CAP5638 Project 1

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

Suhib Sam Kiswani

October 28, 2015

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

The **Speed up using k-d tree** extra credit option was implemented.

1 Maximum likelihood estimation

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:

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

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

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

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

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

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 Experimental Results

1.1.1 Iris Data Set

This method correctly classified 48 of the 51 testing samples (94.1% accuracy). The estimated parameters of $\hat{\theta}$ for each of the classes from the training samples were:

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

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

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

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

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

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

1.1.2 UCI Wine Data Set

This classifier correctly classified 85 out of 89 testing samples (95.51% accuracy). The estimated parameters of $\hat{\mu}_i$ for each of the classes from the training samples were (*note:* the values of sigma are too large to include here):

$$\hat{\mu}_1 = (13.796, 1.865, 2.446, 16.763, 106.2, 2.885, 3.072, 0.289, 2.008, 5.504, 1.07, 3.17, 1108.4)$$

$$\hat{\mu}_2 = (12.281, 1.813, 2.237, 19.803, 93.686, 2.236, 2.027, 0.357, 1.585, 3.128, 1.022, 2.745, 532.057)$$

$$\hat{\mu}_3 = (13.172, 2.985, 2.445, 21.375, 99.875, 1.652, 0.788, 0.438, 1.184, 7.273, 0.683, 1.658, 616.458)$$

1.1.3 Handwriting Data set

This classifier correctly classified 545 out of 2007 testing samples (27.15% accuracy). The corresponding values for this data set are too large to include here.

2 Parzen window estimation

The window function is:

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

Where the window width is given by h_n and the volume is given by:

$$V_n = h_n^d$$

1. **Hypercube Kernel**

$$p(\mathbf{x}|\omega_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \phi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right)$$

2. **Gaussian Kernel**

$$p(\mathbf{x}|\omega_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \phi\left(\frac{1}{\sqrt{2\pi}^d V_n} \exp\left[-\frac{1}{2}\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right)^2\right]\right)$$

All data processed by this classifier was transformed using the *z-score*:

$$z_{\omega_i}(\mathbf{x}) = \frac{\mathbf{x} - \mu_{\omega_i}}{\sigma_{\omega_\mathbf{i}}}$$

A given point $\mathbf{x}$ is classified as class ω_i where:

$$\omega_\mathbf{i} = \arg\max_{\omega_\mathbf{i}} |p(z_{\omega_i}(\mathbf{x})|\omega_\mathbf{i}) P(\omega_\mathbf{i})|$$

2.1 Experimental Results

For all given data sets, the *Hypercube kernel* was the most accurate method. Overall, this classifier was the slowest (e.g. required the most processing time).

2.1.1 Iris Data Set

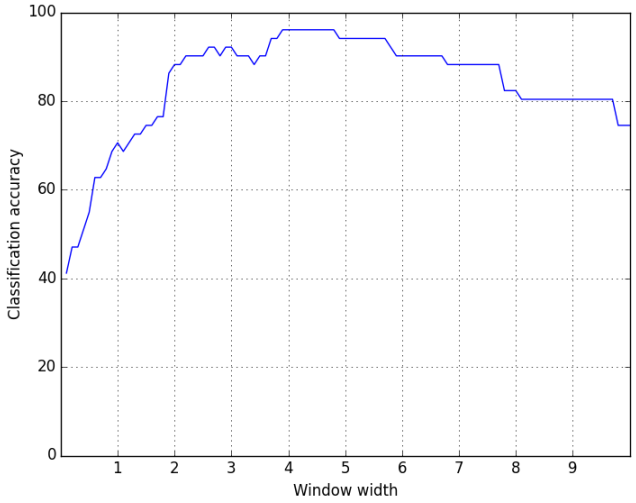

Figure 2.1: Accuracy of the Parzen window classifier using the *Hypercube kernel* across different window widths for the iris data set.

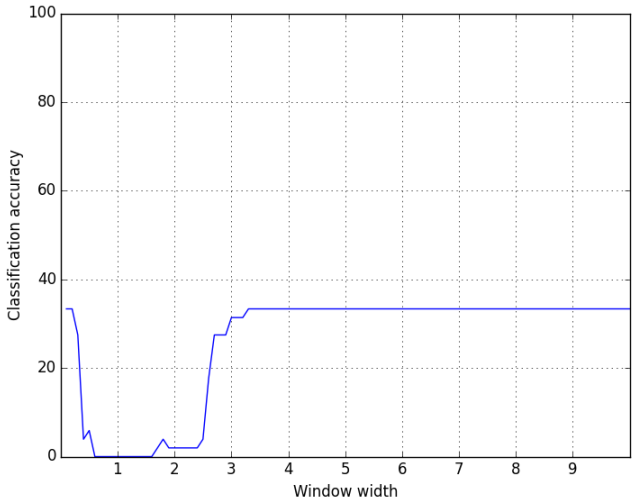

Figure 2.2: Accuracy of the Parzen window classifier using the *Gaussian kernel* across different window widths for the iris data set.

