6.390: Midterm Exam, Spring 2023

Solutions

- This is a closed book exam. One page (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Calculators are not permitted.
- The total exam time is 2 hours.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every piece of paper.**

Name: ____________________ MIT Email: ____________________

Question	Points	Score
1	19	
2	12	
3	19	
4	12	
5	18	
6	11	
7	9	
Total:	100	

Name: ______________________________

Regression or Progression?

1. (19 points) Suppose that you are given a small dataset and you would like to learn the parameters of a linear regressor hypothesis taking the form $h(x) = \theta^\top x + \theta_0$ for fitting the data.

   (a) First, consider the following dataset $\mathcal{D}_1$ containing three feature-label pairs:

x	y
-4	3
2	-1
-1	0

   Suppose that you would like to minimize the following objective function:

$$J_1(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2.$$

   You decide that you would like to find the analytic solution. You start by defining $Y = [y^{(1)},\ y^{(2)}, \dots,\ y^{(n)}]^\top$ and $\bar{\theta} = [\theta,\ \theta_0]^\top$ and rewrite your objective function as,

$$J_2(\bar{\theta}) = \frac{1}{n}(\tilde{X}\bar{\theta} - Y)^\top(\tilde{X}\bar{\theta} - Y).$$

   i. What is the data matrix $\tilde{X}$ with data points as rows corresponding to the dataset $\mathcal{D}_1$ which ensures that the two objective functions J_1, J_2 are equivalent?

Solution: $\tilde{X} = \begin{bmatrix} -4 & 1 \\ 2 & 1 \\ -1 & 1 \end{bmatrix}$

   ii. Does J_2 have an analytic solution for dataset $\mathcal{D}_1$? Explain your answer. (Note: you do not need to compute the analytic solution.)

Solution: Yes; the matrix $\tilde{X}^\top\tilde{X}$ is invertible.
One way to see this is by calculation, e.g. we can see that the determinant of $\tilde{X}^\top\tilde{X}$ is non-zero. Another way to see invertability is that this data set has more data points than features and the features are not linearly dependent.
This objective function therefore has a unique analytic solution.

(b) As your dataset is quite small, you decide to add a second feature for each of the three datapoints. Consider the new dataset $\mathcal{D}_2$ defined as:

x_1	x_2	y
-4	-8	3
2	4	-1
-1	-2	0

i. Does J_2 have an analytic solution for dataset $\mathcal{D}_2$? Explain your answer. (Note: you do not need to compute the analytic solution.)

Solution: No; the second feature is just the first feature scaled by two. The features are linearly dependent and $(\tilde{X}^\top \tilde{X})$ is not invertible. This implies the objective function does not have an analytic solution.

ii. Consider a new objective function with an added regularization term:

$$J_3(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2 + 0.2(\|\theta\|^2 + \theta_0^2)$$

Will adding this particular form of regularization in J_3 improve the generalization of our model? Explain your answer.

Solution: This regularization is not a good idea. Adding this regularization penalizes θ_0. This is going to drive θ_0 towards 0. As discussed in the notes, forcing θ_0 to be zero potentially hurts the generalization of the model.

iii. Does J_3 have an analytic solution for dataset $\mathcal{D}_2$? Explain your answer. (Note: you do not need to compute the analytic solution.)

Solution: Yes. The notes has discussion on that ridge regularization guarantees unique analytical solution.

(c) Consider a new dataset $\mathcal{D}_3$ which includes a fourth data point:

x_1	x_2	y
-4	-8	3
2	4	-1
-1	-2	0
5	8	-4

Does J_2 have an analytic solution for dataset $\mathcal{D}_3$? Explain your answer. (Note: you do not need to compute the analytic solution.)

Solution: The features of the last data point are not linearly dependent; therefore $(\tilde{X}^\top \tilde{X})$ is invertible. This implies that this system has one unique solution.

Discerning Descents

2. (12 points) Indicate whether the following statements are true or false. Explain your answer.

   (a) Gradient descent is sure to converge, to some value, for any step size greater than 0.

   Solution: False. The step-size can be too large, which may cause the algorithm to diverge.

   (b) Stochastic gradient descent reduces the objective function value at every iteration.

   Solution: False. Depending on which index is randomly chosen, the gradient direction with respect to that data point may increase the objective function value at a single iteration.

