, we see that the expected pdf that we get from the box car kernel is just a linear approximation of the gradient of $F(x)$. So intuitively, for cases 1 and 2 below, the estimator will be unbiased since linear approximation of a constant or linear pdf is going to be accurate, but for case 3, the linear approximation of a quadratic pdf is not going to be without bias. Formally:

1. For a uniform pdf, $F(x) = 0$ for $x < 0$, $F(x) = x$ for $0 \leq x \leq 1$ and $F(x) = 1$ for $x > 1$. From part two, bias of the estimator is $F(\frac{5}{6}) - F(\frac{1}{6})/(\frac{2}{3}) = 1 = pdf(\frac{1}{2})$. True

2. For this case, $F(x) = 0$ for $x < 0$, $F(x) = x^2$ for $0 \leq x \leq 1$ and $F(x) = 1$ for $x > 1$. From part two, bias of the estimator is $F(\frac{9}{20}) - F(\frac{1}{20})/(\frac{2}{5}) = \frac{1}{2} = pdf(1/4)$. True.

3. For this case, $F(x) = 0$ for $x < -1$, $F(x) = \frac{x^3+1}{2}$ for $-1 \leq x \leq 1$ and $F(x) = 1$ for $x > 1$. From part two, bias of the estimator is $F(\frac{1}{5}) - F(\frac{-1}{5})/(\frac{2}{5}) \neq 0 = pdf(0)$. False.

Award a point for each part. Intuitive (or formal) explanation is good enough for full credit.

Histograms as MLE(10 points)

Recall that histograms can be used for non-parametric density estimation. Let $X_1, \ldots, X_n$ 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, $B_1, \ldots, B_N$ each with width h.
Define the empricial histogram p.d.f. is as follows

$$\hat{p_h} = \sum_{j=1}^{N} \frac{\hat{\theta_j}}{h} I(x \in B_j) \tag{15}$$

$$\hat{\theta_j} = \frac{1}{n} \sum_{i=1}^{n} I(X_i \in B_j) \tag{16}$$

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 $1, \ldots, N$. Show that the MLE parameters under this model are identical to the histogram based approach for density estimation.

Recall that if our distribution has multiple parameters, we do MLE by computing the log likelihood, taking the partial with respect to each parameter, and setting each partial to 0.
For this case, note that $h\sum_j \theta_j = 1$ since the area under the pdf must be 1. Define $a_j = \sum_i I(x_i \in B_j)$.

$$L(D) = \prod_j \theta_j^{a_j}$$
$$LL(D) = \sum_j a_j \log(\theta_j)$$
$$LL(D) = \sum_{j=1}^{N-1} a_j \log(\theta_j) + (n - \sum_{j=1}^{N-1}(a_j)) \log(\frac{1}{h} - \sum_{j=1}^{N-1} \theta_j)$$

The last step follows because $a_N = n - \sum_j^{N-1} a_j$ and because $\sum_j h\theta_j = 1$ implies $\theta_N = \frac{1}{h} - \sum_{j=1}^{N-1} \theta_j$. Note that we can substitute for θ_N without loss of generality. We could just as easily do this for some other parameter.

$$\frac{\partial LL(D)}{\partial \theta_j} = \frac{a_j}{\theta_j} - \frac{n - \sum_j^{N-1} a_j}{\frac{1}{h} - \sum_{j=1}^{N-1} \theta_j} = 0$$
$$\frac{a_j}{h} - a_j \sum_{j=1}^{N-1} \theta_j - n\theta_j + \theta_j \sum_{j=1}^{N-1} a_j = 0$$

Again, use the fact that $\theta_N = \frac{1}{h} - \sum_j^{N-1} \theta_j$ and $a_N = n - \sum_j^{N-1} a_j$.

$$\frac{a_j}{h} - a_j(\frac{1}{h} - \theta_N) - n\theta_j + (n - a_N)\theta_j = 0$$
$$a_j\theta_N - a_N\theta_j = 0$$

We get such an equation for each $j = 1, \ldots, N-1$. We can then sum all of these to get

$$(n - a_N)\theta_N - a_N(\frac{1}{h} - \theta_N) = 0$$
$$\theta_N = \frac{a_N}{nh} = \hat{p}_h(x)$$

for $x \in B_N$, which is what we want. Without loss of generality, this holds true not only for θ_N but for all θ_j.

There are other ways to prove this. Award 4 points for the correct set up, i.e. award 4 points to anyone who writes out the log likelihood correctly and then correctly takes the partial with respect to each θ_j to try and solve the problem. Award a further 6 points to anyone who solves the system of equations correctly.