2.1.2 UCI Wine Data Set

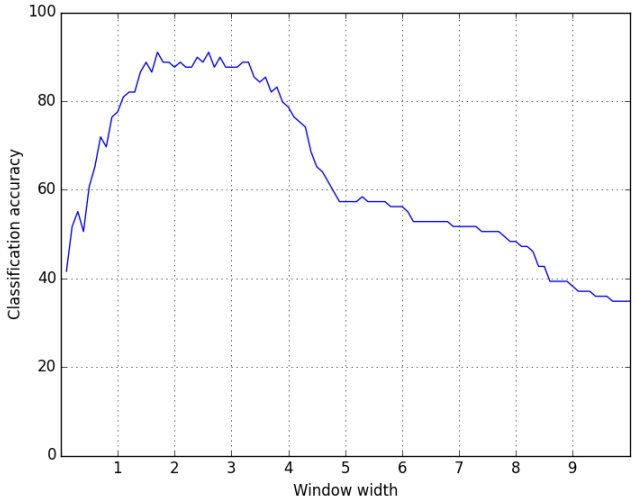

Figure 2.3: Accuracy of the Parzen window classifier using the hypercube window function across different window widths for the wine data set.

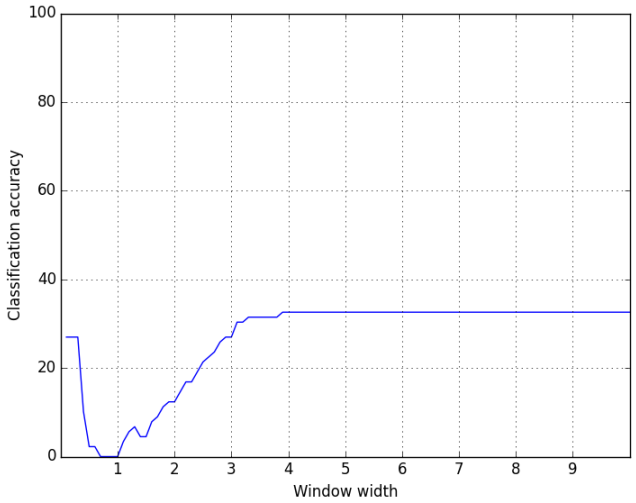

Figure 2.4: Accuracy of the Parzen window classifier using the *Gaussian kernel* across different window widths for the wine data set.

2.1.3 Handwritten Digits Data Set

The Parzen window classifier failed to perform very well for this data set, in addition to its extremely slow run-time (which is expected, given the high dimensionality of this data set).

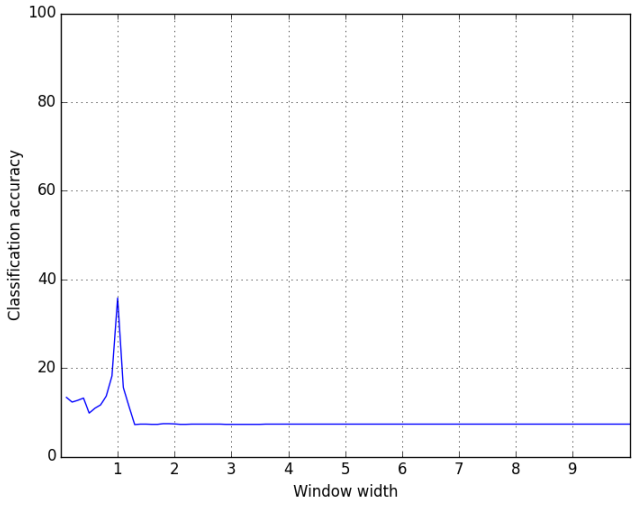

Figure 2.5: Accuracy of the Parzen window classifier using the *Hypercube kernel* across different window widths for the handwriting data set.

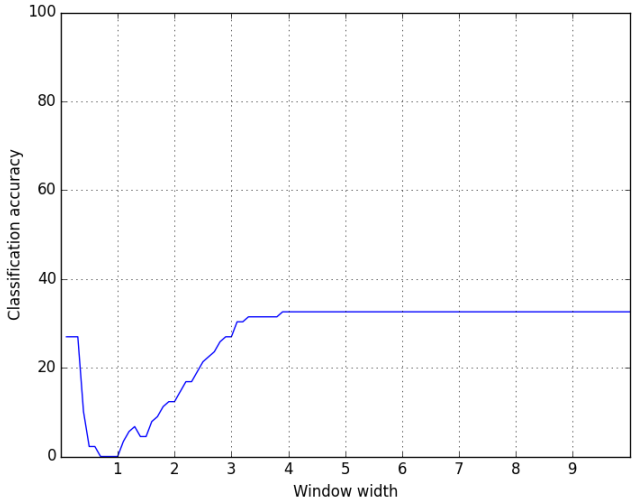

Figure 2.6: Accuracy of the Parzen window classifier using the *Gaussian kernel* across different window widths for the handwriting data set.

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 ($\|\mathbf{x} - \mathbf{y}\|$).

