Advanced Data Analysis

DATA 71200

Class 5: k-Nearest Neighbors and Linear Models

Schedule

13-Jun	**Supervised Learning (k-Nearest Neighbors and Linear Models)**
14-Jun	Async: DataCamp
15-Jun	Supervised Learning (Naive Bayes Classifiers and Decision Trees, Support Vector Machines, and Uncertainty estimates from Classifiers)
16-Jun	Async: DataCamp

Project 2 (Due June 21)

Project 2

Application of two supervised learning techniques on the dataset you created in Project 1. This assignment should be completed as a Jupyter notebook in your GitHub repository.

To **submit** the assignment, submit a link to your project Jupyter notebook on Blackboard.

The **goal** for this assignment is to apply different types of supervised learning algorithms with a range of parameter settings to your data and to observe which performs better.

Step 1: Load your data, including testing/training split from Project 1.

- Your testing and training split should be balanced
- Your data should be clean and missing data should be addressed

Step 2: (If not already done in Project 1) Prepare your data

- Make sure that your all appropriate variables are converted to categorical variables (as ordinal or one hot)
- Perform any necessary feature scaling

Step 3: Examine your target attribute. Based on the data exploration you did in Project 1, confirm and examine the attribute you are going to predict.

- Examine and plot the distribution of the target attribute in your training set (e.g., is it Gaussian, uniform, logarithmic). This will help you interpret the performance of different algorithms on your data.

Step 4: Selected two of the following supervised learning algorithms, ideally one from the first half of the list and one from the second half of the list

- K-Nearest Neighbor
- Linear Models
- Naïve Bayes
- Decision Trees
    - Single tree
    - Random Forest
    - Gradient Descent decision trees
- Support Vectors Machines

Step 5: For each of your selected models

- Run with the default parameters using cross-validation
    - Calculate precision, recall, and F1 for classification
    - Calculate r^2, RMSE, and MAE for regression
- (Where possible) adjust 2-3 parameters for each model
    - Report evaluation metrics for the best and worst-performing parameter settings

Tip: You should make notes on what worked well and what didn't. Such notes will be useful when you write up the paper for your final project.

Supervised Learning

- **Learning paradigm where we have example input/output pairs to learn from**
- **Classification predicts a class label**
  - Binary example: is this email spam? yes/no
  - Multiclass example: what is the species of this flower?
- **Regression predicts a *continuous* number**
  - Example: what is the value of a house given a set of features?

Generalization

- Generalization - a model's ability to make accurate predictions on unseen data
- Typically we want to find the simplest effective model
- Model complexity
  - Underfitting - when a model is too simple to represent the training data
  - Overfitting - when a model is too specific to the training data to generalize to new data

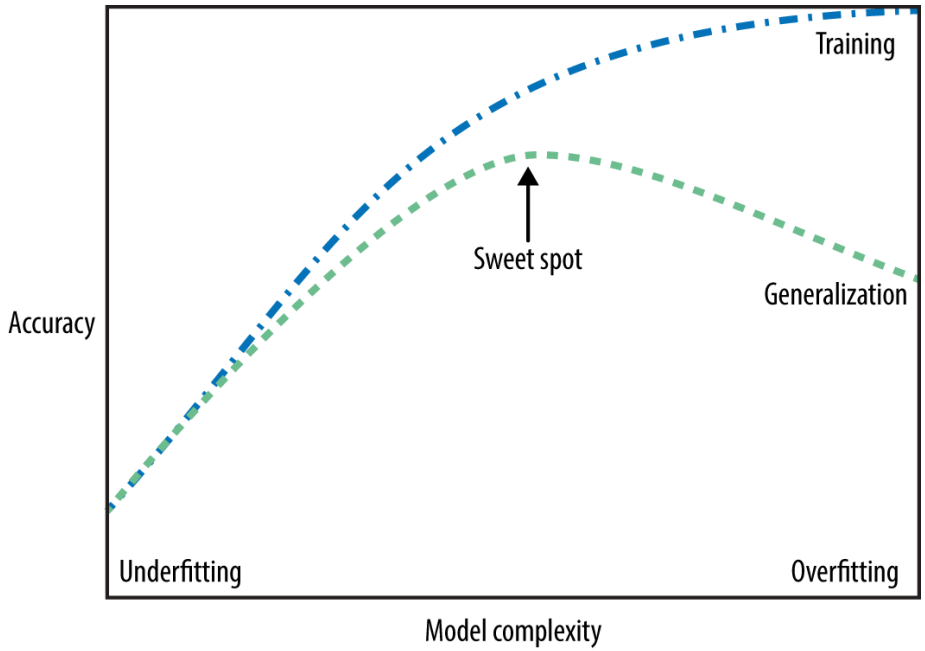


Figure 2-1. Trade-off of model complexity against training and test accuracy

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

k-Nearest Neighbors Classification

- Classification of unlabeled points based on the closest labeled point(s)
- *k* is the number of points considered to determine the class of an unlabeled point
- When *k* = 1 the class of the nearest point

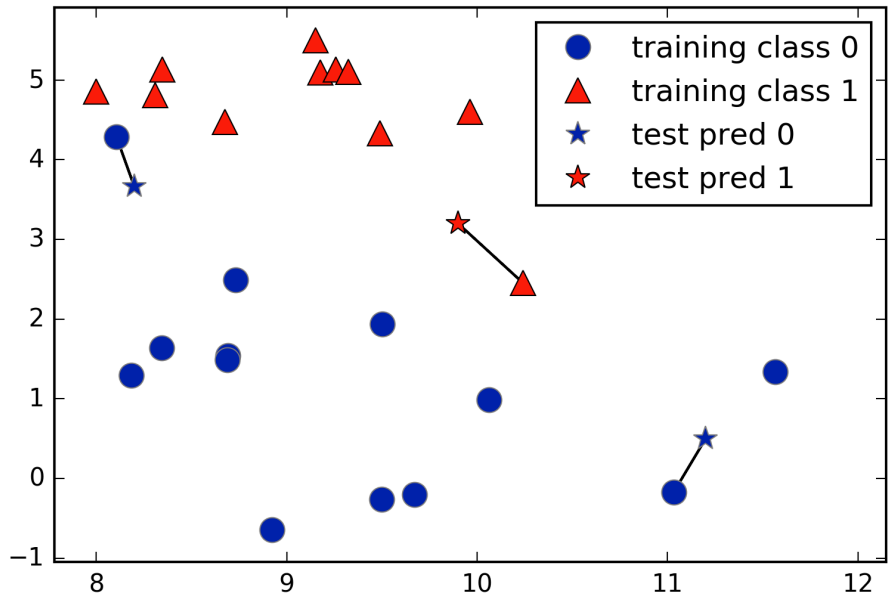


Figure 2-4. Predictions made by the one-nearest-neighbor model on the forge dataset