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: EmaiID, 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 $\mathbf{X} = [x_1, x_2, ..., x_n]$, labels $\mathbf{Y} = [y_1, y_2, ..., y_n]; Y_i \in \{-1, 1\}$, 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 $\mu_+ := \frac{1}{n_+} \sum_{y_i=1} x_i$ and $\mu_- := \frac{1}{n_-} \sum_{y_i=-1} x_i$, 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 $f(x) = sgn(\sum_{i=1}^{n} \alpha_i < x_i, x > +b)$ 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.

Let $f(x) = sgn(\|x - \mu_-\|^2 - \|x - \mu_+\|^2)$ (**2 point**; Give 1 if they don't square the terms)

$= sgn(\|x\|^2 + \|\mu_-\|^2 - 2 < x, \mu_- > - \|x\|^2 - \|\mu_+\|^2 + 2 < x, \mu_+ >)$
$= sgn(2 < \mu_+ - \mu_-, x > + \|\mu_-\|^2 - \|\mu_+\|^2)$ (**3 points** for reducing to this form)

$\|\mu_-\|^2 =< \mu_-, \mu_- >= \frac{1}{n_-^2} \sum_{y_i=y_j=-1} < x_i, x_j >,$

$\|\mu_+\|^2 =< \mu_+, \mu_+ >= \frac{1}{n_+^2} \sum_{y_i=y_j=1} < x_i, x_j >,$

and $< \mu_+ - \mu_-, x >= \frac{1}{n_+} \sum_{y_i=1} < x_i, x > \; - \frac{1}{n_-} \sum_{y_i=-1} < x_i, x >= \sum_{i=1}^{n} (\frac{y_i + 1}{2n_+} + \frac{y_1 - 1}{2n_-}) < x_i, x >$

Let $\alpha_i = (\frac{y_i + 1}{n_+} + \frac{y_1 - 1}{n_-})$ and $b = \|\mu_-\|^2 - \|\mu_+\|^2 = \frac{1}{n_-^2} \sum_{y_i=y_j=-1} < x_i, x_j > \; - \frac{1}{n_+^2} \sum_{y_i=y_j=1} < x_i, x_j >,$

then $f(x) = sgn(\|x - \mu_-\|^2 - \|x - \mu_+\|^2) = sgn(\sum_{i=1}^{n} \alpha_i < x_i, x > +b)$

(Give **3 points** for a correct derivation of α_i; If they use indicator functions something similar, e.g. n_{y_i}, then take off 1.5 points. Give **2 points** for a correct derivation of b. Both $b = \|\mu_-\|^2 - \|\mu_+\|^2$ and $b = \frac{1}{n_-^2} \sum_{y_i=y_j=-1} < x_i, x_j > \; - \frac{1}{n_+^2} \sum_{y_i=y_j=1} < x_i, x_j >$ are acceptable.)

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

$$\hat{\alpha}, \hat{\beta} = \text{argmin}_{\alpha,\beta} \sum_{i=1}^{n} w_i(x)(y_i - \alpha - \beta x_i)^2 \tag{17}$$

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

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

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 $[w_1(x)\ w_2(x)\ \dots\ w_n(x)]$.

(4 pts)

$$\begin{aligned}
&\sum_{i=1}^{n} w_i(x)(y_i - \alpha - \beta x_i)^2 \\
&= \sum_{i=1}^{n} w_i(x)(y_i - [1\ x_i]\mathbf{a})^\mathsf{T}(y_i - [1\ x_i]\mathbf{a}) \quad (\text{where } \mathbf{a} = [\alpha\ \beta]^\mathsf{T}) \\
&= \sum_{i=1}^{n} w_i(x)(\mathbf{y}_i - \mathbf{B}_i\mathbf{a})^\mathsf{T}(\mathbf{y}_i - \mathbf{B}_i\mathbf{a}) \quad (\text{where } \mathbf{y} = [y_1\ y_2\ \dots y_n]^\top, \mathbf{B} = [1\ x_1;\ 1\ x_2;\ \dots;\ 1\ x_n], \\
&\qquad\qquad \mathbf{y}_i \text{ is the } i\text{th row of } \mathbf{y}, \mathbf{B}_i \text{ is the } i\text{th row of } \mathbf{B}) \\
&= (\mathbf{y} - \mathbf{B}\mathbf{a})^\top \Omega(x)(\mathbf{y} - \mathbf{B}\mathbf{a}) \quad (\text{where } \Omega(x) \text{ is a diagonal matrix with diagonal } [w_1(x)\ w_2(x)\ \dots\ w_n(x)])
\end{aligned} \tag{18}$$

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 $\boldsymbol{\ell}(x) = [\ell_1(x)\ \ell_2(x)\ \dots\ \ell_n(x)]^\top$.

