CIS 520, Machine Learning, Fall 2015: Assignment 5
Due: Friday, October 23rd, 11:59pm
[100 points]

Instructions. Please write up your responses to the following problems clearly and concisely. We encourage you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Canvas. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the homework to understand the problem and reach a solution in groups up to size **two students.** However, *each student must write down the solution independently, and without referring to written notes from the joint session.* **In addition, each student must write on the problem set the names of the people with whom you collaborated.** You must understand the solution well enough in order to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

1 Kernel Regression and Locally Weighted Regression [30 points]

Given a set of n examples, $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, a linear smoother is defined as follows. For any $\mathbf{x}$, there exists a vector $\ell(\mathbf{x}) = (\ell_1(\mathbf{x}), \ldots, \ell_n(\mathbf{x}))^\top$ such that the estimated output $\hat{y}$ of $\mathbf{x}$ is $\hat{y} = \sum_{i=1}^n \ell_i(\mathbf{x}) y_i = \ell(\mathbf{x})^\top Y$ where Y is a $n \times 1$ vector, $Y_i = y_i$. This means that the prediction is a linear function of the training responses (y_is) and it varies slowly and smoothly with change or noise in y_is.

1. (6 points) Recall that in linear regression with basis functions h, we assume the data are generated from the model $y_i = \sum_{j=1}^m w_j h_j(\mathbf{x}_i) + \epsilon_i$. The least squares estimate for the coefficient vector $\mathbf{w}$ is given by $\mathbf{w}^* = (H^\top H)^{-1} H^\top Y$, where H is a $n \times m$ matrix, $H_{ij} = h_j(\mathbf{x}_i)$. Given an input $\mathbf{x}$, what is the estimated output $\hat{y}$? (Matrix form solution is required. You may want to use the $m \times 1$ vector $h(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_m(\mathbf{x})]^\top$) Is linear regression is a linear smoother?

2. (6 points) In kernel regression using the kernel $K(\mathbf{x}_i, \mathbf{x}) = \exp\{\frac{-||\mathbf{x}_i - \mathbf{x}||^2}{2\sigma^2}\}$, given an input $\mathbf{x}$, what is the estimated output $\hat{y}$? Is kernel regression is a linear smoother?

3. (6 points) In locally weighted regression, given an input $\mathbf{x}$, what is the estimated output $\hat{y}$? Is locally weighted regression is a linear smoother?

4. (6 points) If we divide the range (a, b) (a and b are real numbers, and $a < b$) into m equally spaced bins denoted by $B_1, \ldots, B_k$. Define the estimated output $\hat{y} = \frac{1}{|B_k|} \sum_{i:\mathbf{x}_i \in B_k} y_i$, for $\mathbf{x} \in B_k$, where $|B_k|$ is the number of points in B_k. In other words, the estimate $\hat{y}$ is a step function obtained by averaging the y_is over each bin. This estimate is called the regressogram. Is this estimate a linear smoother? If yes, give the vector $\ell(\mathbf{x})$ for a given input $\mathbf{x}$; otherwise, state your reasons.

5. (6 points) Suppose we fit a linear regression model, but instead of sum of residual squares $||H\mathbf{w} - Y||_2^2$, we minimized the sum of absolute values of residuals: $||H\mathbf{w} - Y||_1$. Is the result a linear smoother?

Prove (give formula for $\ell(\mathbf{x})$) or disprove (give a counter-example). Hint: Think about the median—for a set of real numbers $(y_1, \ldots, y_n)$ where n is odd, the median y_M minimizes the sum of absolute differences $M = \arg\min_j \sum_{i=1}^{n} |y_j - y_i|$.

2 Supervised Deep Learning [20 points]

Neural networks are among the most powerful machine learning models, which can represent complex, non-linear functions. However, due to a high number of hyperparameters, neural networks are extremely difficult to tune for a particular task. In this exercise, we will apply neural networks for the task of handwritten digit recognition in a supervised and unsupervised setting. This is the same data set and task that you used for decision trees in homework 2 (we won't ask you to for this assignment, but try comparing the performance on your own). As a quick review, you are given samples of 28x28pixel images containing a handwritten digit with value0-255. Our goal is to predict the digit 0 to 9 being drawn. Specifically, for this assignment, we will examine how the choice of different parameters affect the performance of a model. We provide all of the necessary code to complete this homework in *DL_toolbox* folder. You should only need to modify parameters in the parent-level directory, specifically the *demo_NN_*.m* files, but feel free to poke around the other folders!

