ECE597ML-697ML Machine Learning for Engineers, Spring 2021
Quiz

Name (first last) .....

Start date / time .....

End date / time .....

Signature .....

By signing here, I acknowledge that I have read and followed the guidelines below, and I understand that failure to follow these guidelines is a violation of the honor code.

Instructions:

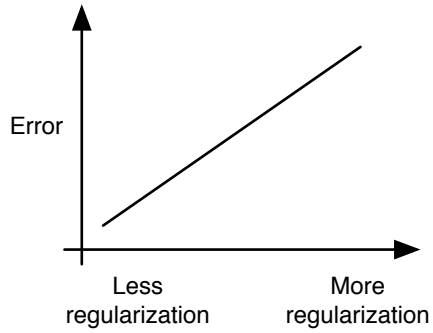
- Complete all of the questions. **Do not look at the questions before you start.**
- Please try fit your solutions in the provided space. If you need extra space, write on the back of the sheet **and clearly indicate that your solution is continued on the back**.

Problem 1. (10 points) Consider training a regularized logistic regression model.

(1 point) Does increasing the regularization parameter λ make the model more complex or less complex? Circle one:

More complex / Less complex

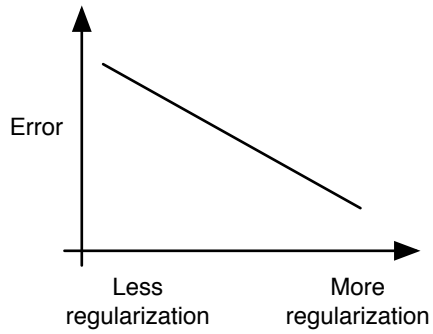
(9 points) Suppose you explore some range of values for λ and record the error both on the training set and on a separate test set for each value of λ. Each plot below shows a possible scenario for how the error (either train or test) varies with the amount of regularization. Your job is to judge whether the plot is consistent with some scenario that could actually occur. (**Note**: "more regularization" means a larger value of λ, and you don't necessarily explore a *full* range of values for λ.) In each case, circle all statements on the right that apply.



(a) This scenario could occur for training error.

(b) This scenario could occur for test error.

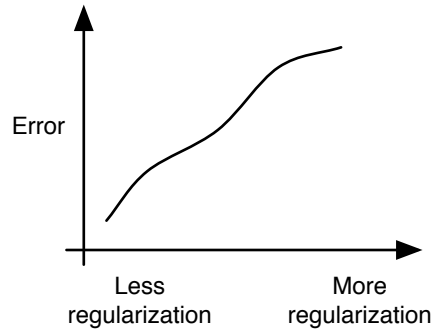
Solution: (a) and (b). This scenario would occur if the model is fit well (on the left of the plot) and then increasing regularization causes underfitting



(a) This scenario could occur for training error.

(b) This scenario could occur for test error.

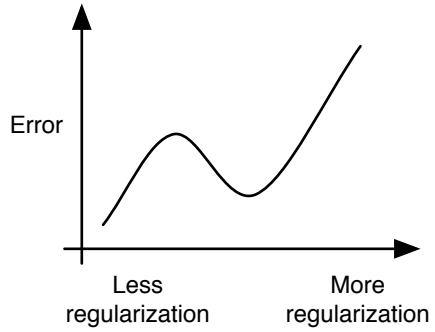
Solution: (b) only. Increasing regularization cannot decrease training error. But it can decrease test error if the original model is overfit.



(a) This scenario could occur for training error.

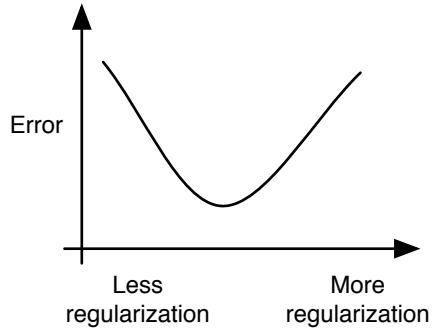
(b) This scenario could occur for test error.

