Regularization for linear models

Practical week 3

Ridge linear regression

Cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) + \frac{\alpha}{2m}\boldsymbol{\theta}^\top\boldsymbol{\theta}$$

- 1. Compare this cost function to the cost function used in "standard" linear regression.
- 2. What is the extra term that is being added to the cost function called?
- 3. What effect will adding this extra term have on the solution, i.e. the parameters theta?
- 4. What can be reasons to choose ridge regression over linear regression?

Ridge linear regression

Cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\boxed{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})} + \boxed{\frac{\alpha}{2m}\boldsymbol{\theta}^\top\boldsymbol{\theta}}$$

- Same cost function as regular linear regression
- Ridge parameter that will reduce values for theta
- Divide by 2m so function does not depend on the number of samples

Ridge linear regression

Gradient of ridge regression loss:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}(\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} - \mathbf{X}^\top \mathbf{y}) + \frac{\alpha}{m}\boldsymbol{\theta}$$

Set gradient to zero, and define the least squares solution for Θ:

$$\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} - \mathbf{X}^\top \mathbf{y} + \alpha \mathbf{I}\boldsymbol{\theta} = 0$$

$$(\mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I})\boldsymbol{\theta} = \mathbf{X}^\top \mathbf{y}$$

$$\boldsymbol{\theta} = (\mathbf{X}^\top \mathbf{X} + \underbrace{\alpha \mathbf{I}}_{\text{the "ridge"}})^{-1} \mathbf{X}^\top \mathbf{y}$$

Exercise 3.1

```python
def _solve_normal(self, X, y):
    # START EXERCISE 3.1 #
    #'''

    X_b = augment_matrix(X)
    n_samples, n_features = X_b.shape

    # # solution with the bias included in the regularization:
    A = X_b.T.dot(X_b) + self.alpha * np.eye(X_b.shape[1])
    # solution with the bias excluded from the regularization:
    A = X_b.T.dot(X_b) + self.alpha * np.diag([0] + [1] * (n_features - 1))

    b = X_b.T.dot(y)
    self.theta = np.linalg.inv(A).dot(b)
    #'''
    # END EXERCISE 3.1 #

    return self.theta
```

$$\boldsymbol{\theta} = \boxed{(\mathbf{X}^\top\mathbf{X} + \underbrace{\alpha\mathbf{I}}_{\text{the "ridge"}})}^{\boxed{-1}}\boxed{\mathbf{X}^\top\mathbf{y}}$$

Ridge linear regression

Gradient descent for ridge regression loss:

Same gradient as before

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}(\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} - \mathbf{X}^\top \mathbf{y}) + \frac{\alpha}{m}\boldsymbol{\theta}$$

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \left(\frac{1}{m}\mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) + \frac{\alpha}{m}\boldsymbol{\theta} \right)$$

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} \underbrace{- \eta \frac{1}{m}\mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\text{same as l.r.}} \underbrace{- \eta \frac{\alpha}{m}\boldsymbol{\theta}}_{\text{"shrinkage"}}$$

Exercise 3.2

```python
def _solve_gradient_descent(self, X, y, n_epochs, learning_rate):

    X_b = augment_matrix(X)
    n_samples, n_features = X_b.shape
    self.theta = np.random.randn(n_features)

    # START EXERCISE 3.2 #
    #'''
    for _ in range(n_epochs):
        predictions = X_b.dot(self.theta)
        errors = predictions - y

        # solution with the bias included in the regularization:
        gradients = 1 / n_samples * X_b.T.dot(errors) + 2 * self.alpha / n_samples * self.theta
        # solution with the bias excluded from the regularization:
        gradients = 1 / n_samples * X_b.T.dot(errors)
        gradients[1:] += self.alpha / n_samples * self.theta[1:]

        self.theta -= learning_rate * gradients
    #'''
    # END EXERCISE 3.2 #

    return self.theta
```

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \boxed{\underbrace{\eta\frac{1}{m}\mathbf{X}^{\top}\boxed{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}}_{\text{same as l.r.}} \underbrace{-\eta\frac{\alpha}{m}\boldsymbol{\theta}}_{\text{"shrinkage"}}}$$

- *Compare the ridge model to the standard linear regression model. You'll notice that the MSE of ridge is higher. Does this mean the model does a poorer job when fitting the data?*

Ridge logistic regression

Cost function:

$$J(\boldsymbol{\theta}) = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]}_{\text{same as logistic regression}} + \underbrace{\frac{\alpha}{2m}\boldsymbol{\theta}^\top\boldsymbol{\theta}}_{\text{ridge penalty}}$$

Gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \underbrace{\frac{1}{m}\mathbf{X}^\top(\mathbf{p} - \mathbf{y})}_{\text{same as logistic regression}} + \frac{\alpha}{m}\boldsymbol{\theta}$$

Ridge logistic regression

The solution, which we implement in 4.1:

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \left(\frac{1}{m} \mathbf{X}^\top (\mathbf{p} - \mathbf{y}) + \frac{\alpha}{m} \boldsymbol{\theta} \right)$$

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \underbrace{\frac{1}{m} \mathbf{X}^\top (\mathbf{p} - \mathbf{y})}_{\text{same as logistic regression}} - \underbrace{\eta \frac{\alpha}{m} \boldsymbol{\theta}}_{\text{ridge penalty}}$$

Ridge logistic regression

- The difference between ridge and regular logistic regression is the implementation of penalties.
- **What types of penalties are there, what do the abbreviations mean and what is the effect of penalties (if any)?**

Penalties

- L1: Lasso: Sum of the absolute values of the coefficients.
- L2: Ridge: Sum of squared values of the coefficients.

Exercise 4.1

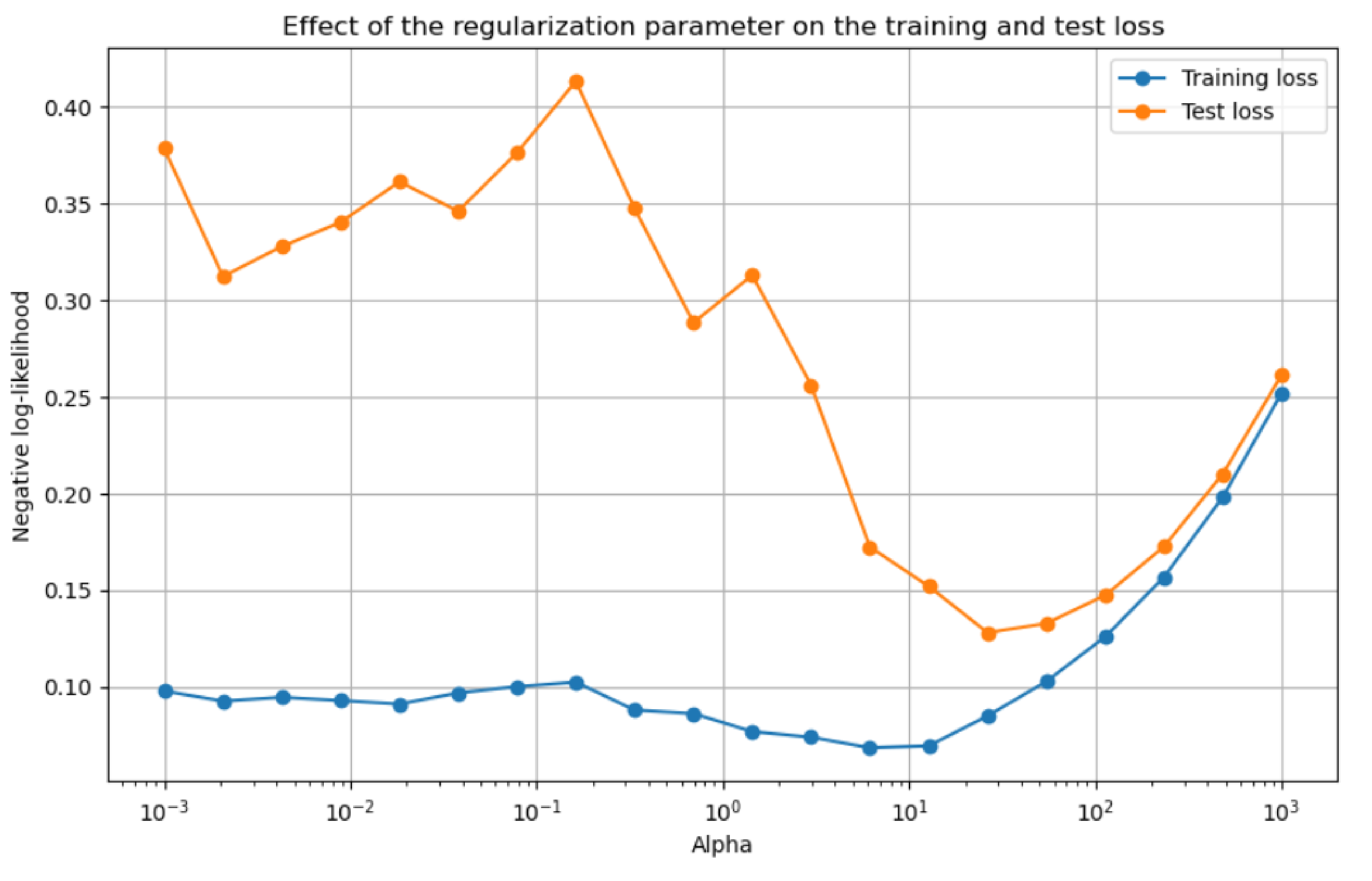
```python
def fit(self, X, y):
    X_b = augment_matrix(X)
    n_samples, n_features = X_b.shape
    self.theta = np.random.randn(n_features)

    # START EXERCISE 4.1 #
    #'''
    for _ in range(self.epochs):
        predictions = sigmoid(X_b.dot(self.theta))
        errors = predictions - y
        gradients = 1 / n_samples * X_b.T.dot(errors) + self.alpha / n_samples * self.theta
        # solution with the bias excluded from the regularization:
        #gradients = 1 / n_samples * X_b.T.dot(errors)
        #gradients[1:] += self.alpha / n_samples * self.theta[1:]
        self.theta -= self.learning_rate * gradients
    #'''
    # END EXERCISE 4.1 #
```