1. (5 points) File *demo_NN.m* contains the code required to train a neural network. You will now train two models: a plain neural network and a neural network that employs L_2 weight decay (another term for L_2 regularization of the weights). These models are both fully connected between each layer with no pooling or convolutional layers. There are 3 layers. 1 input layer where each input maps directly to a single pixel, 1 hidden layer which we can tune, and 1 output layer where each output maps to a single class (a number 0-9). Report the error on the test set of both models. Which model achieves lower testing error? Can you provide an explanation why?

2. (5 points) Now we will examine how a different choice of L_2 weight decay parameter, the same as λ parameter in regression, affects the performance of the model. Use *demo_NN_L2_decay.m* to train the models with L_2 weight decay parameters 0.01 and 0.001. Report the testing error achieved by both models. Which model performs better? How does the performance compare to the model that we trained in the previous part (ie. model that used 0.0001 as its L_2 weight decay parameters)? Why?

3. (5 points) Dropout is a very popular technique in deep learning community that enforces regularization in the neural network. The basic idea behind dropout is simple: during the training a selected fraction of neurons are zeroed out. We will now explore how the choice of a dropout parameter affect the performance of the model. Use *demo_NN_dropout.m* to train the model with dropout parameters of 0.25 and 0.75. That is, each of these models will zero out 25% and 75% of the neurons respectively. Report the testing error achieved by each of these models. Which parameter value achieves the best performance? Why? Does the dropout help to reduce testing error compared to the model without a dropout?

4. (5 points) Finally, we will examine how the size of a hidden layer affect the model's performance. Use *demo_NN_hidden_size.m* to train the model with hidden layer size parameters of $25, 100$ and 175. Report the testing error achieved by each of these models. Which hidden layer size parameter achieves the best performance? Why do you think that is? Are there any disadvantages to the model that achieves lowest testing error relative to the model that achieves worst testing error in this case?

3 Unsupervised Deep Learning [15 points]

We now turn to unsupervised neural nets, which are also known as auto-encoders. (We will cover these in detail later in class.) Instead of minimizing the error with respect to ground truth, these models are trained to minimize the input reconstruction error (ie. treating input as ground truth). Auto-encoders can be viewed as models that learn a new non-linear feature representation. That is, we can take the

hidden layers of a trained auto-encoder and use it as input features to a supervised learning method. In this exercise, we will examine if auto-encoders can learn a good feature representation and also how different parameter choices affect their performance.

5. (5 points) Use *demo_SAE.m* to train a plain auto-encoder. Visualize the learnt filters inside the auto-encoder using the *visualize* function that is provided in the code. Embed the visualized filters in your writeup. Also save the trained model into *models* directory as *SAE.mat*.

6. (5 points) Use *demo_SAE.m* to train an auto-encoder with half of the input features zeroed out randomly. This parameter can be set by changing *inputZeroMaskedFraction* parameter to 0.5. Visualize the learnt filters inside the auto-encoder using the *visualize* function that is provided in the code. Embed the visualized filters in your writeup. Also save the trained model into *models* directory as *SAE_noisy.mat*. Visually compare the learnt filters from this part to the filters that were learnt in the previous part? What is the major difference? Judging from the visualizations do you think one of these auto-encoders learnt better features than the other?

7. (5 points) Take a look at *demo_SAE_supervised.m* and use the hidden layer representation of autoencoder models from the previous parts as input to a supervised neural network. Do this with both models: *SAE.mat* and *SAE_noisy.mat*. Report the testing errors achieved in both cases. Which auto-encoder features produce lower testing error? Does using the features learnt by auto-encoders achieves lower or higher testing error in comparison to using plain features? Compare the actual testing errors achieved by both approaches.

4 Lagrange Duality and the LASSO [35 points]

It is often the case that we will wish to include additional constraints on our regularization terms. For instance if we wish to limit the size of our weights to be below a specific constant term, or if we wish that all of our weights should be non-negative, etc. in these scenarios, Lagrange multipliers are a powerful tool that allows us to incorporate these constraints into our objective functions.

Also, in general, we have seen for $n \times p$ data, with n observation and p features that $n >> p$ where p is the dominant factor for the computation time in things like linear regression training, which is fine. However, in some scenarios $p >> n$, and this becomes a problem such as when we use the kernel trick! We find that reformulating these problems into their *duals* gives us the alternative dominant factor.

Here we will explore both of these phenomena simultaneously and see how to formulate a problem with the Lagrangian dual!