Jupyter Notebook 02-supervised-learning .ipynb [10]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

k-Nearest Neighbors Classification

- **When k > 1 the algorithm uses a simple voting paradigm**

  - Assigned class label is the most frequent label among the neighbors
  - *k* should be larger than the number of classes
  - *k* should not be a multiple of the number of classes

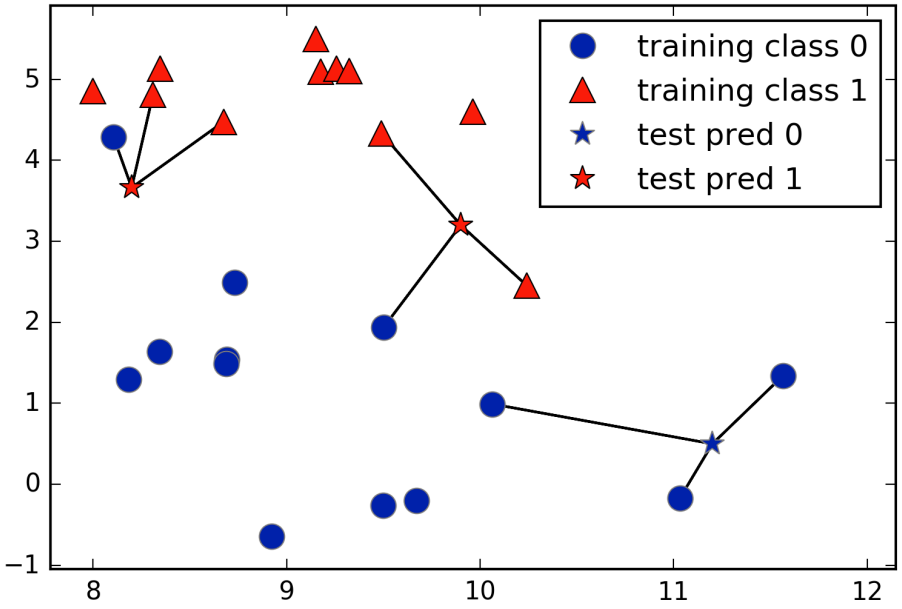


Figure 2-5. Predictions made by the three-nearest-neighbors model on the forge dataset

Jupyter Notebook 02-supervised-learning .ipynb [11–16]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Size of *k*

- **When *k* is too small the model is too complex and tends to overfit**
- **When *k* is too large the model is too simple and tends to underfit**

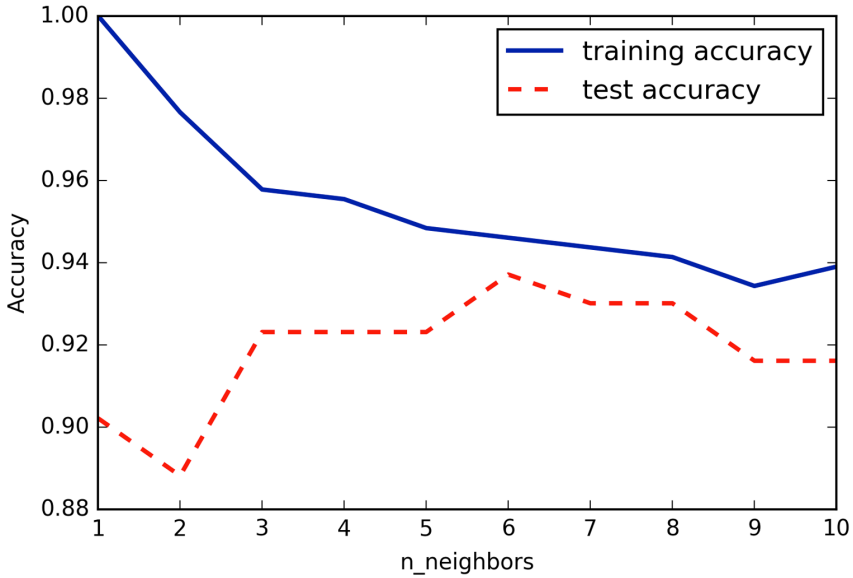


Figure 2-7. Comparison of training and test accuracy as a function of n_neighbors

Jupyter Notebook 02-supervised-learning .ipynb [18]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

k-Nearest Neighbors Overview

Jupyter Notebook
data71200class5a.ipynb

- **Assumptions**
  - *k*NNs do not make any assumptions about the distribution of the data
- **Parameters**
  - Number of neighbors
  - Distance measure (Euclidean, Manhattan, etc)
- **Strengths**
  - Easy to understand
  - Reasonable performance (at most twice as bad as the optimal classifier)
- **Weaknesses**
  - Prediction can be slow on large dataset
  - Often requires pre-processing
  - Performs poorly on datasets with a large number of features
  - Performs poorly on sparse datasets (where many values are 0)

k-Nearest Neighbors Activity

- **Import the wine dataset from Scikit-learn**
  - from sklearn.datasets import load_wine
  - data = load_wine()
- **Based on the code in data71200class5a.ipynb**
  - Split into a testing and training set
  - Import the k-nearest neighbor classifier
  - Select the best parameters using grid search
  - Report evaluation metrics for the best and worst performing parameters

Review

- **(Linear) Regression**
  - Continuous predictive model created by estimating a linear relationship between features and response
- **Logistic Regression**
  - Predictive model of the probability of a certain class

