MATH 829: Introduction to Data Mining and Analysis
Categorical data

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

March 2, 2016

Predicting categorical variables

- So far, we developed methods for modelling *quantitative* or *continuous* outputs.

Predicting categorical variables

- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.

Predicting categorical variables

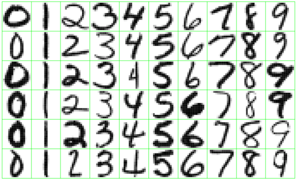
- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).

Predicting categorical variables

- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).
  2. Web browsing analysis: link clicked or not clicked?

Predicting categorical variables

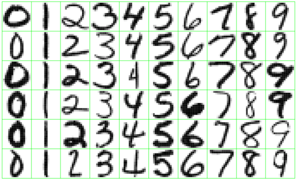
- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).
  2. Web browsing analysis: link clicked or not clicked?
  3. Handwritten digits recognition ($Y \in \{0, \ldots, 9\}$).



ESL, Figure 1.2.

Predicting categorical variables

- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).
  2. Web browsing analysis: link clicked or not clicked?
  3. Handwritten digits recognition ($Y \in \{0, \dots, 9\}$).



ESL, Figure 1.2.

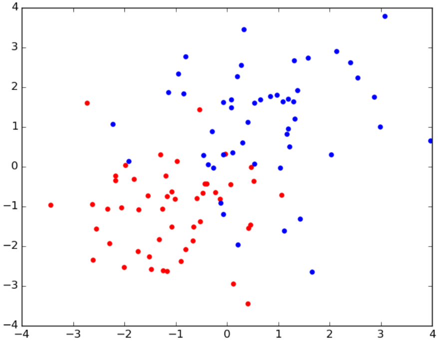
- We begin with two very simple approaches: linear regression and nearest neighbors.

Linear regression

- We are given $X \in \mathbb{R}^{n \times 2}$ and $Y \in \{0, 1\}^n$.
- Think of y_i as x_i's label (red/blue say).

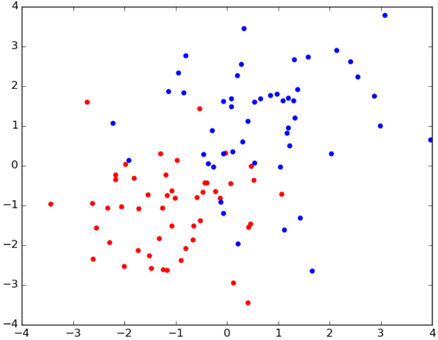
- We are given $X \in \mathbb{R}^{n\times 2}$ and $Y \in \{0,1\}^n$.
- Think of y_i as x_i's label (red/blue say).

x	y
(0.1,0.4)	1
(0.5,0.8)	0
(0.6, 0.2)	1
$\vdots$	$\vdots$



- We are given $X \in \mathbb{R}^{n\times 2}$ and $Y \in \{0,1\}^n$.
- Think of y_i as x_i's label (red/blue say).

x	y
(0.1,0.4)	1
(0.5,0.8)	0
(0.6, 0.2)	1
$\vdots$	$\vdots$



We want to predict new points' category.

Linear regression (cont.)

First approach: use linear regression as if the output was continuous.

Linear regression (cont.)

First approach: use linear regression as if the output was continuous.

- Fit $Y = X\beta + \epsilon$ (linear decision boundary).

Linear regression (cont.)

First approach: use linear regression as if the output was continuous.

- Fit $Y = X\beta + \epsilon$ (linear decision boundary).
- Given $x = (x_1, x_2)^T$, use $x^T\beta$ to predict the label.

Linear regression (cont.)

First approach: use linear regression as if the output was continuous.

- Fit $Y = X\beta + \epsilon$ (linear decision boundary).
- Given $x = (x_1, x_2)^T$, use $x^T\beta$ to predict the label.
- Output is in $\{0, 1\}$, but $x^T\beta \in \mathbb{R}$.

First approach: use linear regression as if the output was continuous.

