CAP5638 Project 1

Classification Using Maximum-likelihood, Parzen Window, and k-Nearest Neighbors

Suhib Sam Kiswani

October 26, 2015

The algorithms were implemented in *Python 3.4*, with a dependence on the *scipy* [1] library.

1 Maximum likelihood estimation

1.1 Parametric Forms

1.1.1 Normal Density

The discriminant function for the normal density is:

$$g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

where:

$$\mathbf{W}_i = -0.5 \boldsymbol{\Sigma}_i^{-1}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1} \mu_i$$

$$w_{i0} = -0.5 \mu_i^T \boldsymbol{\Sigma}_i^{-1} \mu_i - 0.5 \ln(\det \boldsymbol{\Sigma}_i) + \ln P(\omega_i)$$

Using the training samples, the mean and covariance can be estimated with maximum likelihood using the following definitions:

$$\hat{\mu}_i = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k \qquad \widehat{\boldsymbol{\Sigma}}_i = \frac{1}{n} \sum_{k=1}^{n} (\mathbf{x}_k - \hat{\mu}_i)(\mathbf{x}_k - \hat{\mu}_i)^T$$

Which yields:

$$p(\mathbf{x}|\omega_\mathbf{i}) = \frac{1}{\sqrt{(2\pi)^n \det \widehat{\boldsymbol{\Sigma}}_i}} \exp\left(-\frac{1}{2}(\mathbf{x} - \hat{\mu}_i)^T \widehat{\boldsymbol{\Sigma}}_i^{-1} (\mathbf{x} - \hat{\mu}_i)\right)$$

The classification of instance $\mathbf{x}$ is $\omega_\mathbf{i} = \arg\max_{\omega_\mathbf{i}} |p(\mathbf{x}|\omega_\mathbf{i}) P(\omega_\mathbf{i})|$

1.1.2 Uniform

TODO

write out eqns

1.2 Experimental Results

1.2.1 Iris Data Set

1. **Normal Density** The estimated parameters of $\hat{\theta}$ for each of the classes from the training samples were:

$$\hat{\mu}_1 = (4.98181818, 3.39090909, 1.45151515, 0.25151515)$$

$$\widehat{\mathbf{\Sigma}}_1 = \begin{pmatrix} 0.10876033 & 0.08619835 & 0.02033058 & 0.01093664 \\ 0.08619835 & 0.13597796 & 0.01410468 & 0.00865014 \\ 0.02033058 & 0.01410468 & 0.03401286 & 0.00825528 \\ 0.01093664 & 0.00865014 & 0.00825528 & 0.0134068 \end{pmatrix}$$

$$\hat{\mu}_2 = (5.89090909, 2.78787879, 4.26363636, 1.31515152)$$

$$\widehat{\mathbf{\Sigma}}_2 = \begin{pmatrix} 0.2353719 & 0.06867769 & 0.165427 & 0.05256198 \\ 0.06867769 & 0.08530762 & 0.07743802 & 0.04442608 \\ 0.165427 & 0.07743802 & 0.20110193 & 0.06933884 \\ 0.05256198 & 0.04442608 & 0.06933884 & 0.0394674 \end{pmatrix}$$

$$\hat{\mu}_3 = (6.66060606, 2.94848485, 5.58484848, 1.99393939)$$

$$\widehat{\mathbf{\Sigma}}_3 = \begin{pmatrix} 0.36359963 & 0.11887971 & 0.2851607 & 0.03976125 \\ 0.11887971 & 0.10613407 & 0.08861341 & 0.03817264 \\ 0.2851607 & 0.08861341 & 0.29219467 & 0.03202938 \\ 0.03976125 & 0.03817264 & 0.03202938 & 0.06784206 \end{pmatrix}$$

This method correctly classified 48 of the 51 testing samples (94.1% accuracy).

TODO

add decision boundary graphs

---

2. **Uniform**

TODO

gotta do it

1.2.2 UCI Wine Data Set

2 Parzen window estimation

TODO

You need to choose proper window functions, which you need to specify in the report along with the resulting discriminant functions for classification for each dataset. Here we choose the parameter values using the leave-one-out performance on the training set: For a set of candidate values, we compute the leave-one-out performance on the training set for each candidate and the optimal one is the one that gives the best leave-one-out performance (in case there are ties, specify how the ties will be broken).

Using hypercube, the window function is:

$$\phi(\mu) = \begin{cases} 1 & |\mu_j| \leq 1/2; \quad j \in [1, ..., d] \\ 0 & otherwise \end{cases}$$

2.1 Experimental Results

2.1.1 Iris Data Set

2.1.2 UCI Wine Data Set

2.1.3 Handwritten Digits Data Set

3 k-nearest neighbors

The k-nearest neighbors classifier was implemented using a kd-tree, subdividing along the median of the training data. This distance metric d used for this classifier was Euclidean distance ($||\mathbf{x} - \mathbf{y}||$).

