**Project 2. Basic Classifier and Regressor**

**Major Tasks:**

1. Learn more about how to utilize data sets and Python libraries;
2. Learn the basics of classifiers and regressors;
3. Improve Python programming and report writing skills.

**Part 1. Classification:**

* 1. Run the example program code and fix any problems you encounter;

Q3.2. classify digit zero and non-zero for the binary classifier, instead five and non-five in the example. Show your results.

Q3.3. What are the performance metrics of binary classifiers and how to interpret them? Use the zero/non-zero example to explain those metrics.

Measuring Accuracy Using Cross-Validatio (“Implementing Cross-Validation”)

Confusion Matrix

Precision and Recall (tradeoff threshold)

The ROC Curve (receiver operating characteristic)

(sensitivity (recall) versus 1 – specificity.)

area under the curve (AUC). Perfect: ROC AUC = 1

Q3.4. What are the differences between these concepts: multi-class, multi-label, and multi-output classification?

multi-class : distinguish between more than two classes

multi-label : outputs multiple binary labels

multi-output: generalization of multilabel classification where each label can be multiclass

* 1. Work through Exercise 1 and 2 in Chapter 3. Answer questions Q3.5.

Q3.5. In Exercise 2, what are the other ways to alter the image data set?

* 1. For ECE450 students: Images may be rotated, shrunk, filtered, noise added, or blurred. Choose another one or two methods to argument the data set and train the classifier again; Report your results.

Rotate. **Results?**

Q3.6. What did you learn most in Part 1 of Project 2?

Basic theoretical concepts of classifier

**Part 2. Regressor:**

2.1. Follow the instructions of chapter 4 in the Hands-on ML book and work through the basic project.

Q4.1. There is some error in the textbook and the error is noted in the notebook file. What is the error? How is pseudoinverse calculated differently from the Normal Equation?

the first releases of the book implied that the LinearRegression class was based on the Normal Equation. This was an error. It is based on the pseudoinverse, which ultimately relies on the SVD matrix decomposition of **𝐗**. Its time complexity is O(n2) and it works even when m<n or when some features are linear combinations of other features (in these cases, XTX is not invertible so the Normal Equation fails), see [issue #184](https://github.com/ageron/handson-ml/issues/184) for more details.

(However, this does not change the rest of the description of the LinearRegression class, in particular, it is based on an **analytical solution**, it does **not scale well** with the number of features, it scales linearly with the number of instances, all the data must fit in memory, it does **not require feature scaling** and the order of the instances in the training set does not matter.)

2.2. There are three gradient descent algorithms in the example: batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. When you work through these examples, pay close attention to the results. You will need those results to be compared with the ones you get in Step 2.3.

2.3 Change the training data set in cell [2] from *X = 2 \* np.random.rand(100, 1)* to *X = 4 \* np.random.rand(100, 1).* Restart the kernel to run through the program. Change the corresponding plt.axis parameters to plot the correct figures.

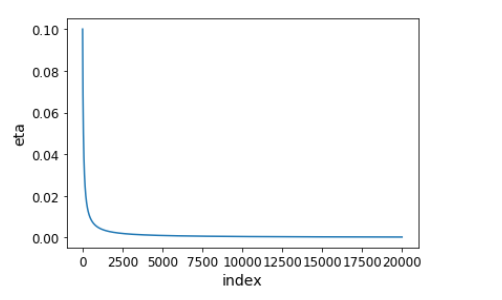
Q4.2. Do you observe any difference in the results when changing the inputs? If so, include them in your report. Explain why they are different.

Theta[1] becomes bigger. Don’t know why

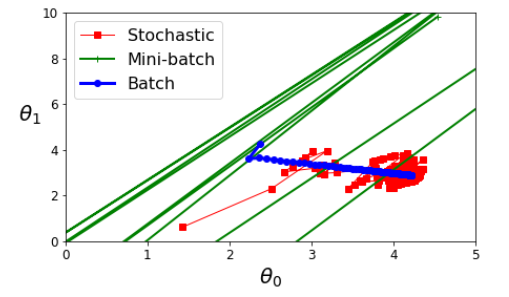
Notice BGD’s result is always the same as normal equation, but not SGD and mini-BGD. Coincidence?

Q4.4. In the stochastic gradient descent algorithm, how is the learning rate eta changed? Try to plot the value eta vs. the iteration index (epoch\*m+i) for the epoch and i values used in the example.

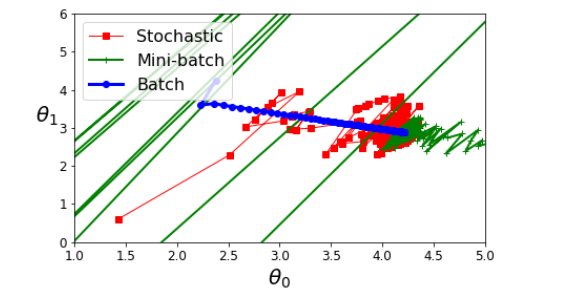
In a exponential way –



Q4.5. In the mini-batch gradient descent algorithm, how is the results different from the original example? Do you observe any problems? If so, how to fix the problems?