   (c) When the algorithms converge, stochastic gradient descent always finds the same solution as gradient descent.

   Solution: False. For example, if there are multiple local minima, even two different runs of gradient descent may converge to different local minima, depending on the random initial condition. Adding on top of this the stochastic/randomness from stochastic gradient descent makes the statement definitely false.

   (d) The more features that we use to represent our data, the better the learning algorithm will generalize to new data points.

   Solution: Two lines of reasoning are okay:

   - False; it depends on the kind of features you are adding. For instance, if we just added "nonsense" or irrelevant features, these features would not be helping.
   - True; *if* one assumes that the features are more informative it should reduce the chance of over-fitting.

Logistic Regression Logistics

3. (19 points) You would like to train a binary linear logistic classifier of the form,

$$h(x;\theta,\theta_0)=\sigma(\theta^T x+\theta_0),$$

where $\sigma(z)=1/(1+e^{-z})$ is the sigmoid function, by minimizing an objective function,

$$J(\theta,\theta_0)=\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(g^{(i)},a^{(i)})+\lambda\|\theta\|^2,$$

where $g^{(i)}=h(x^{(i)};\theta,\theta_0)$ is the "guess" or model output and $a^{(i)}=y^{(i)}$ is the "actual" label.

(a) Typically, we use the negative log-likelihood (NLL) as the loss function for classification,

$$\mathcal{L}_{\text{NLL}}(g,a)=-(a\log g+(1-a)\log(1-g)).$$

Additionally, recall that

$$\frac{\partial\mathcal{L}_{\text{NLL}}(g,a)}{\partial g}=\frac{1-a}{1-g}-\frac{a}{g}.$$

We could also consider using squared loss for classification,

$$\mathcal{L}_{\text{squared}}(g,a)=(g-a)^2.$$

i. Consider the case when our guess is $g\approx 0$ but the actual value should be $a=1$. Compute the following values:

$\mathcal{L}_{\text{NLL}}(g,a)\approx$ **∞ (or undefined, DNE, etc.)**

$\mathcal{L}_{\text{squared}}(g,a)\approx$ **1**

$\partial\mathcal{L}_{\text{NLL}}(g,a)/\partial g\approx$ **$-\infty$ (or undefined, DNE, etc.)**

$\partial\mathcal{L}_{\text{squared}}(g,a)/\partial g\approx$ **-2**

ii. Now, let's instead assume that $g=0.5$ and the actual label is still $a=1$. Compute the following values:

$\partial\mathcal{L}_{\text{NLL}}(g,a)/\partial g=$ **-2**

$\partial\mathcal{L}_{\text{squared}}(g,a)/\partial g=$ **-1**

iii. Is it ever possible to obtain a guess of $g = 1$? Explain why or why not.

Solution:
We can never get $g = 1$ since sigmoid only approaches 1 asymptotically.

iv. Based on these observations, briefly explain why we might prefer one of the loss functions presented above over the other when training a linear logistic classifier.

Solution:
NLL typically has a larger gradient and leads to faster, more reliable convergence to a low-error solution.
If you consider "these observations" from part (a) i. in a vacuum, you may be tempted to conclude that the squared-error loss may be preferable because it avoids the issue of have undefined gradients revolving around division by zero. However, as observed in part iii., we can never get to $g = 1$ (or $g = 0$), as the sigmoid only approaches these values asymptotically.

(b) You decide to train your binary linear logistic classifier by minimizing the J objective function with the NLL loss function by using gradient descent. A few of your friends comment on your methodology.

i. Chris thinks that sigmoids are too confusing! He suggests that you can replace the sigmoid function with a step function for training your model.

Does Chris's scheme make sense? Explain your answer.

Solution: Recall that we are trying to learn a classifier by minimizing the objective function using gradient descent. Here, with the step function, the gradient is zero everywhere or undefined and the gradient updates will not be meaningful.
Credit was also given for noting that using the step function with NLL loss may lead to issues similar to part (a)i.
It is key to note that the issues arise during training using gradient descent. A linear classifier on its own is a valid choice if the parameters are already given.

ii. Jojo claims that you can generalize from binary to trinary classification as follows: let the hypothesis take the form $h_3(x; \theta, \theta_0) = 3\sigma(\theta^\top x + \theta_0)$, and learn the parameters θ, θ_0 by minimizing the same objective J with NLL loss, with class labels $0, 1, 2$. Then, you can assign classes using the output of your hypothesis as,

$$\begin{aligned} &\text{class } 0, && \text{if } 2 \le h_3(x), \\ &\text{class } 1, && \text{if } 1 \le h_3(x) < 2, \\ &\text{class } 2, && \text{if } h_3(x) < 1. \end{aligned}$$

Does Jojo's scheme make sense? Explain your answer.

Solution: This scheme does not make sense for many reasons:

- Dividing the range of the sigmoid into three regions does not have any particular meaningful implications for the classification
- The monotonic nature of sigmoid would imply that the data would have to be such that, spatially, the decision region for class 0, 1, and 2 must happen sequentially.
- Trying to use the NLL loss function with a label of 2 is also not sensible.

In essence, this scheme abstracts away from all of the intenions of training a linear logistic classifier.

iii. Mona overhears your conversation with Jojo and suggests that instead of using a single classifier, you learn two different binary linear logistic classifiers.
First, learn one binary classifier h_{01} by minimizing J with the NLL loss function on a dataset where class 0 is labeled 0 and class 1 is labeled 1, and class 2 feature vectors are omitted.
Then, learn a second classifier h_{12} by minimizing J with the NLL loss function on a dataset where class 0 feature vectors are omitted, class 1 is labeled 0, and class 2 is labeled 1.
You can assign classes using the output of your two hypotheses by performing the following sequentially:

$$\begin{aligned} \text{if } h_{01}(x) < 0.5, &\quad \text{assign class } 0, \\ \text{else, if } h_{12}(x) < 0.5, &\quad \text{assign class } 1, \\ \text{otherwise}, &\quad \text{assign class } 2. \end{aligned}$$

Does Mona's scheme make sense? Explain your answer.

Solution: Notice that this scheme is a sort of incomplete one-vs-one set up. A third binary classifier should be used to differentiate between classes 0 and 2, in conjunction with some sort of notion of "confidence" or "margins" to break ties. While each classifier may be learned in a sensible way, the classifications outlined will not be sensible when trying to do this multi-class classification task.

Name: ______________________________

Won't You Be My Neighbor?

4. (12 points) Indicate whether the following statements are true or false. Explain your answer.

   (a) Consider a classification problem where the training dataset consists of n data points that all have different feature values.

      i. A k-nearest-neighbor classifier with $k = 1$ will always have 100% training accuracy on this dataset.

      Solution: True. For any training data point, the 1 nearest neighbour is always itself. Therefore, the predicted class label will by construction be correct. So the training accuracy is 100%.

      ii. A k-nearest-neighbor classifier with $k > 1$ will always have 100% training accuracy on this dataset.

      Solution: False. The data points in the training dataset do not necessarily have the same label values. Some data points may have labels different from the particular training data point in consideration. When taking a majority vote of class labels from any point's k neighbours, the majority vote may differ from this point's true label. Therefore, no guarantee of 100% training accuracy.

      iii. In general, using a k-nearest-neighbors classifier with $k > 1$ as opposed to 1-nearest-neighbor can effectively reduce the tendency of the model to overfit to training data.

      Solution: True. With $k = 1$, the classifier's decision boundary is heavily influenced by each individual data point. Any potential noise in the training data set is significantly impacting the decision boundary – classical symptom of over-fitting. When k increases, the decision boundary is smoothed out, meaning that some local noises are getting ignored, therefore higher k reduces the tendency to overfit to the local data.

(b) Consider a regression problem where the training dataset consists of n data points that all have different feature values.

i. The MSE (mean squared error) of a k-nearest-neighbor regressor with $k = 1$ will always be 0 on this dataset.

Solution: True. Similar to the classification case, in regression, running 1-nn on the training dataset would also return the label of the particular training data itself. Graphically, when the training data is all of different feature values, the regressor will go through all the data points. This implies that MSE on this dataset will be zero.

ii. The MSE (mean squared error) of a k-nearest-neighbor regressor with $k > 1$ will always be 0 on this dataset.

Solution: False. For instance, say $k = 2$, and for a training data (x, y) its two nearest neighbors are (x_1, y_1) and (x, y). The prediction $(y_1 + y)/2$ is not necessarily equal to y.

iii. A k-nearest-neighbor regressor with $k = n$ trained on this dataset will always output constant prediction.

Solution: True. Suppose the labels of data point i in the training data set is $y^{(i)}$, then the prediction made by this k-NN with $k = n$ (because n-NN sounds funny) will be $\frac{1}{n}\sum_{i=1}^{n} y^{(i)}$.

Name: ____________________

The Rise of ELU

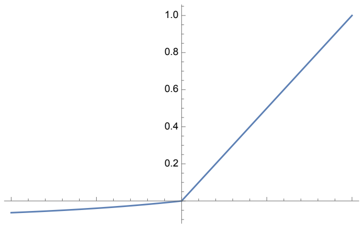
5. (18 points) We have been considering the ReLU function for an internal non-linearity in neural networks:

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let's also consider the ELU (exponential linear unit) function:

$$\text{ELU}_\alpha(z) = \begin{cases} z & \text{if } z > 0, \\ \alpha(e^z - 1) & \text{otherwise,} \end{cases}$$

which is shown here for $\alpha = 0.1$:

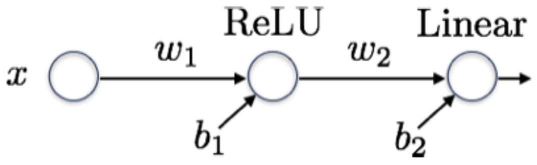


(a) What is $\partial\text{ELU}_\alpha(z)/\partial z$?

Solution:

$$\partial\text{ELU}_\alpha(z)/\partial z = \begin{cases} 1 & \text{if } z > 0 \\ \alpha e^z & \text{otherwise} \end{cases}$$

(b) We have the simple neural network shown below:



We will train the neural network using the squared loss function,

$$\mathcal{L}_{\text{squared}}(g, a) = (g - a)^2.$$

Our target output is 1 and the initial weights (including the offsets b_i) are -0.1.

i. Assume the input is $x = 1$. What is the sign of the update that will be applied to w_1, using a step size of 1? Explain your answer.

Solution: The gradient of the loss with respect to w_1, has a factor which is the derivative of ReLU with respect to z. Since the activation of the ReLU is negative, the gradient of the ReLU is zero. Therefore the gradient update is zero.

ii. Assume that instead of the ReLU unit we have an **ELU** unit. What is the sign of the gradient update that will be applied to w_1, using a step size of 1? Explain your answer.

Solution: The gradient update is

$$w_1 := w_1 - \eta \partial \mathcal{L}_{\text{squared}} / \partial w_1$$

$$\partial \mathcal{L}_{\text{squared}} / \partial w_1 = 2(g - a) w_2 \alpha e^z x$$

Note that $(g - a)$ is negative, w_2 is negative, e^z is positive and x is positive. So the gradient of the loss will be positive and thus the update will be negative. (assuming $\alpha > 0$ and small).

iii. Assume we initialize our networks with weights drawn uniformly from the range $[-.1, .1]$ What advantage does using **ELU** have over **ReLU** for use in a multi-layer neural network?

Solution: Initially, when the weights are small, many of the ReLU units will be "dead", with zero output and zero gradient. Which could be problematic as zero gradient means no update to the weight.

iv. What advantage does using **ELU** have over the **linear** activation function for use in the internal units of a multi-layer neural network?

Solution: Linear activations produce an overall linear classifier. ELU introduces a necessary non-linearity for improved expressiveness of the network.

Name: ______________________________

Classification (feat. Features)

6. (11 points) Here is a training data-set of four data points (with 1-dimensional feature x) for a classification problem:

x	y
-1	-1
0	$+1$
1	$+1$
2	-1

(a) What are the feature values under the feature transformation $\phi(x) = [x, x^2, (x-1)^2]^\top$ for each of these four data points?

Solution: $\phi(-1) = [-1, 1, 4]^\top,\ \phi(0) = [0, 0, 1]^\top,\ \phi(1) = [1, 1, 0]^\top,\ \phi(2) = [2, 4, 1]^\top$

(b) The given data set is linearly separable in the new feature space $\phi(x) = [x, x^2, (x-1)^2]^\top$. Suppose we have a linear classifier with $\theta = [-1, -1, -1]^\top$ and an unknown θ_0. Provide a value of θ_0 that results in a correct separator on the training data. Justify your answer.