TODO

Speed up using k-d tree *With k from 1 to 10 with an increment of 1, first build a k-d tree from the training set and then classify the test samples using the k-nearest neighbor classifier by finding the nearest neighbors using the k-d tree. Compare the classification accuracy and the number of distance calculations with the basic k nearest neighbor implementation on the three datasets. Summarize your observations and justify your results.*

3.1 Experimental Results

3.1.1 Iris Data Set

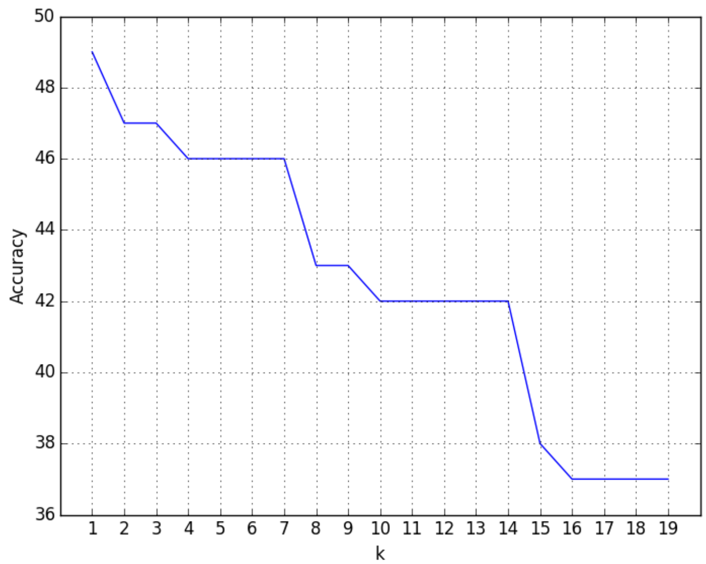

Figure 3.1: Accuracy of the k-nearest neighbors classifier using a kd-tree for $1 \leq k \leq 19$ on the Iris data set.

The k-nearest neighbors classifier achieved a maximum classification rate of 96.08% accuracy for $k = 1$.

3.1.2 UCI Wine Data Set

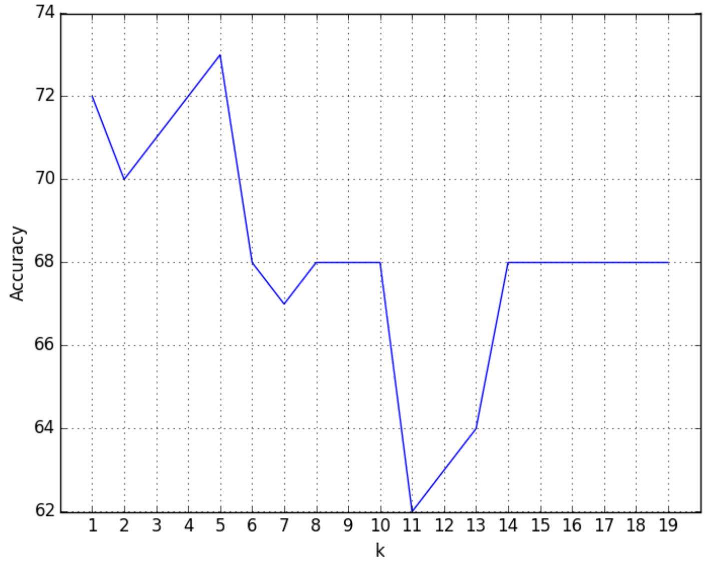

Figure 3.2: The accuracy of the k-nearest neighbors classifier using a kd-tree for $1 \le k \le 19$ on the UCI wine data set.

Here, the k-nearest neighbors classifier achieved a maximum classification rate of 82.02% accuracy for $k = 5$.

3.1.3 Handwritten Digits Data Set

4 Analysis

TODO

You need to compare different methods in terms of classification performance and required time for classification, and give justifications for your observed empirical results.

---

5 Extra Credit

TODO

Please state clearly in your report if you have implemented any of the following extra credit options.

5.1 Recognition of my handwritten digits

TODO

Apply the best classifier you have for hand written digit recognition on a test set consisting of your own written digits (you need to create the dataset). Document the classification performance, what you have done to improve the performance, and any additional issues you have handled.

References

[1] Jones E, Oliphant E, Peterson P, *et al.* **SciPy: Open Source Scientific Tools for Python**, 2001-, `http://www.scipy.org/` [Online; accessed 2015-10-24].