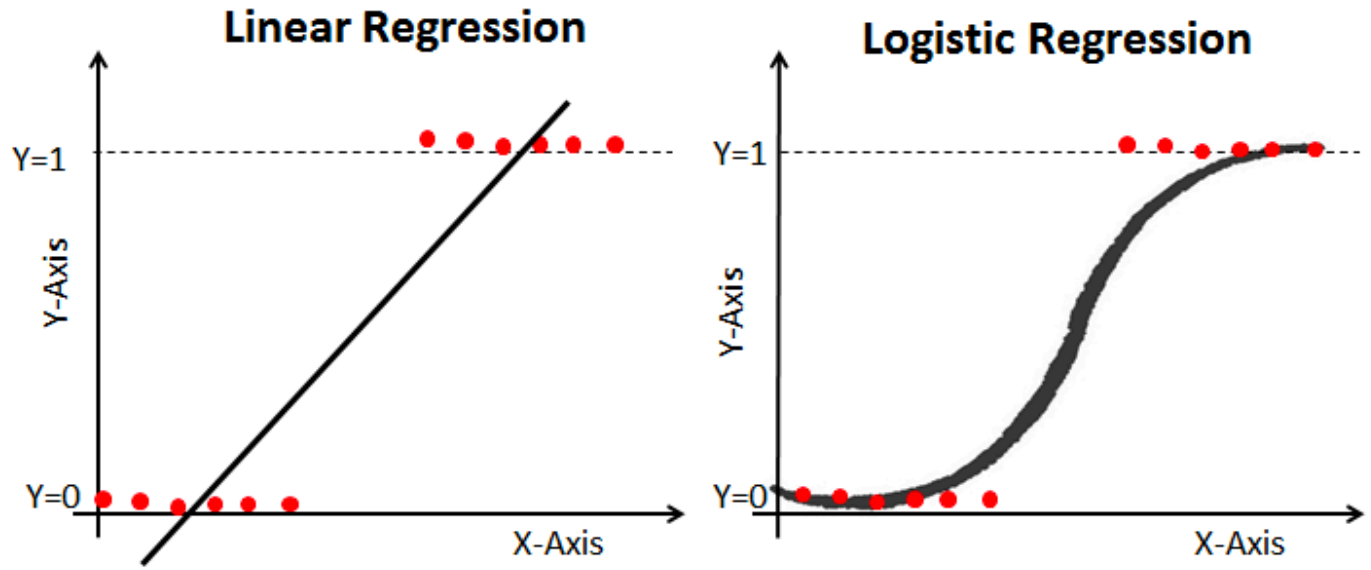
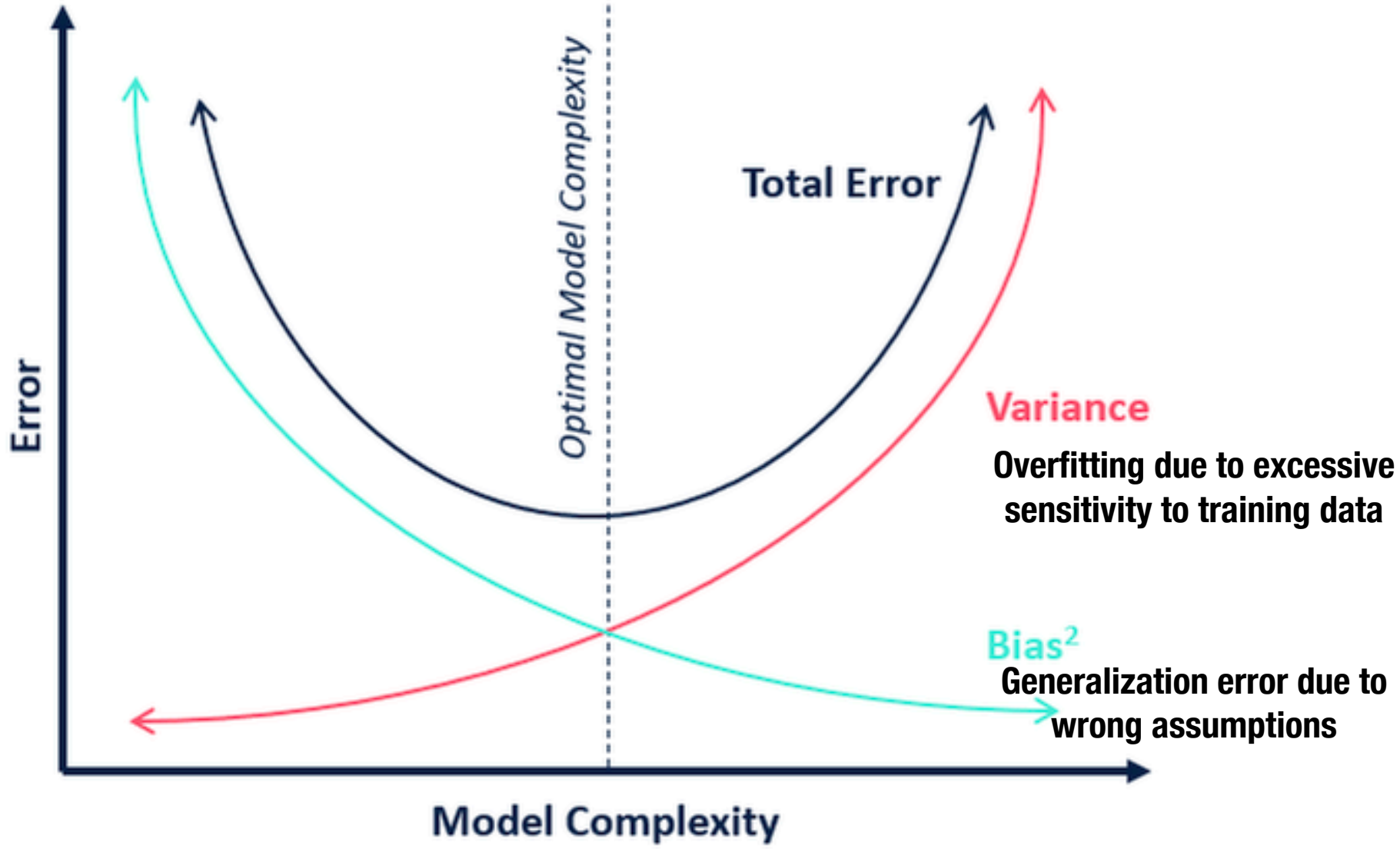