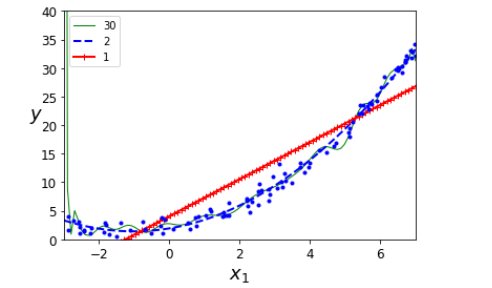
Don’t seem to convergence when iteration is 50



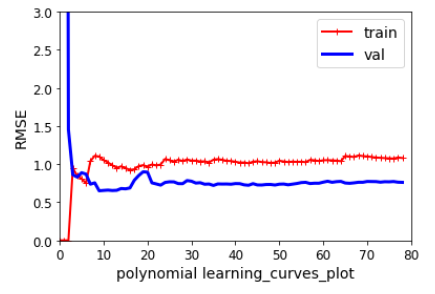
Seem to convergence when iteration is 300

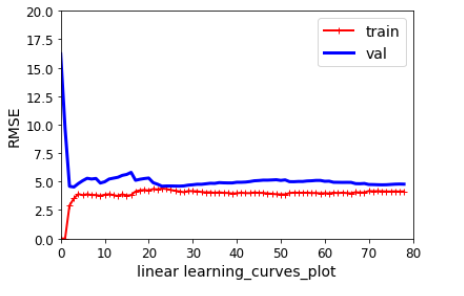
2.4. In the polynomial regression example, change the training data to *X = 10 \* np.random.rand(m, 1) – 3* and change the prediction range of X\_new. Change the overfitting degree from 300 to 30. Run the example to plot the figure that compares the three models. Adjust the plt.axis if needed. Comment on how the models fit the data and how well they can predict using the learning curves.

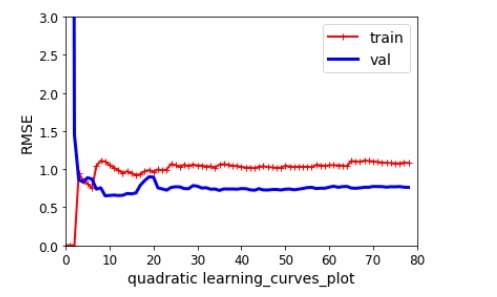
Comparison:



How well:







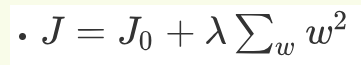
Linear model fits bad.

2.5 For ECE 450 students, briefly describe the five algorithms: stochastic gradient descent with L1 penalty, stochastic gradient descent with L2 penalty, ridge regression, LASSO regression, and elastic net.

stochastic gradient descent with L1 penalty,

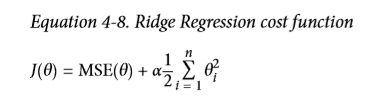


stochastic gradient descent with L2 penalty,



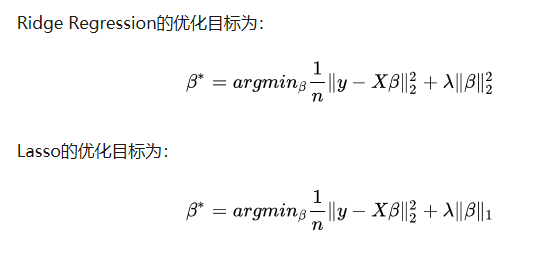
ridge regression,

L2 penalty



LASSO regression,

L1 penalty



上面的图从几何意义上解释了L1与L2正则化的区别，同时这也解释了L1与L2最大的不同：L1可以带来稀疏的结果，即L1会使得部分参数为零。这样的好处是什么呢？一方面，可以用来选择特征，一方面可以用来降维压缩数据等等。

那么，介绍到这里，L1正则化总是和稀疏挂钩，那么L2正则化呢，L2正则化做了什么事情？其实，和L2正则化挂钩的则是Weight Decay(权值衰减)。下面来简单说一下，考虑一般的优化问题：

  
利用梯度下降来求解问题，得到：



所以可以看到，第t+1步的参数在第t步的参数前乘以了 ，所以会使得权重趋向于零，即Weight Decay的过程。

elastic net

**ElasticNet综合了L1正则化项和L2正则化项**

ElasticNet在我们发现用Lasso回归太过(太多特征被稀疏为0),而岭回归也正则化的不够(回归系数衰减太慢)的时候，可以考虑使用ElasticNet回归来综合，得到比较好的结果。

2.6. Work through the logistic regression and softmax regression examples. Explain how regression and classification are related.

As we discussed in Chapter 1, some regression algorithms can be used for classification as well (and vice versa). Logistic Regression (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?). If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does not (i.e., it belongs to the negative class, labeled “0”). This makes it a binary classifier

**Lab Report and Submission: See report guideline for detailed requirements.**