This problem follows the formulation in Osborne et al, 2000:
http://www4.stat.ncsu.edu/~hzhang/paper/osborne.pdf

By its original definition, the LASSO estimator solves the optimization problem

$$\text{minimize}_{w_1,\ldots w_m} \quad f(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} x_{ij}w_j\right)^2 \tag{1}$$

subject to

$$f_1(\mathbf{w}) = \sum_{j=1}^{m}|w_j| \leq t \quad \text{for some } t > 0. \tag{2}$$

I.e. it minimizes half the sum of squared errors, subject to the constraint that $|\hat{\mathbf{w}}|_1 \leq t$

In this question, you will use Lagrangian duality to show that this formulation is equivalent to the following optimization problem (i.e. standard L_1-regularized regression):

$$\text{minimize}_{w_1,\ldots w_m} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} x_{ij}w_j\right)^2 + \lambda\sum_{j=1}^{m}|w_j|, \quad \text{for some } \lambda \geq 0. \tag{3}$$

Specifically: you will convert the constrained optimization problem 1 into a Lagrange Multiplier problem. From there you will find that **when w is an optimal solution to the constrained optimization problem and λ is the Lagrange multiplier associated with w, then w is also an optimal solution of the regularized regression problem with penalty coefficient λ.** Your task is to complete part of this proof.

1. (5 points) The function 3 is convex in $\mathbf{w}$ (quadratic plus L_1 norm), which means its minima satisfy $\nabla_w(\text{Eq. 3}) = 0$. Compute the gradient with respect to $\mathbf{w}$ of this expression and show that the resulting condition for any optimal $\mathbf{w}$ is

$$\mathbf{0} = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{v}, \tag{4}$$

where $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{R}^m$ satisfies $v_i = \begin{cases} 1 & \text{if} \quad w_i > 0 \\ -1 & \text{if} \quad w_i < 0 \\ \in [-1, 1] & \text{if} \quad w_i = 0 \end{cases}$

This awkward-seeming vector $\mathbf{v}$ is the result of taking the derivative of absolute value terms $|w_i|$, where the w_i are of unknown sign. *Hint: this is similar to the derivation of the normal equation in the regression notes.*

2. (5 points) Set up the constrained optimization problem 1 to be solved using the method of Lagrange multipliers. Write down the Lagrangian $L(\mathbf{w}, \lambda)$ and Karush-Kuhn-Tucker conditions.

3. (5 points) The next step is to take each case of the KKT conditions and solve the constrained optimization problem using Lagrange multipliers. To do this, note that you showed in HW 3 that if $\mathbf{X}$ is full-rank and $m \leq n$, then the ordinary least-squares regression (OLS) problem has a unique solution:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{5}$$

On the other hand, if $m > n$, then the solution $\hat{w}$ to the OLS problem is not unique. Let $\mathbf{W}$ be the set of solutions to a particular OLS problem. One can show (see Osbourne et al.) that if t_0 is defined as

$$t_0 = \min_{\mathbf{w}\in\mathbf{W}} ||\mathbf{w}||_1 \qquad \text{(L1 norm)} \tag{6}$$

then the constraint 2 is active if and only if $t < t_0$. In other words, if $t \geq t_0$, then the constraint doesn't matter– a solution to 1 just has to optimize the unconstrained OLS problem. On the other hand, if $t < t_0$, then the constraint 2 must be active for any solution $\mathbf{w}$; in other words, this inequality constraint becomes an equality. Moreover, one can show[1] that when $t < t_0$, at least one optimal $\mathbf{w}$ for the constrained problem 1 is guaranteed to exist on the boundary of the feasible region (the *feasible region* is the set of all $\mathbf{w}$ such that the constraint 2 is satisfied).

Based on this discussion, we only need to find solutions for one case of the KKT conditions when $t < t_0$. Which case is this, and why?

4. (5 points) Define[2]

$$L^*(\mathbf{w}) = \max_{\lambda\geq 0} L(\mathbf{w}, \lambda) \tag{7}$$

[1] since $f(\mathbf{w})$ is continuous on its feasible region and the feasible region is compact

[2] To be technical, this should be the supremum (least upper bound) rather than the maximum; we will not worry about this distinction here.