Image: https://medium.com/@hpsuresh12345/logistic-regression-60694a973bee

Bias/Variance Tradeoff



https://ai-pool.com/a/s/bias-variance-tradeoff-in-machine-learning

Ridge Regression

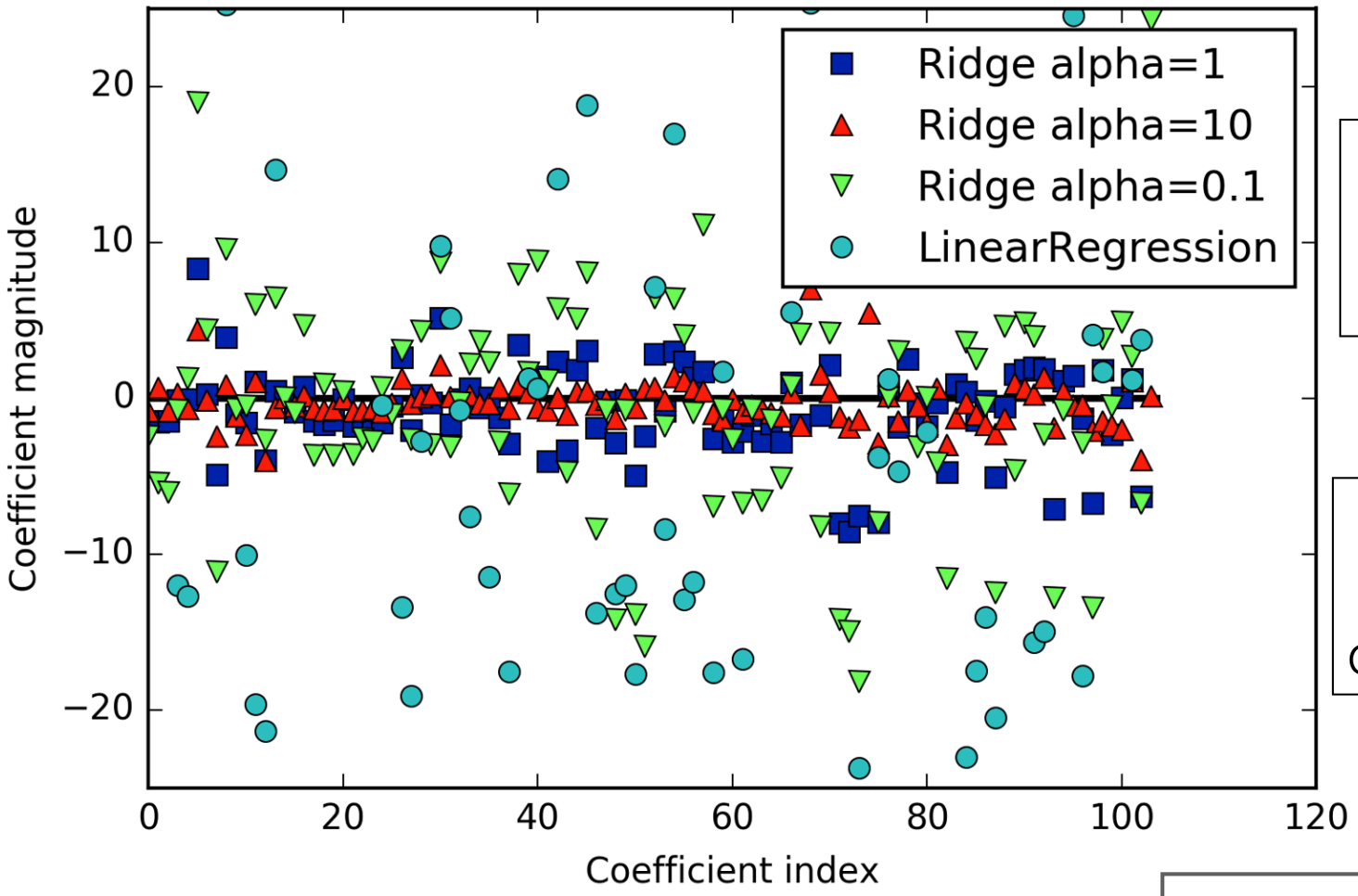


Figure 2-12. Comparing coefficient magnitudes for ridge regression with different values of alpha and linear regression

Trade off between model simplicity and model performance

Increasing alpha moves more coefficients towards 0

Jupyter Notebook 02-supervised-learning .ipynb [33]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Ridge Regression

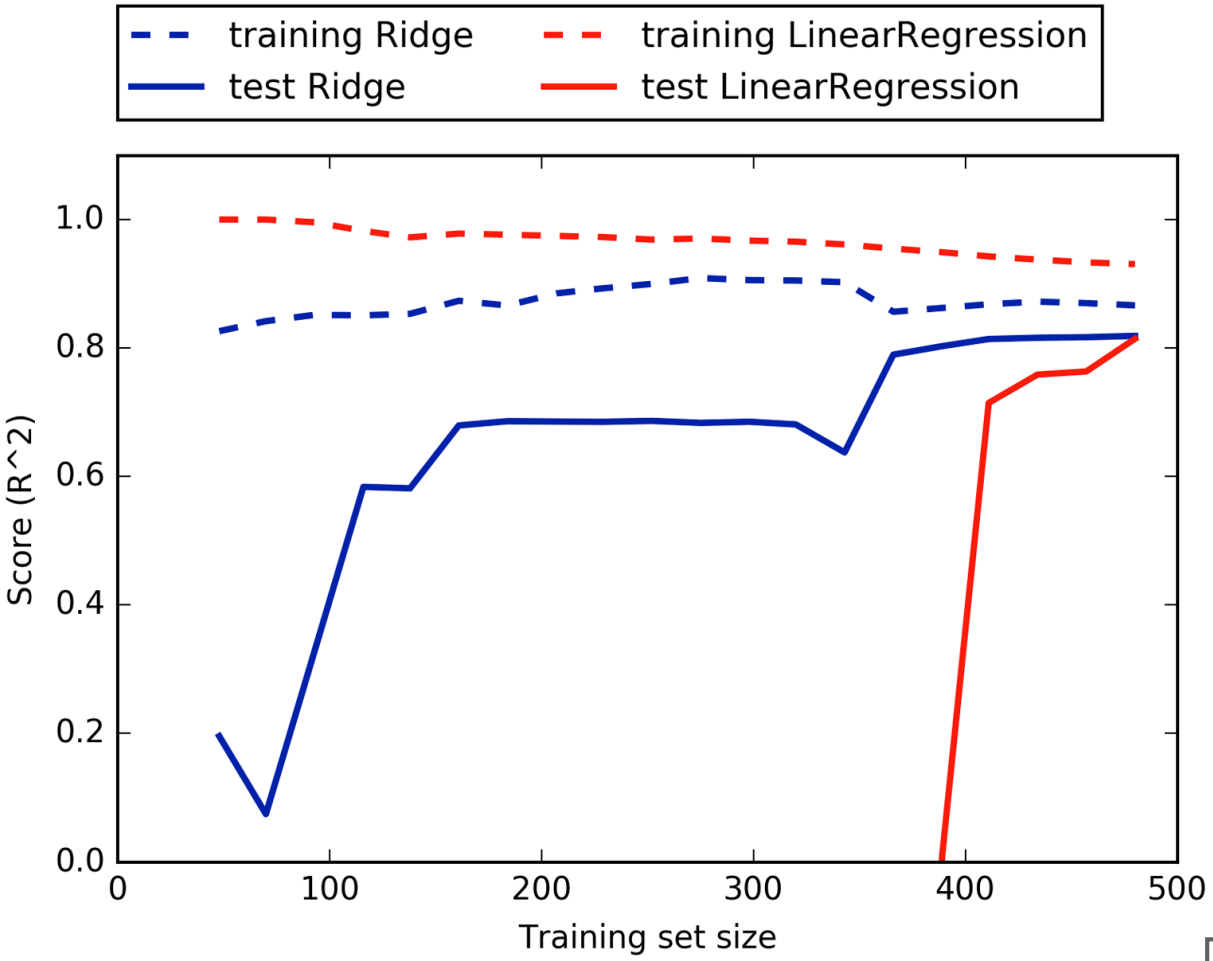


Figure 2-13. Learning curves for ridge regression and linear regression on the Boston Housing dataset

Ridge performs worse than OLS (MSE) on training set but better on testing set

OLS improves performance on testing set with more data (generalizes better so performance on training data declines)

Jupyter Notebook 02-supervised-learning .ipynb [34]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Lasso versus Ridge Regression

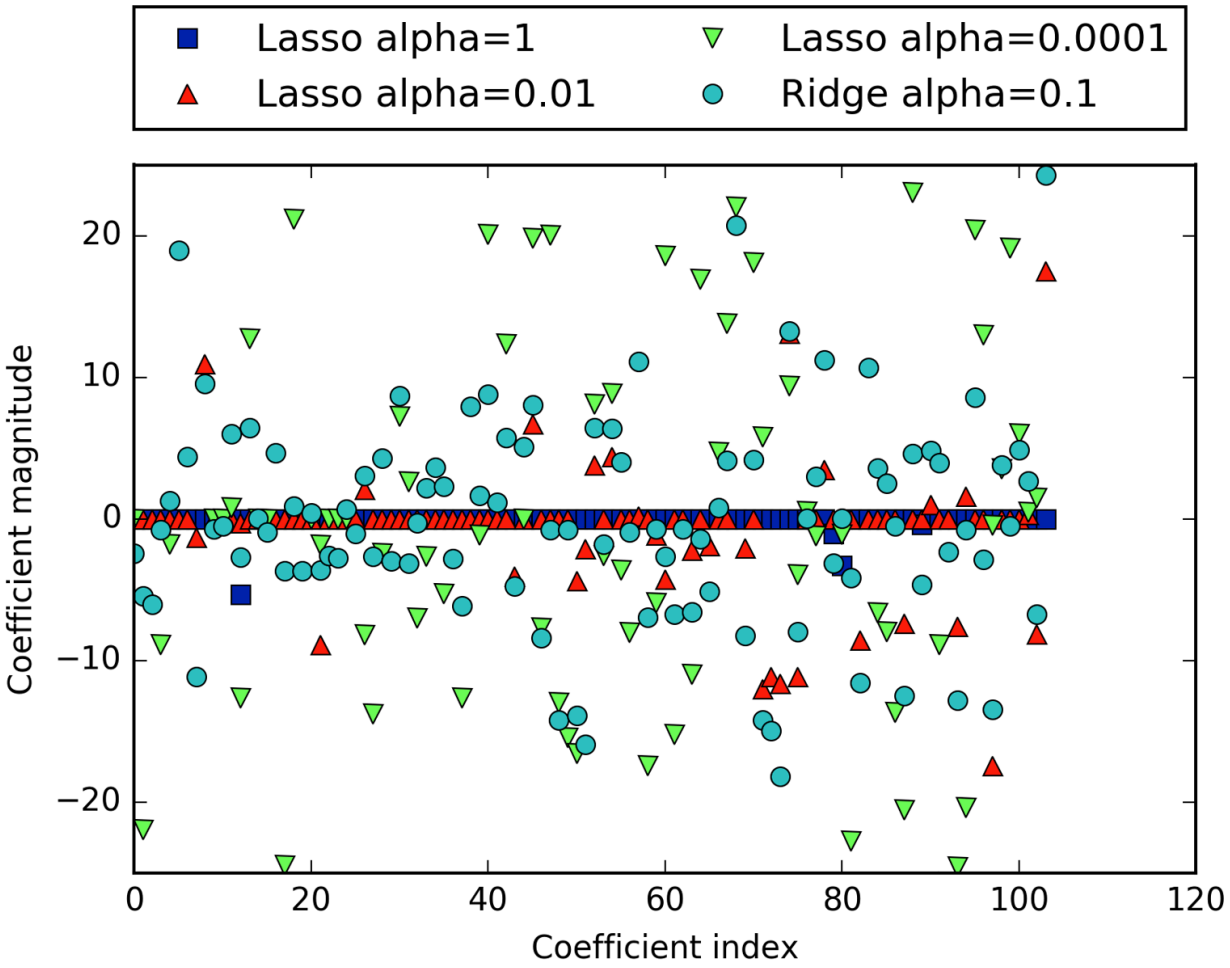


Figure 2-14. Comparing coefficient magnitudes for lasso regression with different values of alpha and ridge regression

When alpha is too high, too many features are zeroed out

When alpha is too low, the model is effectively unregularized

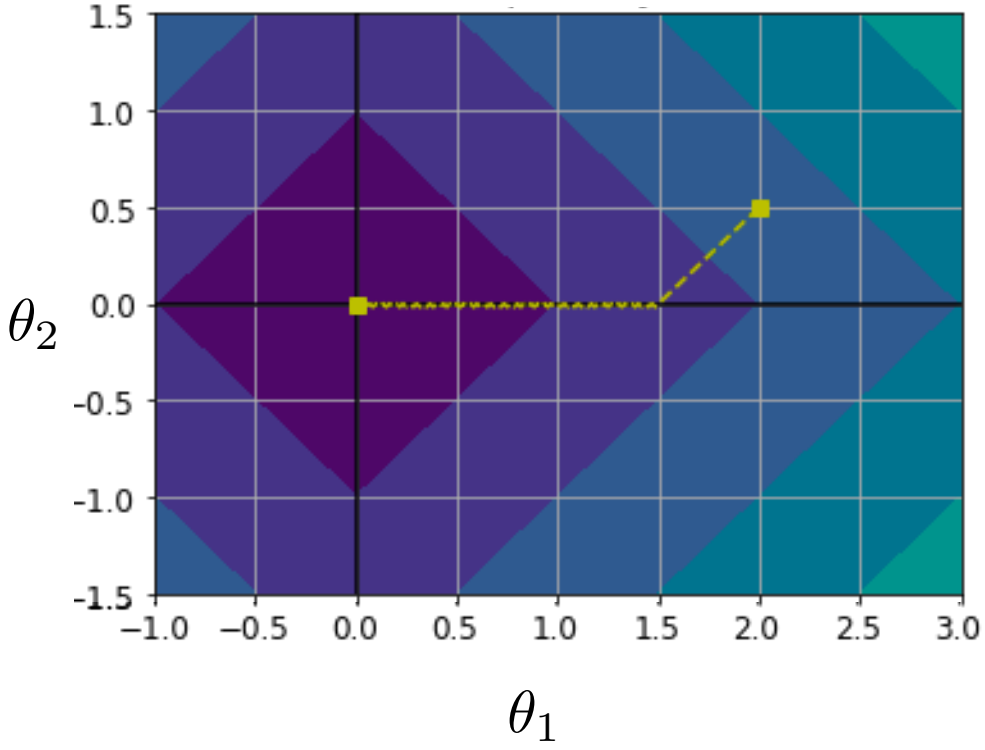
Lasso alpha=0.01 is similar to Ridge alpha=0.1 but some coefficients are zeroed out

Jupyter Notebook 02-supervised-learning .ipynb [38]

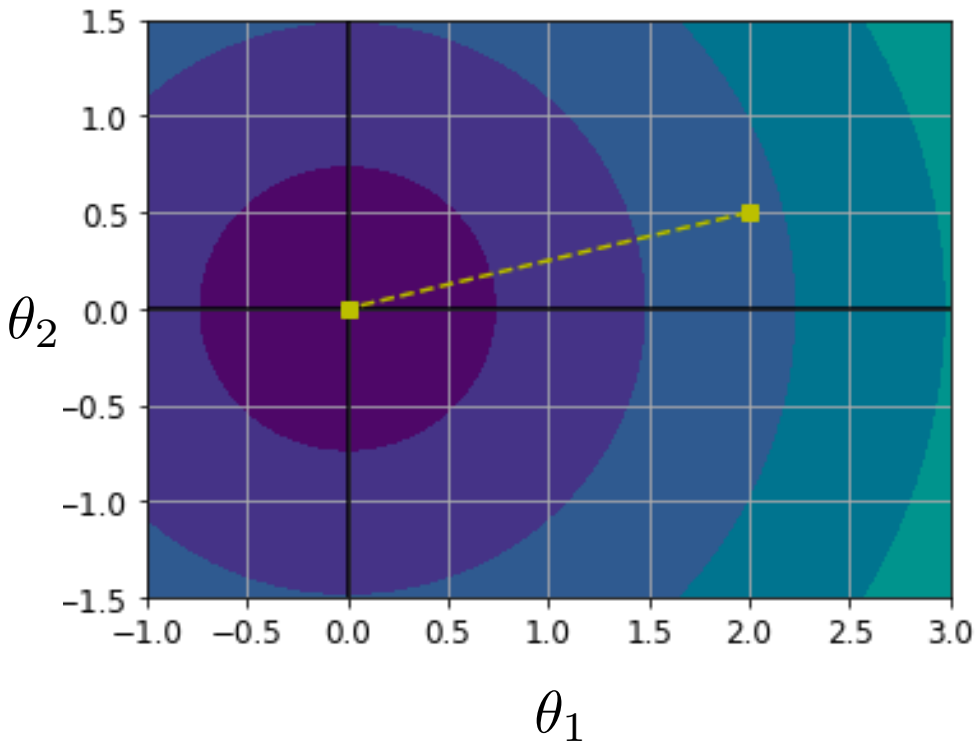
Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Lasso versus Ridge Regression

ℓ_1 penalty (Lasso)



ℓ_2 penalty (Ridge)

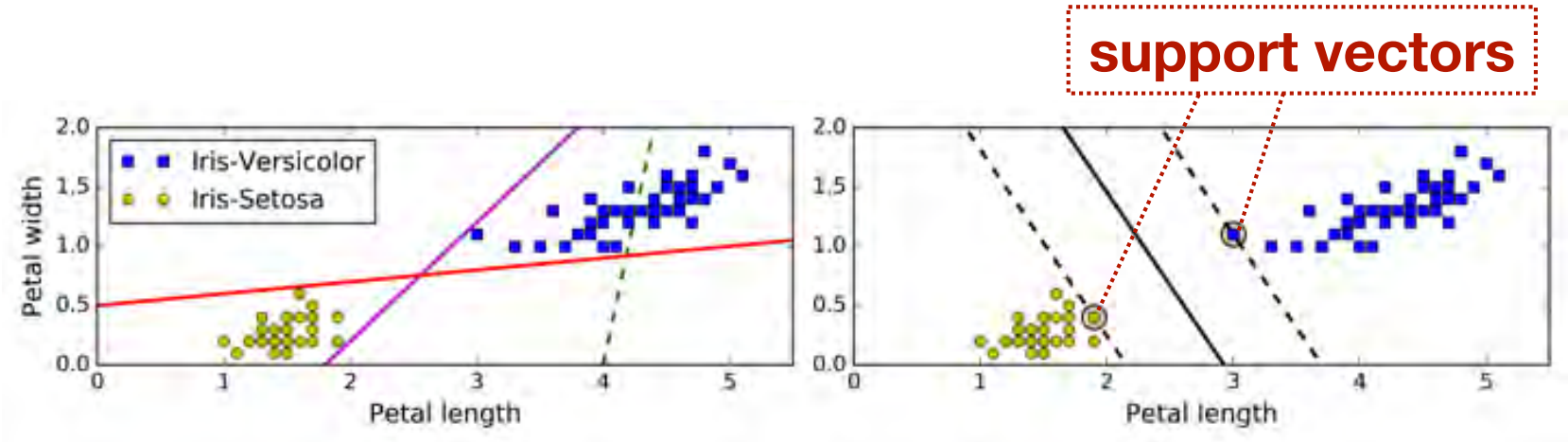


Observe that in Lasso θ_2 remains 0 until it reaches a minimum value (0.5) while in Ridge the value of θ_2 can be a value between 0 and 0.5

Géron, Aurélien. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* O'Reilly Media, Inc.

