Chapter 4

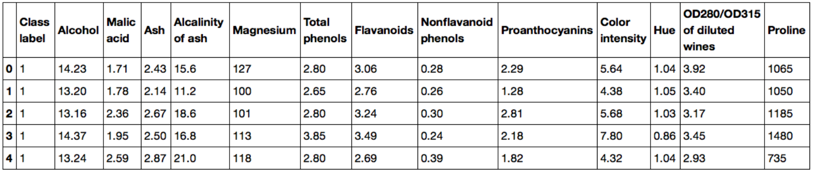
Data Preprocessing: Practical Issues

July 10, 2016

Splitting data into train and test

- Download wine dataset
    - Three classes which map to different types of grapes in Italy
- Cannot train and test on the same data
- So allocate some portion for testing and use the rest for training
    - 70-30 or 80-20 split
- Splitting three ways is a better idea to allocate some dev data
- N-fold cross-validation
- Scikit-learn helper methods (e.g. train_test_split())
- Chapter 4 iPython notebook

Wine dataset



	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

Feature scaling

- Imagine we have two features
    - $1 < x_1 < 10$
    - $1 < x_2 < 100000$
- Algorithm will likely focus on optimizing w_2
- As this will produce the largest changes in perceptron error
- KNN based on Euclidean distance will be dominated by x_2
- Two common approaches
    - Normalization
    - Standartization

Normalization

Normalization refers to the rescaling of the features to a range of [0, 1]. To normalize the data, we apply the min-max scaling to each feature column, where the new value $x_{norm}^{(i)}$ of a sample $x^{(i)}$ is calculated as follows:

$$x_{norm}^{(i)} = \frac{x^{(i)} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}$$

Here, $x^{(i)}$ is a particular sample, x_{min} is the smallest value in a feature column, and x_{max} the largest value, respectively.

Standartization

- Normalization gives us values in a bounded interval
- Standartization can be more practical:
- Many ML algorithms initialize the weights to zero
- Standartization centers the columns at $mean = 0$ and $std = 1$
- So feature columns take the form of a normal distribution
- This makes it easer to learn the weights
- Standartization encodes useful info about outliers
- Vs. normalization which scales the data to a fixed range

The procedure of standardization can be expressed by the following equation:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

Here, μ_x is the sample mean of a particular feature column and σ_x the corresponding standard deviation, respectively.

- Example of using normalization and standardization

L1 regularization

Recall L2 regularization – one approach to reduce model complexity

$$L2 : \|\mathbf{w}\|_2^2 = \sum_{j=1}^{m} w_j^2$$

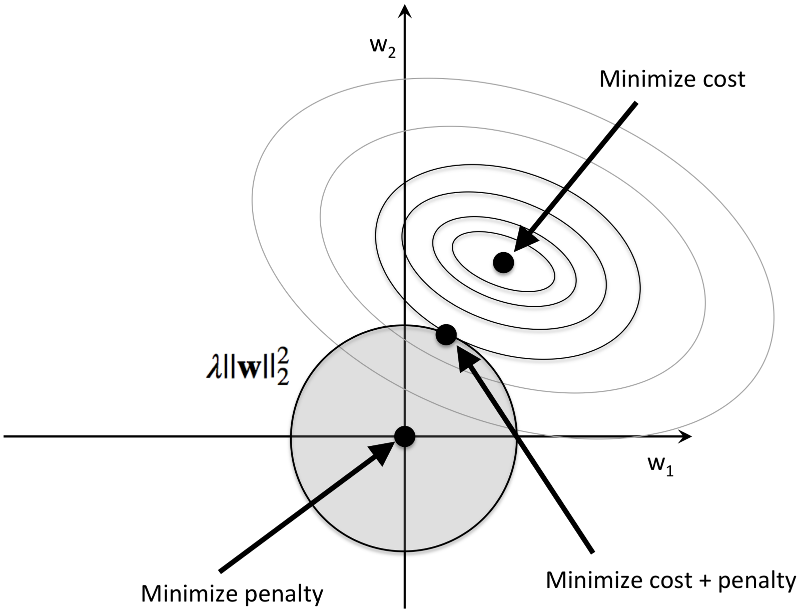
An alternative approach is *L1 regularization*:

$$L1 : \|\mathbf{w}\|_1 = \sum_{j=1}^{m} |w_j|$$

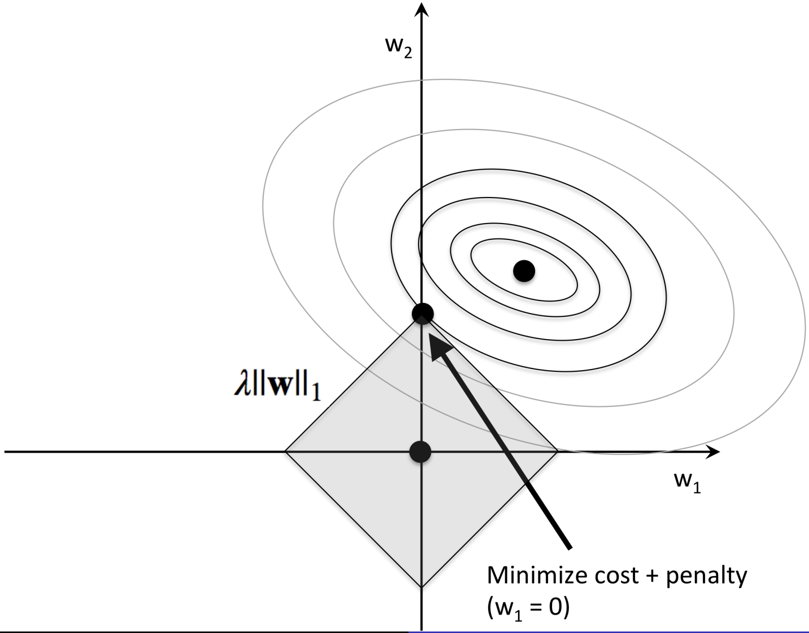
L1 regularization

- L1 yields sparse solutions
- Most feature weights will be zero
- Useful for high-dimensional datasets with irrelevant features
- It can be viewed as a technique for feature selection
- Some intuition as to why this is the case will follow

L2 regularization



L1 regularization



Sparcity

- Regularization penalty and cost pull in opposite directions
- Regularization wants the weight to be at (0, 0)
- I.e. regularization prefers a simpler model
- And decreases the dependence of the model on the training data
- L1 in scikit-learn

●