Training Models

Linear Regression

- A linear model predicts by computing a weighted sum of the input features plus a constant called the bias or intercept term.
- $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$
    - $\hat{y}$ is the predicted value.
    - n is the number of features.
    - x_i is the ith feature.
    - θ_j is the jth model parameter, including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \ldots, \theta_n$.
- The vectorized form of the equation is $\hat{y} = h_\theta(x) = \theta \cdot x$
    - hθ is the hypothesis function, using the model parameters θ.
    - θ is the model's parameter vector, containing the bias term θ0 and the feature weights θ1 to θn.
    - x is the instance's feature vector, containing x0 to xn, with x0 always equal to 1.
    - θ · x is the dot product of the vectors θ and x, which is equal to θ0x0 + θ1x1 + θ2x2 + ... + θnxn.
- We need to find the value of θ that minimizes the RMSE to train the linear regression model.
    - We can use the Normal Equation to find this value.
    - Through the **add_dummy_feature()** function and NumPy's **linalg.inv()** method, we can compute the best value of θ.
    - Note that in Python, the @ operator performs matrix multiplication.
    - Then, we can make predictions using this best value of θ.

Gradient Descent

- The general idea of gradient descent is to tweak parameters iteratively to minimize a cost function.
- The gradient descent measures the local gradient of the error function with regard to the parameter vector θ, and it goes in the direction of the descending gradient. Once the gradient is zero, you have reached a minimum.
    - We start by filling θ with random values.
    - Then, we improve θ gradually, taking one step at a time, to decrease the cost function until the algorithm converges to a minimum.

- The gradient descent is the size of the steps, controlled by the learning rate hyperparameter.
    - If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time.
    - On the other hand, if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before.
- Sometimes, the cost function will not be bowl-shaped. In these cases, the function may have local and global minimums.
    - The MSE cost function for a linear regression model is a convex function, which means that if you pick any two points on the curve, the line segment joining them is never below the curve.

Batch Gradient Descent

- We need to calculate how much the cost function will change if we change θj just a little bit to implement a gradient of the cost function. This is called a partial derivative.
- We can use the learning rate η to determine the size of each step.
- To find a good learning rate, we can use a grid search but limit the number of epochs so that the grid search does not take too long to converge.
    - Each iteration over the training set is called an epoch.
- To find a good number of epochs, we can set a very large number of epochs but interrupt the algorithm when the gradient vector becomes small.
    - This is handled by ϵ, the tolerance.

Stochastic Gradient Descent

- Stochastic gradient descent picks a random instance in the training set at every step and computes the gradients based only on that single instance.
- Due to its stochastic (i.e., random) nature, this algorithm is much less regular than batch gradient descent.
    - Instead of gently decreasing until it reaches the minimum, the cost function will bounce up and down, decreasing only on average
    - Randomness is good to escape from local optima but bad because the algorithm can never settle at the minimum.
    - To solve this problem, we gradually reduce the learning rate.
    - The function that determines the learning rate at each iteration is called the learning schedule.
    - We cannot reduce the learning rate too early or too late.
- To perform linear regression using stochastic GD with Scikit-Learn, we can use the **SGDRegressor()** class.
    - The *max_iter* parameter controls the number of epochs.
    - The *tol* parameter controls the value at which the algorithm should stop.
    - The *eta0* parameter controls the learning rate.
    - The *penalty* parameter controls the amount of regularization.

Polynomial Regression

- We can use a linear model to fit nonlinear data.
    - We add powers of each feature as new features, then train a linear model on this extended set of features.
    - We can add powers using the **PolynomialFeatures()** class and build a linear regression model with **LinearRegression()**.

Learning Curves

- We can look at the learning curves, which are plots of the model's training error and validation error as a function of the training iteration.
    - We evaluate the model at regular intervals during training on both the training set and the validation set and plot the results.
    - We can simply use the **learning_curve()** function together with Pyplot to visualize the learning curves.
- The training error curve fits the data well during the first few instances but starts to perform worse as new instances are added until it hits a plateau.
- In contrast, the validation error curve cannot generalize properly during the first few instances, but it gradually improves until it hits a plateau.