Solution: (a) and (b). This is essentially the same as the first plot.



(a) This scenario could occur for training error.

Solution: None. Decreasing regulariztion can only cause training error to decrease.



(a) This scenario could occur for training error.

(b) This scenario could occur for test error.

Solution: (b) only. This cannot represent training error, which can never decrease with more regularization, as we see on the left of the plot. However, it could represent test error going from overfit on the left of the plot, then decreasing to "just right", and then increasing to underfit on the right of the plot.

Problem 2. (10 points) Geometry of linear regression and logistic regression.

On the following page there are contour plots of five two-dimensional functions labeled (a)-(e). Match the plots with the functions below. For each function, write the letter of the contour plot of that function next to it.

1. $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$, $\boldsymbol{\theta} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$
2. $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$, $\boldsymbol{\theta} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
3. $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$, $\boldsymbol{\theta} = \begin{bmatrix} -10 \\ 10 \end{bmatrix}$
4. $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{logistic}(\boldsymbol{\theta}^T\mathbf{x})$, $\boldsymbol{\theta} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
5. $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{logistic}(\boldsymbol{\theta}^T\mathbf{x})$, $\boldsymbol{\theta} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Solution:

1. (e)
2. (b)
3. (c)
4. (a)
5. (d)

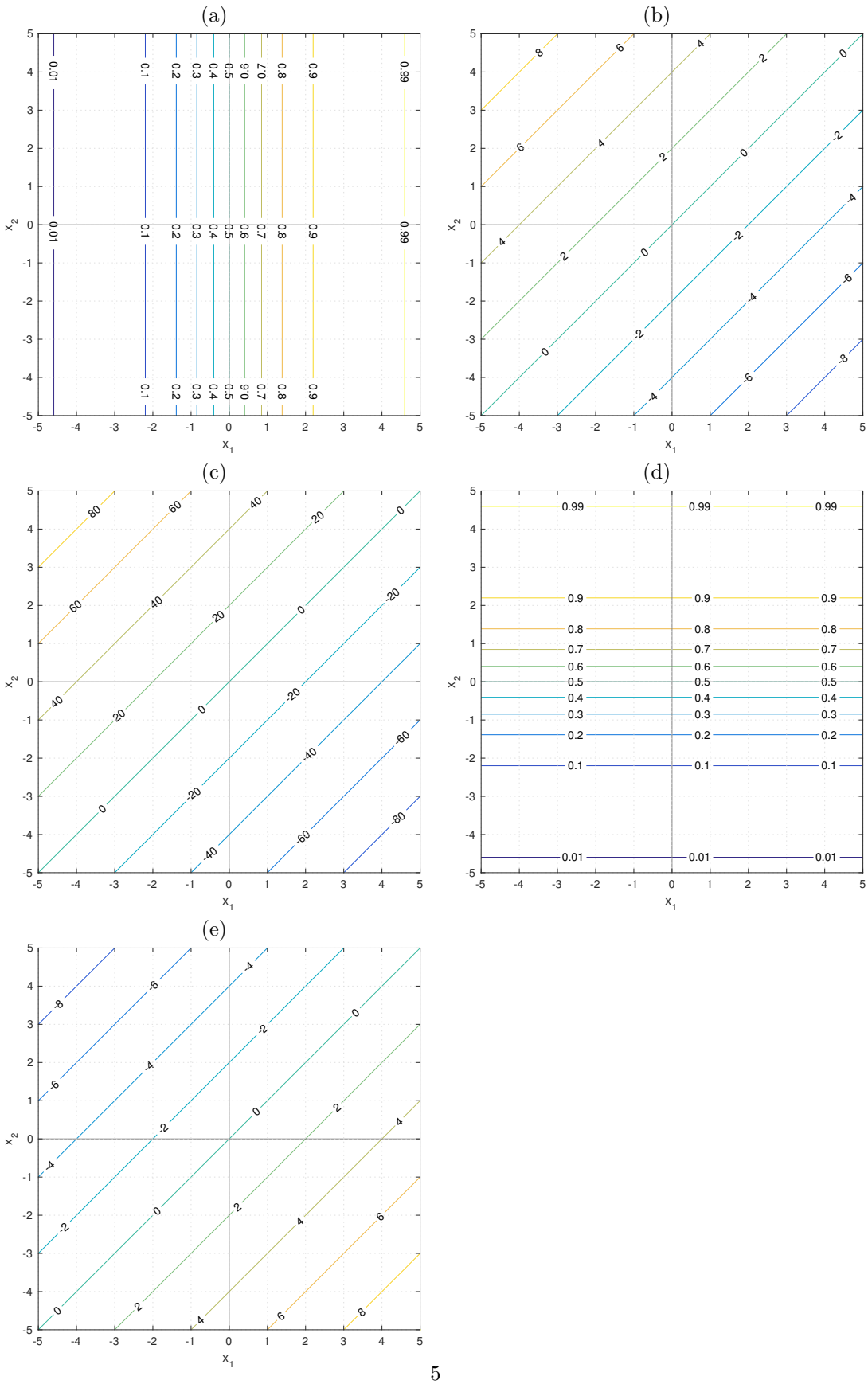


Figure 1: Contour plots for Problem 2

.

Problem 3. (10 points) Overfitting

Several scenarios are described below in which a logistic regression model is fit to a training set and then error is measured both on the training set and a separate test set.

For each scenario, indicate which of the following could be true: (1) the model is underfit, (2) the model is "just right" (neither underfit nor overfit), (3) the model is overfit. Circle each possibility that is consistent with the evidence (there may be anywhere from zero to two correct answers). Provide a brief explanation of your answer.