- Fit $Y = X\beta + \epsilon$ (linear decision boundary).
- Given $x = (x_1, x_2)^T$, use $x^T\beta$ to predict the label.
- Output is in $\{0, 1\}$, but $x^T\beta \in \mathbb{R}$.
- Use

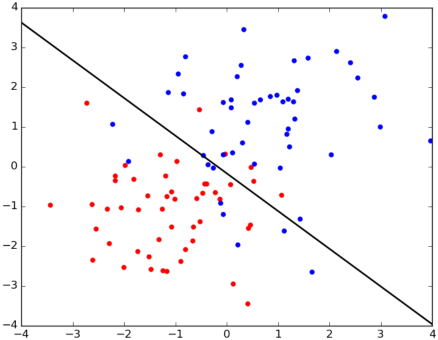
$$\hat{y} = \begin{cases} 0 & \text{if } x^T\beta < 0.5 \\ 1 & \text{if } x^T\beta \geq 0.5 \end{cases}.$$

Linear regression (cont.)

```
# X = 2*n by 2, Y = 2*n by 1 {0,1} labels
# Include an intercept
Xp = np.ones((2*n,3))
Xp[:,1:3] = X

# Use regression
beta = np.linalg.lstsq(Xp,Y)[0]

# Our decision boundary is
# beta[0] + beta[1] *x + beta[2]*y = 0.5,
# or y = (0.5-beta[0]-beta[1]*x)/beta[2]
```

Linear regression (cont.)

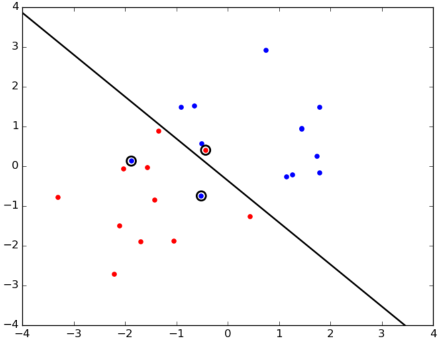
```
# X = 2*n by 2, Y = 2*n by 1 {0,1} labels
# Include an intercept
Xp = np.ones((2*n,3))
Xp[:,1:3] = X

# Use regression
beta = np.linalg.lstsq(Xp,Y)[0]

# Our decision boundary is
# beta[0] + beta[1] *x + beta[2]*y = 0.5,
# or y = (0.5-beta[0]-beta[1]*x)/beta[2]
```



Test error

- As usual, we split our data into train and test sets.
- Train model on training set.
- Compute classification error on test set.

```
Xp_train, Xp_test, y_train, y_test =
  train_test_split(Xp, Y, test_size=0.25)
beta = np.linalg.lstsq(Xp_train,y_train)[0]
Y_hat = Xp_test.dot(beta)
```



Test set predicted labels

In general, when working with k categories, can use a loss-function

$$(L(i,j))_{i,j=1}^{k},$$

where $L(i,j)$ = cost for classifying i as j.

Nearest neighbors

Nearest neighbors: use closest observations in the training set to make predictions.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i.$$

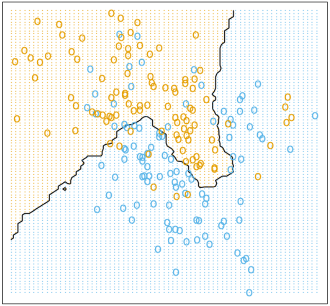
Here $N_k(x)$ denotes the k-nearest neighbors of x (w.r.t. some metric, e.g. Euclidean distance).