The Bias/Variance Trade-Off

- A model's generalization error can be expressed as the sum of three different errors.
    - Bias, which is due to wrong assumptions.
    - Variance, which is due to the model's excessive sensitivity to small variations in the data, such as a high-degree polynomial model.
    - Irreducible error, which is due to noise in the data.
- Increasing a model's complexity will typically increase its variance and reduce its bias. On the other hand, decreasing the complexity will increase the bias but reduce variance.

Regularized Linear Models

- A good way to reduce overfitting is to regularize the model (i.e., to constrain it): the fewer degrees of freedom it has, the harder it will be for it to overfit the data.
- For a polynomial model, we could simply reduce the number of polynomial degrees.
- For a linear model, regularization is typically achieved by constraining the weights of the model.

Ridge Regression

- Ridge regression is a regularized version of a linear regression.
    - A regularization term is added to the MSE, forcing the learning algorithm to not only fit the data but also keep the model weights as small as possible.
    - Note that the regularization term should only be added to the cost function during training.
- The hyperparameter α controls how much you want to regularize the model.

    - If α = 0, the ridge regression is just linear regression.
    - If α is very large, all weights end up close to zero, resulting in a flat line through the data's mean.
    - Note that it is important to use a scaler to scale the data before performing ridge regression.
    - Note how increasing α leads to flatter (i.e., less extreme, more reasonable) predictions, thus reducing the model's variance but increasing its bias.
    - We can use the **Ridge()** class together with a stochastic gradient descent.

Lasso Regression

- Lasso stands for Least Absolute Shrinkage and Selection Operator.
- Lasso regression is another regularized version of linear regression.
    - Just like ridge regression, it adds a regularization term to the cost function, but it uses the ℓ1 norm of the weight vector instead of the square of the ℓ2 norm.
    - An important characteristic of lasso regression is that it tends to eliminate the weights of the least important features (i.e., set them to zero).
    - We can simply use the **Lasso()** class.

Elastic Net Regression

- Elastic net regression is a middle ground between ridge regression and lasso regression.
    - The regularization term is a weighted sum of both ridge and lasso's regularization terms, and you can control the mix ratio r.
    - If r = 0, the elastic net is equivalent to ridge regression.
    - If r = 1, the elastic net is equivalent to lasso regression.
    - We can simply use the **ElasticNet()** class.
- How do we know which form of regression to use?
    - It is almost always preferable to have at least a little bit of regularization, so generally you should avoid plain linear regression
    - Ridge regression is a good default.
    - If the data has only a few features that are useful, then use lasso or elastic net regression.
    - Generally, elastic net regression is preferred over lasso because it behaves less erratically with a larger number of features.

Early Stopping

- A very different way to regularize iterative learning algorithms such as gradient descent is to stop training as soon as the validation error reaches a minimum
- We can use early stopping to prevent the model from overfitting the data once the validation error reaches its minimum.

Logistic Regression

- Logistic regression (also called logit regression) is commonly used to estimate the probability that an instance belongs to a particular class.

- If we're dealing with a binary classifier, then if the estimated probability is greater than the threshold (usually 50%), the model assigns the instance to the positive class and to the negative class otherwise.
    - Generally, y = 0 if the estimated possibility < 0.5,
      y = 1, otherwise.
- Just like a linear regression model, a logistic regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly, it outputs the logistic of this result.
    - Logistic regression models use a sigmoid function, which is S-shaped.
- The objective of training is to set the parameter vector θ so that the model estimates high probabilities for positive instances (y = 1) and low probabilities for negative instances (y = 0).
- We can use the **LogisticRegression()** class to create a logistic regression model.

Softmax Regression

- The logistic regression model can be generalized to support multiple classes directly, without having to train and combine multiple binary classifiers.
    - In this case, the threshold might not be 50%. If there are 3 classes, the thresholds might be 33% and 66%.
- When given an instance x, the softmax regression model first computes a score sk(x) for each class k, then estimates the probability of each class by applying the softmax function (also called the normalized exponential) to the scores.
- As always, we can use the **argmax()** function to get the value of a variable that maximizes a function.
- Cross entropy is frequently used to measure how well a set of estimated class probabilities matches the target classes.
    - We can use cross entropy to minimize the cost function.