1. You use regularization with $\lambda = .01$ and the error on the test set is 15%. You try $\lambda = 1$ and the error on the test drops to 10%. Which of the following are possible for the $\lambda = .01$ model?

   underfit / "just right" / overfit

   Solution: The $\lambda = .01$ model is overfit. By reducing λ, we have decreased the complexity of the model and got a better fit.

2. The error on the training set is 10% and the error on the test set is 11%.

   underfit / "just right" / overfit

   Solution: The model could be underfit or "just right". In either case the training error should be similar to the test error.

3. You use regularization with $\lambda = 100$ and the error on the test set is 18%. You try $\lambda = 10$ and the error on the test drops to 13%. Which of the following are possible for the $\lambda = 100$ model?

   underfit / "just right" / overfit

   Solution: The $\lambda = 100$ model is underfit. By reducing λ, we increased the complexity of the model and got a better fit.

4. The error on the training set is 25% and the error on the test set is 1%.

   underfit / "just right" / overfit

   Solution: This situation is impossible.

Problem 4. (8 points) Complexity

For each of the following items, indicate whether it is likely to lead to more overfitting or less overfitting in a learning problem.

(a) Adding irrelevant features to a linear regression problem.

Solution: More overfitting

(b) Increasing the value of λ in regularized logistic regression.

Solution: Less overfitting

(c) Increasing the amount of training data.

Solution: Less overfitting

(d) Increasing the degree of the polynomial in polynomial regression.

Solution: More overfitting

Problem 5. (12 points) Gradient Descent, Normal Equations, Feature Normalization

Suppose we use gradient descent to minimize the cost function $J(\boldsymbol{\theta})$ of either multivariate linear regression or logistic regression. Let $\alpha > 0$ be the gradient descent learning rate.

True or False? If α is too small, gradient descent may take a very long time to converge.

Solution: True

True or False? The value of $J(\boldsymbol{\theta})$ will decrease in every iteration of gradient descent, regardless of how we set α.