Linear Support Vector Classifier

- **Find the linear classifier with the best separation (margin) between the two classes**
  - Operationally the data points used to make this calculation are the support vectors



Three possible linear classifiers

Linear classifier with the widest margin

Géron, Aurélien. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* O'Reilly Media, Inc.

Logistic Reg versus Linear SVC

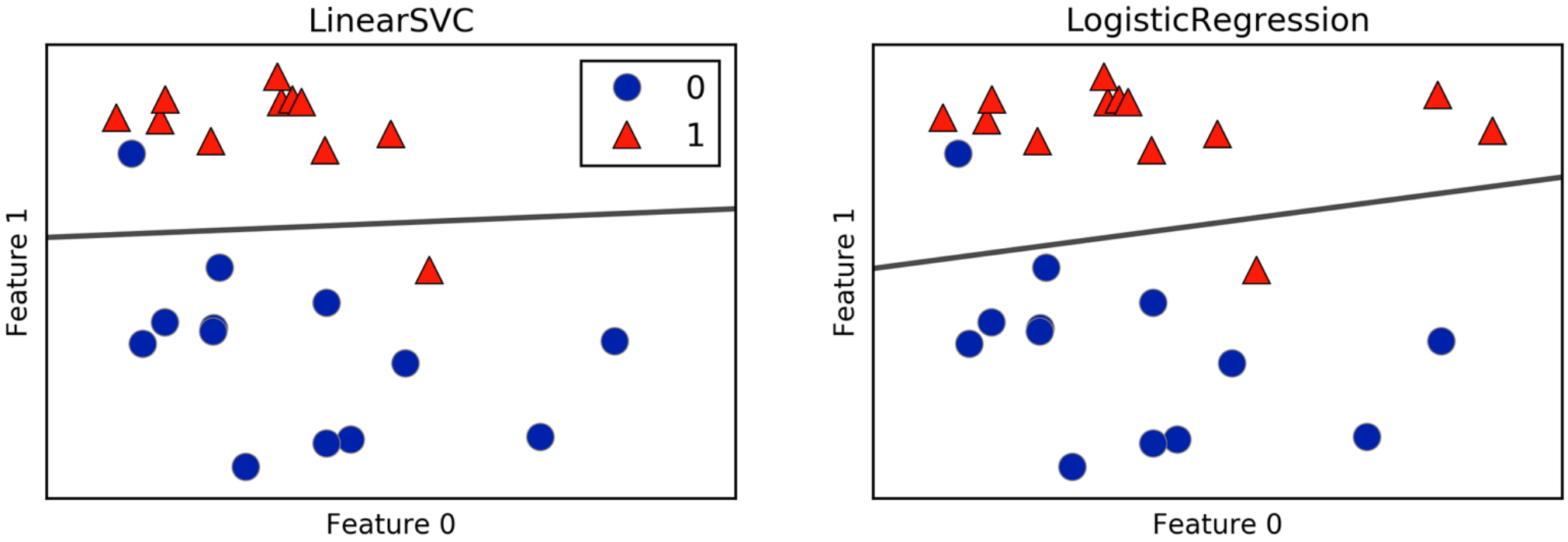


Figure 2-15. Decision boundaries of a linear SVM and logistic regression on the forge dataset with the default parameters

Jupyter Notebook 02-supervised-learning .ipynb [39]

Coefficient Magnitude

- **C is the inverse of regularization strength**

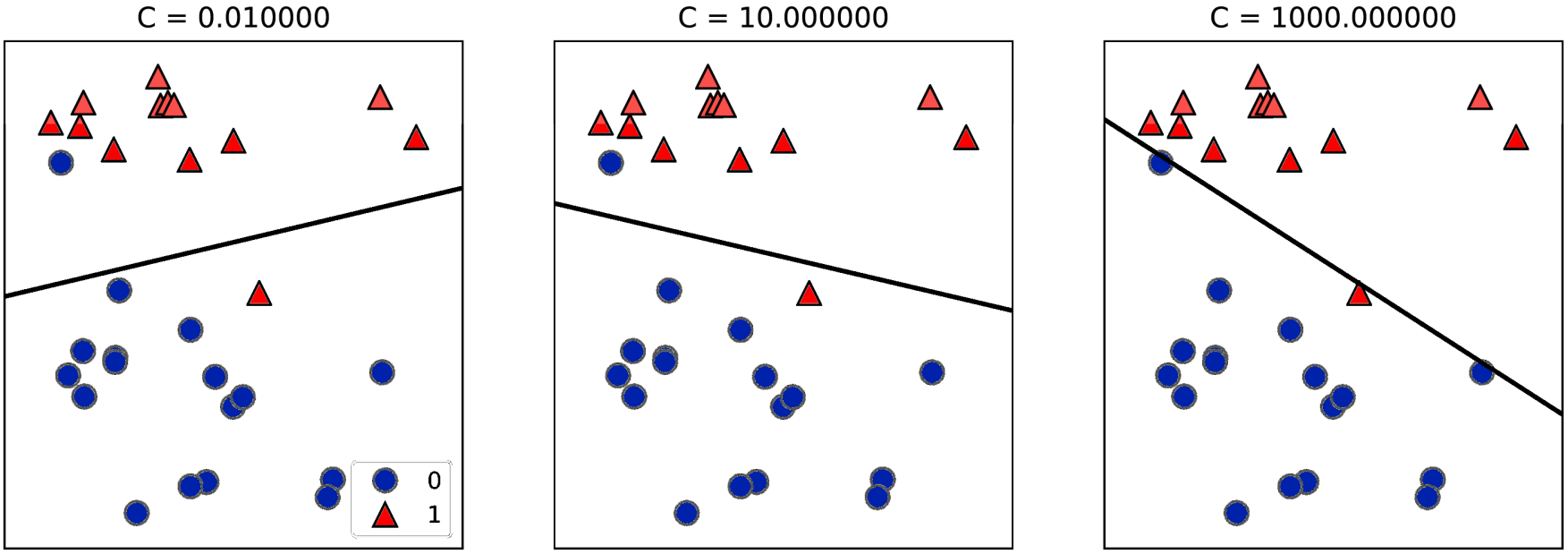


Figure 2-16. Decision boundaries of a linear SVM on the forge dataset for different values of C