3.1 Experimental Results

3.1.1 Iris Data Set

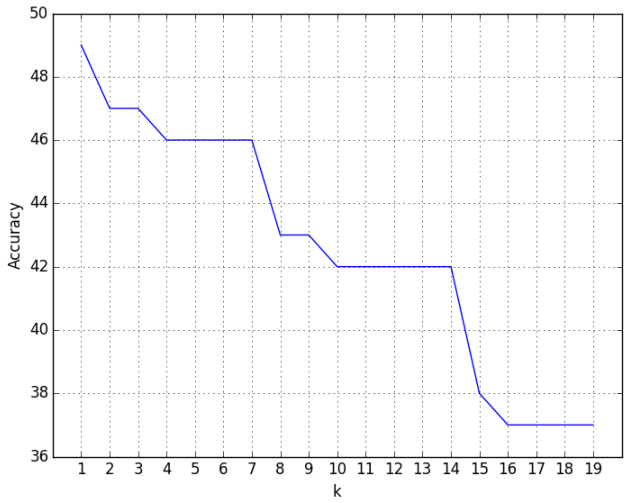

Figure 3.1: The number of correct classifications by$k$-nearest neighbors classifier using a kd-tree for $1 \leq k \leq 19$ on the iris data set.

The k-nearest neighbors classifier achieved a maximum classification rate of 96.08% accuracy for $k = 1$. Performance of this classifier degraded rapidly, which is most likely a side of effect of the fact that *sepal width* had poor class correlation, yet was given equal weight as other features. This can be improved by using applying a normalizing transform to the data set, or choosing a better suited distance metric for this data set.

3.1.2 UCI Wine Data Set

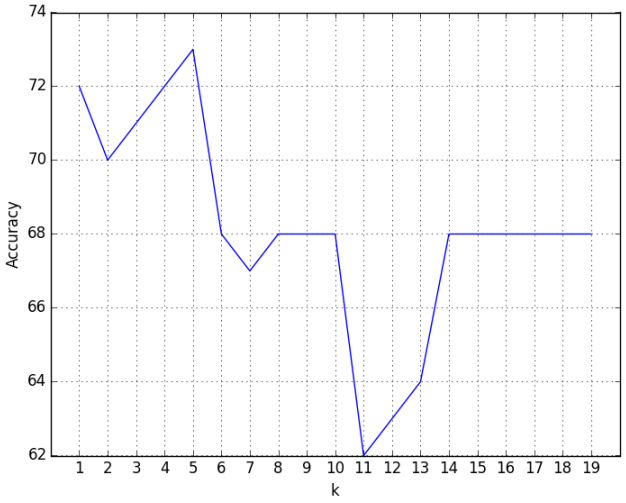

Figure 3.2: The number of correct classifications by$k$-nearest neighbors classifier using a kd-tree for $1 \leq k \leq 19$ on the wine data set.

Here, the k-nearest neighbors classifier achieved a maximum classification rate of 82.02% accuracy for $k = 5$. Notably, in this instance, the classifier could not exceed the threshold of 90% accuracy. Most likely this is due to the fact that Euclidean distance may not be a suitable distance metric for this data set. In the *README* included with this data set, a 1-NN classifier achieved 96.1% accuracy on z-transformed data. As such, that may result in a significant improvement (as was the case in the iris data set).

3.1.3 Handwritten Digits Data Set

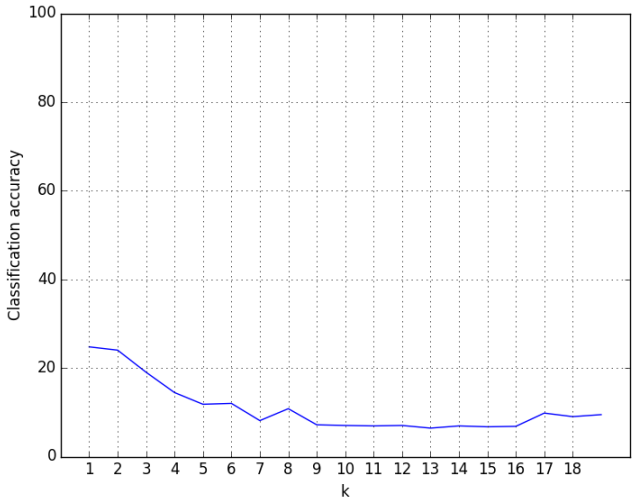

Figure 3.3: The accuracy of the k-nearest neighbors classifier using a kd-tree for $1 \leq k \leq 19$ on the handwriting data set.

Again, the optimal k for the classifier is 1, with 497 correct classifications out of 2007 (24.76% accuracy). No transformations were applied to this data set, as reflected in the poor accuracy of this classifier.

4 Analysis

1. **Iris data set.** All methods performed very well on the iris data set. This is most likely due to the fact that there is a high class correlation for two of the features.

2. **Wine data set.** This data set proved more difficult for my implementation of the classifiers.

3. **Handwriting data set.** Every classifier seemed to have difficulties properly classifying samples from this data set. Most likely, this is because the domain of each feature was $[-1, -1]$ and no normalizing transformations were used in any of the classifiers. This is in addition to the difficulties in classifying high-dimensional data.

Also, there is most likely an error in the implementation of the Gaussian kernel for the Parzen Windows classifier, since it performed atrociously on every data set.

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].