Solution: False.

True or False? If $\boldsymbol{\theta}$ is initialized at the minimum of $J(\boldsymbol{\theta})$, then one iteration of gradient descent will not change $\boldsymbol{\theta}$, regardless of how we set α.

Solution: True: the partial derivatives are all zero.

True or False? Feature normalization changes the problem so that we can find a lower value of $J(\boldsymbol{\theta})$ than without normalization.

Solution: False.

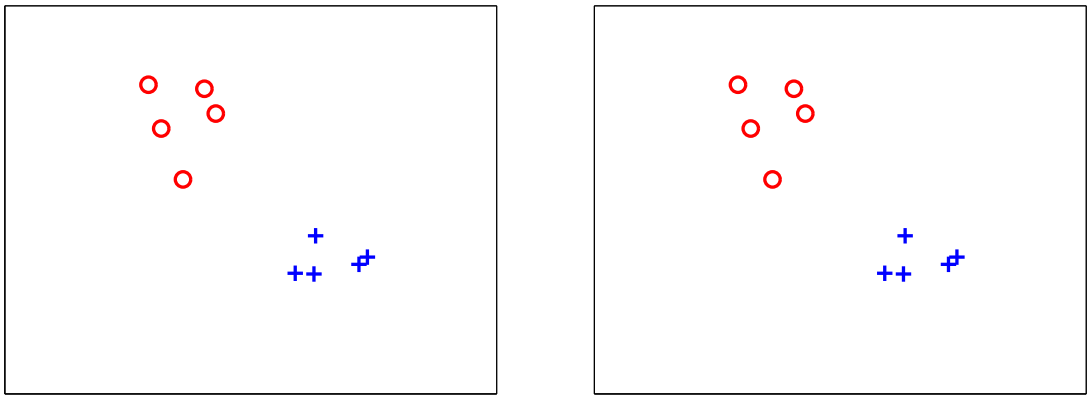
True or False? Feature normalization makes it faster to learn $\boldsymbol{\theta}$ using the normal equations.

Solution: Finding $\boldsymbol{\theta}$ using the normal equations is not an iterative procedure, so its running time is the same with or without feature normalization.

True or False? Feature normalization makes it easier to select a step size that allows gradient descent to converge quickly.

Solution: True.

Problem 6. (10 points) SVMs: margin and support vectors



bad best

The figure above shows a sample data set in two dimensions. The plus signs represent positive training examples ($y^{(i)} = +1$) and the circles represent negative training examples ($y^{(i)} = -1$).

There are many different linear separators for this data. Illustrate a bad separator (one with small margin) on the left plot, and a good separator (the one with the biggest possible margin) on the right plot. For each of the two cases, do the following:

- Draw the decision boundary $\mathbf{w}^T\mathbf{x} + b = 0$
- Draw and label the contours $\mathbf{w}^T\mathbf{x} + b = +1$ and $\mathbf{w}^T\mathbf{x} + b = -1$
- Circle the support vectors

Problem 7. (8 points) Feature Expansion and Kernel Trick

You have a data set with 100 labeled examples and $n = 2$ features, which you choose to split into a training set of 50 points and a test set of 50 points. You train a linear regression model of the form $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$ by minimzing the the squared error cost function without regularization:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{50} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2.$$

The learned parameters $\boldsymbol{\theta}$ achieve cost $J(\boldsymbol{\theta}) = 29$ on the training set.

(2 points) You evaluate the cost of the learned parameters on the test set and get $J_{\text{test}}(\boldsymbol{\theta}) = 31$. Is this possible? Circle one:

Yes / No

Solution: Yes, this is possible. The cost on the test set is generally higher than the cost on the training set.