(6 pts)

First, we solve $\hat{\mathbf{a}} = \text{argmin}_{\mathbf{a}} \quad (\mathbf{y} - \mathbf{B}\mathbf{a})^\top \Omega(x)(\mathbf{y} - \mathbf{B}\mathbf{a})$. Taking the derivative of $(\mathbf{y} - \mathbf{B}\mathbf{a})^\top \Omega(x)(\mathbf{y} - \mathbf{B}\mathbf{a})$ w.r.t $\mathbf{a}$ and setting the derivative to zero, we get

$$-\mathbf{B}^\mathsf{T}\Omega(x)(\mathbf{y} - \mathbf{B}\mathbf{a}) = 0 \tag{19}$$

Solving for $\mathbf{a}$, we have

$$\hat{\mathbf{a}} = (\mathbf{B}^\mathsf{T}\Omega(x)\mathbf{B})^{-1}\mathbf{B}^\mathsf{T}\Omega(x)\mathbf{y} \tag{20}$$

$$\begin{aligned}
\hat{f}(x) &= b^\mathsf{T}(x)\hat{\mathbf{a}} \\
&= b^\mathsf{T}(x)(\mathbf{B}^\mathsf{T}\Omega(x)\mathbf{B})^{-1}\mathbf{B}^\mathsf{T}\Omega(x)\mathbf{y} \\
&= \boldsymbol{\ell}(x)^\mathsf{T}\mathbf{y}
\end{aligned} \tag{21}$$

where $b^\mathsf{T}(x) = [1 \quad x]$ and $\boldsymbol{\ell}(x)^\mathsf{T} = b^\mathsf{T}(x)(\mathbf{B}^\mathsf{T}\Omega(x)\mathbf{B})^{-1}\mathbf{B}^\mathsf{T}\Omega(x)$

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 $\hat{\mathbf{x}}_i^\mathsf{T}$ 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 $\boldsymbol{\beta}^* = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\alpha}^*$, where $\boldsymbol{\alpha}^* \in \mathbb{R}^{N \times 1}$. (Hint: $\boldsymbol{\beta}$ can be decomposed into $\boldsymbol{\beta} = \boldsymbol{\beta}_{\parallel} + \boldsymbol{\beta}_{\perp}$, where $\boldsymbol{\beta}_{\perp}$ is orthogonal to the span of rows in $\boldsymbol{\Phi}$ and $\boldsymbol{\beta}_{\parallel}$ lies in the span of rows in $\boldsymbol{\Phi}$.)

(4 pts)

$$\begin{aligned} J(\boldsymbol{\beta}) &= J(\boldsymbol{\beta}_{\parallel} + \boldsymbol{\beta}_{\perp}) \\ &= \tfrac{1}{2}||\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}_{\parallel} - \boldsymbol{\Phi}\boldsymbol{\beta}_{\perp}||^2 + \lambda||\boldsymbol{\beta}_{\perp}||^2 + \lambda||\boldsymbol{\beta}_{\parallel}||^2 \\ &= \tfrac{1}{2}||\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}_{\parallel}||^2 + \lambda||\boldsymbol{\beta}_{\perp}||^2 + \lambda||\boldsymbol{\beta}_{\parallel}||^2 \end{aligned} \tag{22}$$

To minimize this objective, $\boldsymbol{\beta}_{\perp}$ needs to be $\mathbf{0}$. Thereby, $\boldsymbol{\beta}^*$ lies in the span of rows in $\boldsymbol{\Phi}$.

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

(4 pts)

Plug in $\boldsymbol{\beta} = \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\alpha}$ into $J(\boldsymbol{\beta})$

$$\begin{aligned} &\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \\ &= \|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 + \lambda\boldsymbol{\alpha}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} \end{aligned} \tag{23}$$

Taking derive of Eq.(23) w.r.t $\boldsymbol{\alpha}$ and setting it to zero, we get

$$-2\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) + 2\lambda\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = 0 \tag{24}$$

Solving for $\boldsymbol{\alpha}$, we get

$$\boldsymbol{\alpha} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda I)^{-1}\mathbf{y} \tag{25}$$

(c) In practice, it is very hard to design the feature mapping function ϕ. Even if we can design it, computing the inner product $\phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{x}')$ 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., $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{x}')$. 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 $\phi(\mathbf{x})^{\mathsf{T}}\boldsymbol{\beta}^*$. Replace all inner products in $\phi(\mathbf{x})^{\mathsf{T}}\boldsymbol{\beta}^*$ with kernel function.)