$$\boldsymbol{\theta}^{(\text{next})} = \boldsymbol{\theta}^{(\text{current})} - \eta \underbrace{\frac{1}{m}\mathbf{X}^\top(\mathbf{p}-\mathbf{y})}_{\text{same as logistic regression}} - \underbrace{\eta\frac{\alpha}{m}\boldsymbol{\theta}}_{\text{ridge penalty}}$$

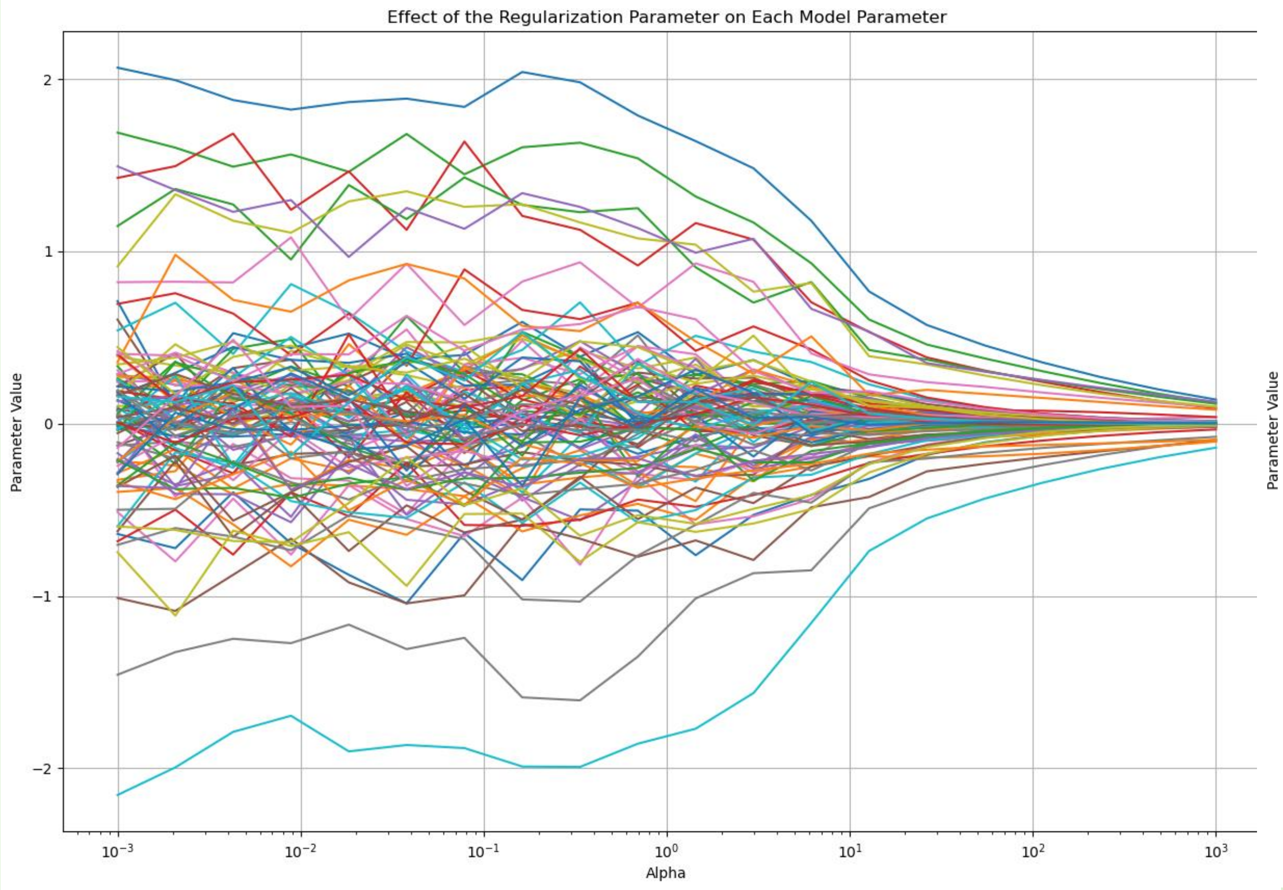
Influence of alpha

- *What value of α is optimal in this case and why?*
- *Having selected the optimal α, how would you progress?*
- *Explain the U-shaped curves- why do the train and test losses increase for large values of α?*



Lasso vs ridge

- Ridge or L2:



- Lasso or L1: