Week 2

RA : Aysun Koçak
Instructor : Dr. Aykut Erdem

Department of Computer Engineering
Hacettepe University
Outline

- k Nearest Neighbor (k-NN)
- Linear Regression
- Note for Assignment
k-Nearest Neighbor (k-NN)

- Nonparametric algorithm
- To classify a new point, find its K nearest neighbors from the training data.

If \( K = 5 \), then in this case query instance \( x_q \) will be classified as negative since three of its nearest neighbors are classified as negative.

*credit - Siddharth Deokar slides*
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets

n_neighbors = 10

# import some data to play with
iris = datasets.load_iris()
x = iris.data[:, 2]  # we only take the first two features. We could 
                    # avoid this ugly slicing by using a two-dim dataset
y = iris.target

h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF3300', '#33FF00', '#00FF00'])

for weights in ['uniform', 'distance']:
    # we create an instance of Neighbors Classifier and fit the data.
    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(x, y)

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, m_max]x[y_min, y_max].
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("K-Class Classification (k = %i, weights = '%s')" % (n_neighbors, weights))

plt.show()

k-NN Results

3-Class classification (k = 3, weights = 'distance')

3-Class classification (k = 3, weights = 'uniform')

3-Class classification (k = 10, weights = 'distance')

3-Class classification (k = 10, weights = 'uniform')
Drawbacks

- Keep all data to classify a new input
- All neighbors have same importance
- Determining hyper-parameters
  - What is the best distance to use?
  - What is the best value of k to use?
- Distance usually relates to all the attributes and assumes all of them have the same effects on distance (curse of dimensionality)
Determining Parameters

- k Nearest Neighbor (k-NN)
- Linear Regression

Credit - Fei-Fei Li & Andrej Karpathy slides
Determining Parameters

Accuracy = (\# of correctly classified examples / \# of examples) \times 100

Credit - Fei-Fei Li & Andrej Karpathy slides
Linear Regression

Housing Prices (Portland, OR)

Price (in 1000s of dollars)

Size (feet²)

Credit - Andrew Ng slides
Linear Regression

Training Set

<table>
<thead>
<tr>
<th>Size in feet² (x)</th>
<th>Price ($) in 1000's (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>460</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>1534</td>
<td>315</td>
</tr>
<tr>
<td>852</td>
<td>178</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Hypothesis: \( h_\theta(x) = \theta_0 + \theta_1 x \)

\( \theta_i \)'s: Parameters

How to choose \( \theta_i \)'s?

Credit - Andrew Ng slides
Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal: minimize $$J(\theta_0, \theta_1)$$
Linear Regression

(for fixed $\theta_1$, this is a function of $x$)

$\theta_1 = 0.5$

Credit - Andrew Ng slides
Linear Regression

Credit - Andrew Ng slides