(2 pts)

In the training phase, we need to compute $\boldsymbol{\alpha}$. Let $\mathbf{K}$ denote a matrix where $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$. It can be easily checked that $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}}$. So, $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$. In the testing phase, we need to compute $\phi(\mathbf{x})^{\mathsf{T}}\boldsymbol{\beta}^* = \phi(\mathbf{x})^{\mathsf{T}}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\alpha} = \sum_{i=1}^{N}\alpha_i k(x_i, x)$

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 [3] 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:

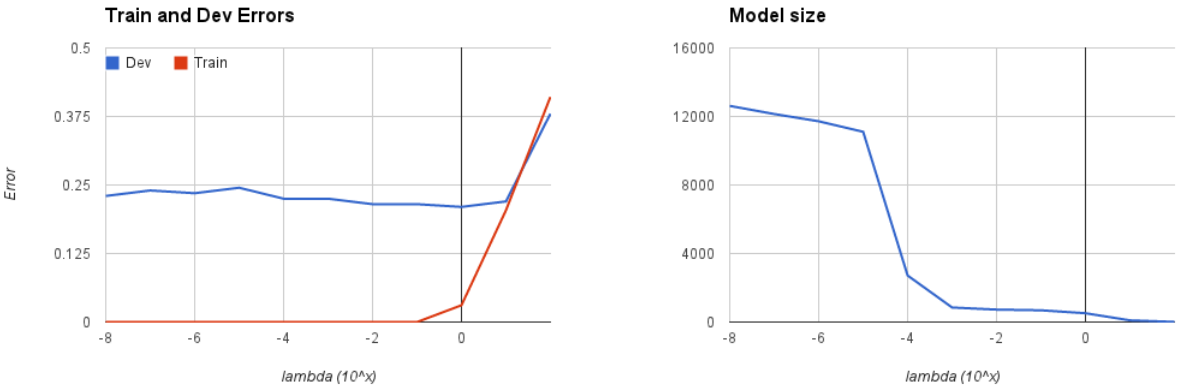


Figure 1: Train and dev errors and model size.

- 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.

See Figure 1. Test set error: 0.24 ($\lambda = 1$). In general, for typical datasets, you should see a "U" shape error curve for the development data (i.e., the error first goes down when you increase lambda then it goes back up again), while for training data the error keeps going up as you increase λ. In this case, the "U" shape is less visible because of the scale in my plot, but we can see that the error goes down a bit then goes up again. Increasing lambda results in less number of nonzero coefficients.
Note: 3 points for the error plot, **3 points** for the model size plot, and **2 points** for the test set error.

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?

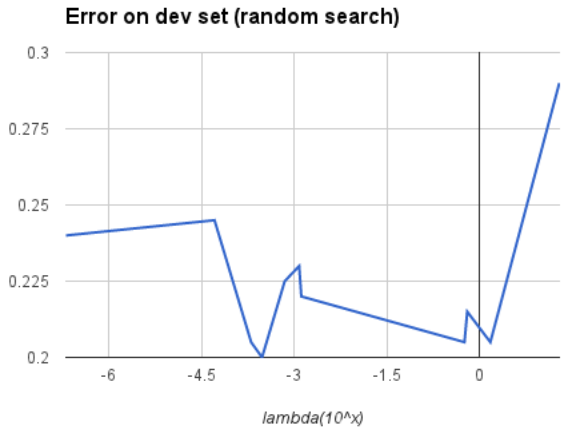


Figure 2: Dev errors using random search.

We want our parameter estimates to minimize the mean squared error:

$$\text{MSE} = \text{bias}^2 + \text{variance}$$

Greater λ corresponds to higher bias (because the true model could be based on more features), less variance (because there are less parameters to estimate), and less complex model (because there are less number of features).
Note: 1 point each for bias, variance, and model complexity.

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.

I would pick models with the smallest size (least number of nonzero coefficients) for computational and memory efficiency.
Note: some people might say they would rather choose the bigger model. As long as they can reasonably justify their choice, you can give **2 points**. If the reasoning does not make sense, please give **0 point** for both smaller or bigger models answer.

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.

See Figure 2. In my case, the test set error is 0.28 ($\lambda = 3 \times 10^{-4}$).
Note: the plot and test set error will be different because we use random search. As long as the plot looks correct, you can give **5 points**.

Random vs. grid search [2 points]

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

For models with a small number of hyperparameters (e.g., logistic regression), they both are good and reliable. We can perform both random search and grid search as long as we define the grid and interval + sampling distribution in a reasonable manner.
However, random search has been shown to perform better compared to grid search for models with a large number of hyperparameters (e.g., neural networks). This is mainly because not all hyperparameters are equally important to tune, and grid search spends a lot of time in dimensions that matter less (because it searches in a grid). See [2] for a complete analysis and empirical evidence.
Note: there is not really a correct answer for this, I just want you to be aware of this research area. If the explanation makes sense, you can give **2 points**. If not clear, give **1 point**. If it looks really wrong, give **0 point**.

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] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research, 13:282:305, 2012.

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