Show that

$$L^*(\mathbf{w}) = \begin{cases} f(\mathbf{w}) & \text{if} \quad (t - f_1(\mathbf{w})) \geq 0 \\ \infty & \text{if} \quad (t - f_1(\mathbf{w})) < 0 \end{cases} \tag{8}$$

Argue that, therefore, when $t < t_0$, minimizing $L^*(\mathbf{w})$ over all possible $\mathbf{w}$ is equivalent to solving the constrained optimization problem 1. Minimizing $L^*(\mathbf{w})$ is known as the *primal problem.*
Hint: Show that if $\mathbf{w}$ *optimizes the constrained problem, then it also must minimize* L^**. Next, show that if* $\mathbf{w}$ *minimizes* L^**, then it also must optimize the constrained problem. Proof by contradiction may be useful.*

5. (5 points) Define[3]

$$g(\lambda) = \min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{w}, \lambda) \tag{9}$$

Maximizing $g(\lambda)$ for $\lambda \geq 0$ is the *dual problem* to the primal shown above.

One can show[4] that, for fixed λ, $L(\mathbf{w}, \lambda)$ has at least one minimum over $\mathbf{w}$. We can solve for a minimum in the usual way: by taking the gradient of L with respect to $\mathbf{w}$ and setting it to zero. Do this to show that if $\bar{\mathbf{w}}$ minimizes $L(\mathbf{w}, \lambda)$ for fixed λ, then

$$\mathbf{0} = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}\bar{\mathbf{w}}) + \lambda \mathbf{v}, \tag{10}$$

with $\mathbf{v}$ defined as before, in 4.

6. (5 points) Show that $\mathbf{v}^T\bar{\mathbf{w}} = ||\bar{\mathbf{w}}||_1$. Consequently, show that if $\bar{\mathbf{w}}$ minimizes $L(\mathbf{w}, \lambda)$ for fixed λ, then

$$\lambda = (\mathbf{y} - \mathbf{X}\bar{\mathbf{w}})^T\mathbf{X}\bar{\mathbf{w}}/||\bar{\mathbf{w}}||_1. \tag{11}$$

We call this value of λ *the Lagrange multiplier associated with* $\bar{\mathbf{w}}$*. Hint: recall that* $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$.

It turns out that *strong duality* holds for this problem (see Theorem 3 in Osborne et al), which means: if $\mathbf{w}^*$ is a solution of the primal problem (equivalently, of the constrained optimization problem 1) and λ^* is the Lagrange multiplier corresponding to $\mathbf{w}^*$, then λ^* is a solution of the dual problem, and hence the optimal primal and dual function values are equal:

$$g(\lambda^*) = L^*(\mathbf{w}^*) = f(\mathbf{w}^*), \tag{12}$$

for f as defined in 1.

Let's review: we began by trying to solve the constrained optimization problem 1. We saw that this is equivalent to solving the primal problem 7. Due to strong duality, we can solve either the primal or dual problem to obtain an optimal $\mathbf{w}$ for 1. Given a solution $\mathbf{w}$ for 1, we can compute the Lagrange multiplier associated with it using 11. On the other hand, given λ, any optimal $\mathbf{w}$ for the penalized regression problem 3 satisfies exactly the same equations (4) as are satisfied by solutions to 1. This means: when $\mathbf{w}$ is an optimal solution to the constrained optimization problem and λ is the Lagrange multiplier associated with $\mathbf{w}$, then $\mathbf{w}$ is also an optimal solution of the penalized regression problem with penalty coefficient λ.

7. (5 points) Load the file lagrange_simulation.m into Matlab. This is a numerical simulation of what you have seen in the previous questions. Specifically: the simulation will generate data, and using numerical methods, it will compute an (approximately) optimal solution $\mathbf{w}_{OPT}$ to the constrained optimization problem 1. It will then compute the associated Lagrange multiplier λ and show that $\mathbf{w}_{OPT}$ is also an optimal solution to the penalized regression problem 3 with penalty coefficient λ. (This makes sense, since the zero-gradient relation between λ and $\mathbf{w}$ is the same for both problems!) Thus we can look

[3] Likewise, technically this should be the infimum (greatest lower bound) rather than the minimum.
[4] since $L(\mathbf{w}, \lambda)$ is convex and goes to $+\infty$ as $||\mathbf{w}||_1 \to \infty$

empirically at how t and λ are related.

Run the file a few times each for $t = 6, 10, 14$.

(a) For the different values of t, what do you notice about the differences between $\mathbf{w}_{MLE}$ (the solution computed for the OLS problem) and $\mathbf{w}_{OPT}$? How does the L1 penalty tend to zero out coefficients of $\mathbf{w}_{OPT}$ for the different values of t? Based on the coefficients used to generate the data (see in the file), is this what you'd expect?

(b) How do the computed λ's change with t? Why is this?