Low values of C - algorithm adjusts to the majority of the data points

High values of C - algorithm attempts to correctly classify as many individual data points as possible

Jupyter Notebook 02-supervised-learning .ipynb [40]

One versus Rest Classification

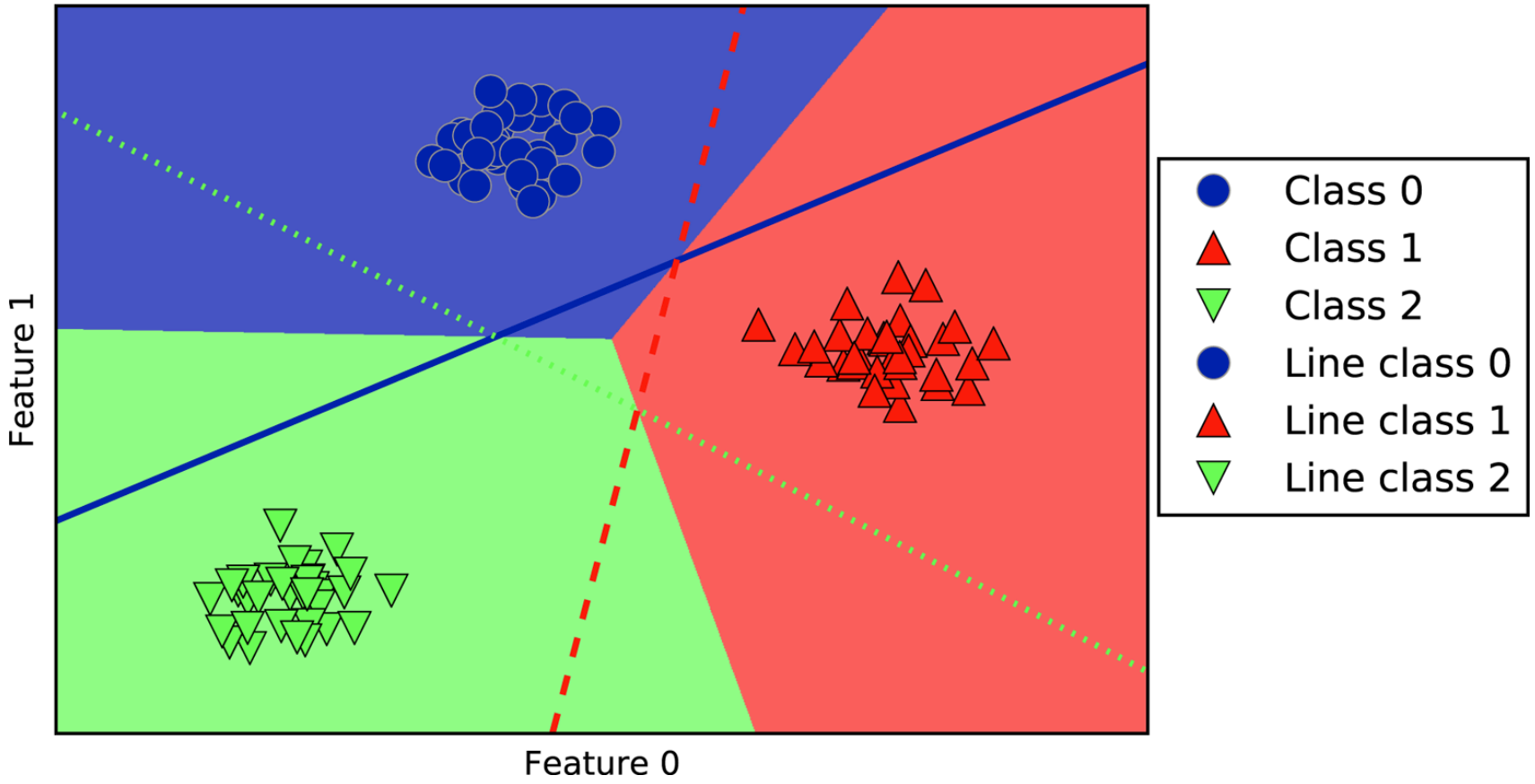


Figure 2-21. Multiclass decision boundaries derived from the three one-vs.-rest classifiers

Jupyter Notebook 02-supervised-learning .ipynb [49]

Linear Models Overview

Jupyter Notebook
data71200class5b.ipynb

- **Assumptions**
  - Linear relationship between input and output
  - No noise in input or output
  - Independence (lack of correlation in input)
  - Gaussian (normally) distributed data

- **Best Practices**
  - Scale inputs

Linear Model Activity

- **Using the wine dataset in the same notebook as the K-nearest neighbor activity and based on the code in data71200class5b.ipynb**
  - Import the logistic regression and linear SVC models
  - For each select the best parameters using grid search
  - Report evaluation metrics for the best and worst performing parameters

Linear Models Overview

Jupyter Notebook
data71200class5b.ipynb

- **Parameters**
  - Regularization parameter: alpha (regression) or C (classification)
  - Type of regularization (L1 or L2)
- **Strengths**
  - Fast to train (good for large datasets)
  - Fast at prediction
  - Easy to understand the algorithm
  - Work well even when there are a large number of features compared to the number of samples
- **Weaknesses**
  - Not so easy to interpret a model's coefficients
  - Generalization may be poor with small number of features

Upcoming Work

- **Project 1 due on June 13**
- **DataCamp for June 13**
  - *Linear Classifiers in Python*
- **Videos for June 17 (password: data71200)**
  - Naive Bayes/Decision Trees: https://vimeo.com/404803334
  - SVMs/Uncertainty Estimates: https://vimeo.com/409982391
- **Reading for June 17**
  - Ch 2: “Supervised Learning” in Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O’Reilly Media, Inc. 70–106 and 121–131