(2 points) Instead of training on the original data $\mathbf{x} = [x_1, x_2]^T$, you use a feature expansion

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix},$$

which includes all original features and all products of original features. When you retrain the model you find that the training cost $J(\boldsymbol{\theta})$ has increased from 29 to 45. Is this possible? Circle one:

Yes / No

Solution: No, this is not possible. We have only *added* new features, so the training error cannot go up. (We could always set the weights on the new features to zero and keep the old model.

(2 points) This time, you try a different feature expansion $\phi(\mathbf{x})$ and the training cost goes down to 20. You decide to continue exploring different models and fit a kernelized linear regression model using the kernel $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$ corresponding to the feature expansion you just tried. You find that now your training cost goes down even further to 15. Is this possible?

Yes / No

Solution: No, this is not possible. Using a kernel is equivalent to the feature expansion. It does not change the hypothesis class.

(2 points) True or false? The kernel trick allows us to fit models corresponding to complex feature expansions without actually having to do the feature expansion.

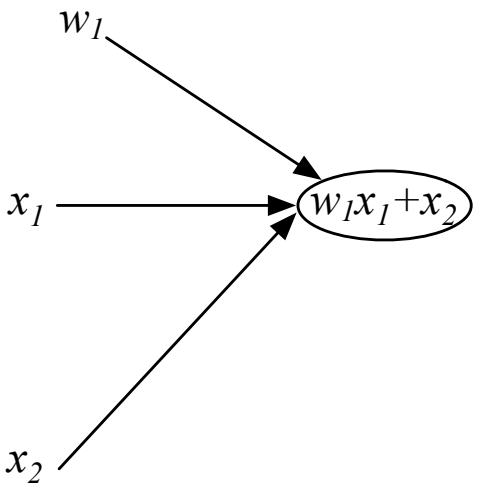
Yes / No

Solution: Yes.

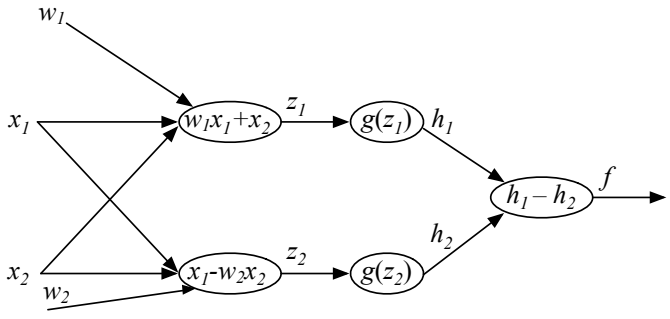
Problem 8. (12 points) Backprop

Consider the function $f(x_1, x_2, w_1, w_2) = g(w_1x_1 + x_2) - g(x_1 - w_2x_2)$ where $g(z) = \text{ReLU}(z)$ is the rectified linear function. (This function is a neural network with one hidden layer of two units, and all weights but two fixed to convenient values like 1 or -1 so they are easy to work with.)

(2 points) Complete the drawing of the computation graph below. Label the output of each node with the name of the intermediate variable it computes. To make sure your names are consistent with the next part of the question, use the variable h_1 to represent the value of the expression $g(w_1x_1 + x_2)$ (this is the value of the first hidden unit in the neural net).



Solution:



(3 points) Write pseudocode for forward propagation to compute the value of f given particular settings of the input variables w_1, w_2, x_1, x_2. Follow the exact computation (with the same variable names) implied by your computation graph. You may assume the availability of a function `ReLU` to compute the logistic function.

Solution:

```
z1 = w1*x1 + x2
z2 = x1 - w2*x2
h1 = ReLU(z1)
h2 = ReLU(z2)
f = h1 - h2
```

(5 points) Now, write pseudocode to compute the partial derivatives $\frac{\partial f}{\partial w_1}$, $\frac{\partial f}{\partial w_2}$, $\frac{\partial f}{\partial x_1}$, and $\frac{\partial f}{\partial x_2}$. Assume that your forward propagation code has already run to populate the values of all intermediate variables. (Hint: remember that if a variable feeds into two more than one computation node, its partial derivative is the sum of the backpropagated contribution from each of those nodes.)