Nearest neighbors: use closest observations in the training set to make predictions.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i.$$

Here $N_k(x)$ denotes the k-nearest neighbors of x (w.r.t. some metric, e.g. Euclidean distance). Use a "majority vote" to determine final labels

$$\hat{G}(x) = \begin{cases} 0 & \text{if } \hat{Y}(x) < 0.5 \\ 1 & \text{if } \hat{Y}(x) \geq 0.5 \end{cases}.$$

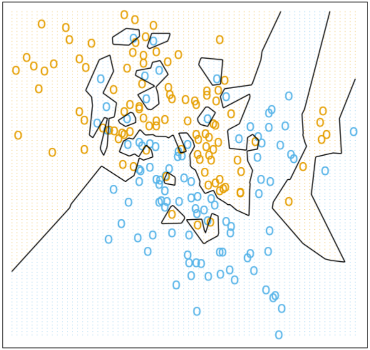


ESL, Fig. 2.2: 15 Nearest Neighbor classifier

Nearest neighbors

Reducing the number of neighbors leads to:

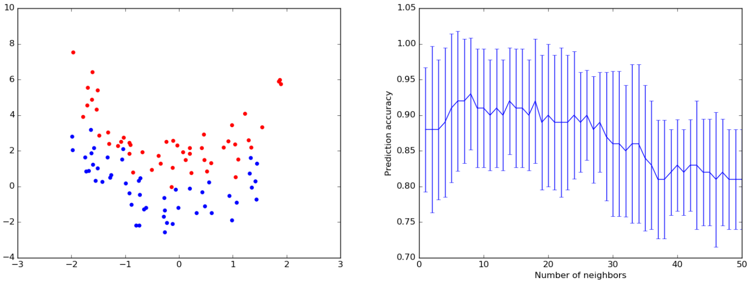
- A smaller training error (training error is 0 when using $k = 1$ neighbor).
- Can use train/test sets to choose k.
- Although a small k leads to a small training error, the model may not generalize well (large test error).



ESL, Fig. 2.3, 1 Nearest classifier

Example

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=i)
```




```
for i in range(1,N_max+1):
    neigh = KNeighborsClassifier(n_neighbors=i)
    for j, (train, test) in enumerate(kf):
        X_train, X_test, y_train, y_test =  X[train], X[test],
                                            Y[train], Y[test]
        neigh.fit(X_train,y_train)
        scores[i-1,j] = neigh.score(X_test, y_test)
```

Linear regression vs Nearest neighbors

A bias-variance tradeoff:

Linear regression:

- Relies on the assumption that the decision boundary is linear.
- Decision boundary is smooth.
- High bias, low variance.

Linear regression vs Nearest neighbors

A bias-variance tradeoff:

Linear regression:

- Relies on the assumption that the decision boundary is linear.
- Decision boundary is smooth.
- High bias, low variance.

Nearest neighbors:

- Adaptive, less assumptions on the data.
- A particular decision may depend only on a handful of points. Less robust.
- More wiggly and unstable.
- Low bias, high variance.

Linear regression vs Nearest neighbors

A bias-variance tradeoff:

Linear regression:

- Relies on the assumption that the decision boundary is linear.
- Decision boundary is smooth.
- High bias, low variance.

Nearest neighbors:

- Adaptive, less assumptions on the data.
- A particular decision may depend only on a handful of points. Less robust.
- More wiggly and unstable.
- Low bias, high variance.

Each method has its own situations for which it works best

Linear regression vs Nearest neighbors

A bias-variance tradeoff:

Linear regression:

- Relies on the assumption that the decision boundary is linear.
- Decision boundary is smooth.
- High bias, low variance.

Nearest neighbors:

- Adaptive, less assumptions on the data.
- A particular decision may depend only on a handful of points. Less robust.
- More wiggly and unstable.
- Low bias, high variance.

Each method has its own situations for which it works best
Both methods can lead to very good predictions.

Linear regression vs Nearest neighbors

A bias-variance tradeoff:

Linear regression:

- Relies on the assumption that the decision boundary is linear.
- Decision boundary is smooth.
- High bias, low variance.

Nearest neighbors:

- Adaptive, less assumptions on the data.
- A particular decision may depend only on a handful of points. Less robust.
- More wiggly and unstable.
- Low bias, high variance.

Each method has its own situations for which it works best

Both methods can lead to very good predictions.

Many strategies exist to improve these methods (as we will see later).