Solution: Any value between 2 and 4 will work. One can see this by computing the $\theta^\top \phi(x)$:
$\theta^\top\phi(-1) = -4,\ \theta^\top\phi(0) = -1,\ \theta^\top\phi(1) = -2,\ \theta^\top\phi(2) = -7.$
Adding, e.g., $\theta_0 = 3$ to each of these numbers will result in $\theta^\top\phi(x) + \theta_0 > 0$ for the data points labeled $+1$ and $\theta^\top\phi(x) + \theta_0 < 0$ for data points labeled -1.

(c) Suppose we use a linear classifier $\theta = [-1, -1, -1]^\top$ and $\theta_0 = 4$ in the new feature space $\phi(x) = [x, x^2, (x-1)^2]^\top$. For what range of values in the original x feature space would a data point be classified as positive?

Solution: For a point to be predicted positive we need $\theta^\top\phi(x) + \theta_0 = -x - x^2 - (x-1)^2 + 4 > 0$ which occurs when $-1 < x < \frac{3}{2}$.

Name: ________________________

Detective on the Case

7. (9 points) As a machine learning expert, your friends often come to you looking for advice on how to fix their code. For each of the following snippets of pseudocode, identify which line(s) of code have errors and explain how you would change them.

   Here are five commonly referenced functions. You do not need to debug their contents. You can assume that they operate correctly.

```
def objective_function(feature_vectors, labels, model, hyperparams=None)
    # Defines the objective function
    return objective_function

def objective_value(feature_vectors, labels, model, hyperparams=None ...
                    objective_fn=objective_function):
    # Computes the objective function value
    return objective_value

def train_model(feature_vectors, labels, hyperparams=None, ...
                    objective_fn=objective_function):
    # Learns a model using some algorithm on the dataset
    # e.g., by minimizing an objective function
    return model

def load_dataset(fname):
    # Loads the dataset
    return feature_vectors, labels

def train_val_split(feature_vectors, labels):
    # splits the given dataset into two datasets
    # e.g., for training and validating a model
    return train_vectors, train_labels, val_vectors, val_labels
```

   (a) Find the error(s) in this pseudocode for training and validating a model:

```
1.  x_full, y_full = load_dataset("data.csv")
2.  my_model = train_model(x_full, y_full)
3.  x_train, y_train, x_val, y_val = train_val_split(x_full, y_full)
4.  error = objective_value(x_val, y_val, my_model)

print("My model's performance: " + str(error))
```

Solution: In lines 2 and 3, the model is trained on the entire dataset and then validated on a subset of the dataset, resulting in attempting to validate the model on data it has already seen during training. These two lines should be switched such that the model is trained on one subset of the dataset and validated on the other.

Name: ______________________________

(b) Find the error(s) in this pseudocode for training and testing a regularized model:

```
1.  lam = 0.01
2.  x_full, y_full = load_dataset("data.csv")
3.  x_train, y_train, x_val, y_val = train_val_split(x_full, y_full)
4.  my_model = train_model(x_train, y_train, hyperparams=lam)
5.  error = objective_value(x_val, y_val, my_model, hyperparams=lam)

print("My regularized model's performance: " + str(error))
```

Solution: In line 5, when the testing error is being calculated, it is including the value for the hyperparameter. The `hyperparams` variable should be explicitly set to None or not included in the call to `objective_value`, as it is None by default.

(c) Find the error(s) in this pseudocode for shipping a regularized model with a validated hyperpameter value:

```
1.  x_full, y_full = load_dataset("data.csv")
2.  x_train, y_train, x_val, y_val = train_val_split(x_full, y_full)

3.  lambda_values = [0.0001, 0.001, 0.01, 0.1, 1.0]
4.  min_error = 1e10 # assume that any model will have error < 1e10

5.  for lam in lambda_values:
6.      model = train_model(x_train, y_train, hyperparams=lam)
7.      error = objective_value(x_val, y_val, model)
8.      if error < min_error:
9.          lambda_star = lam
10.         min_error = error

11. best_model = train_model(x_train, y_train, hyperparams=lambda_star)

print("Finished training validated model. Ready for testing!")
```

Solution: In line 11, the final model is only trained on the "train" dataset when it should be trained on the entire dataset `x_full,y_full`, as it is specified that the model is to be shipped (for being used on future, unseen testing data).