Solution:

```
f_bar = 1 h1_bar = 1 * f_bar
h2_bar = -1 * f_bar
z1_bar = 1{h1 ≥ 0}* h1_bar
z2_bar = 1{h2 ≥ 0} * h2_bar
w1_bar = x1 * z1_bar
w2_bar = -x2 * z2_bar
x1_bar = w1* z1_bar + z2_bar
x2_bar = z1_bar - w2*z2_bar
```

(2 points) Now, suppose the output value of f is used within a loss function $L(f, y)$ where y is the desired output, and that the value of $\frac{\partial L}{\partial f}$ is available and stored in the variable `f_bar`. Decribe precisely what needs to change in your backprop code to compute the partial derivatives of L (instead of f) with respect to all other variables.

Solution: At the start of backpropagation compute $L(f, y)$ and set `f_bar` equal to $\frac{dL}{df}$ instead of 1.

Problem 9. (10 points) L1 regularization. **This question is for ECE697 students, extra credit for ECE597ML students**

In this problem you will derive the gradient descent learning update for a cost function with "L1-regularization". This is similar to the regularization we saw in class except the cost function penalizes the absolute value of parameter values instead of the square of parameter values.

L1-regularized cost function:

$$J(\boldsymbol{\theta}) = \lambda \sum_{k=1}^{n} |\theta_k| + J_{\text{orig}}(\boldsymbol{\theta})$$

To keep things simple, we will leave things in terms of the original (unregularized) cost function $J_{\text{orig}}(\boldsymbol{\theta})$. (Your answer should include the term $\frac{\partial}{\partial \theta_j} J_{\text{orig}}(\boldsymbol{\theta})$.)

Write the gradient descent update rule for θ_j, $j \geq 1$ below. Use the fact that $\frac{d}{dz}|z| = 1$ if $z > 0$ and $\frac{d}{dz}|z| = -1$ if $z < 0$. Assume that θ_j will never become exactly zero during the execution of the algorithm, so you do not need to worry about the update rule when $\theta_j = 0$.

(a) (7 points) Derive the udpate rule:

Solution:

$$\begin{aligned}
\theta_j &= \theta_j - \alpha \frac{\partial}{\partial \theta_j} \big(\lambda \sum_{k=1}^{n} |\theta_k| + J_{\text{orig}}(\boldsymbol{\theta})\big) \\
&= \theta_j - \alpha\lambda \sum_{k=1}^{n} \frac{\partial}{\partial \theta_j} |\theta_k| - \alpha \frac{\partial}{\partial \theta_j} J_{\text{orig}}(\boldsymbol{\theta}) \\
&= \theta_j - \alpha\lambda \frac{\partial}{\partial \theta_j} |\theta_j| - \alpha \frac{\partial}{\partial \theta_j} J_{\text{orig}}(\boldsymbol{\theta}) \\
&= \begin{cases} \theta_j - \alpha\lambda - \alpha \frac{\partial}{\partial \theta_j} J_{\text{orig}}(\boldsymbol{\theta}) & \text{if } \theta_j > 0 \\ \theta_j + \alpha\lambda - \alpha \frac{\partial}{\partial \theta_j} J_{\text{orig}}(\boldsymbol{\theta}) & \text{if } \theta_j < 0 \end{cases}
\end{aligned}$$

(b) (3 points) Write one or two sentences to interpret the update rule and explain how it encourages parameters to be close to zero.

Solution: The update first adds or subtracts the positive amount $\alpha\lambda$ from θ_j. If θ_j is positive, it subtracts $\alpha\lambda$. If θ_j is negative, it adds $\alpha\lambda$. In either case, this moves θ_j closer to zero (as long as $\alpha\lambda$ is not too big!) before taking a gradient step with respect to the original cost function J_{orig}.