MATH 567: Mathematical Techniques in Data Science
Categorical data

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

February 27, 2017

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

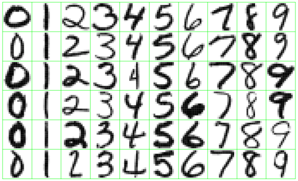
- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).

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

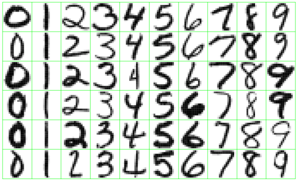
- So far, we developed methods for modelling *quantitative* or *continuous* outputs.
- We will now discuss techniques to model *categorical* or *discrete* outputs.
- Examples of problems:
  1. You receive an email. Is it spam or not? (binary response).
  2. Web browsing analysis: link clicked or not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



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

Example:

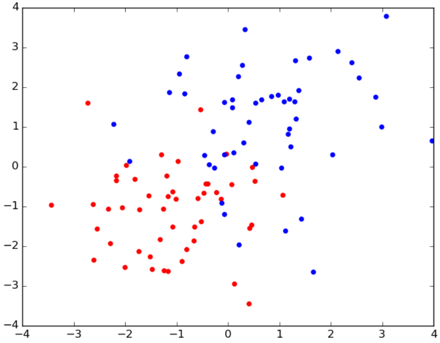
- We are given $X \in \mathbb{R}^{n \times 2}$ and $Y \in \{0, 1\}^n$.
- Think of y_i as x_i's label (red/blue say).

Linear regression

Example:

- 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
⋮	⋮

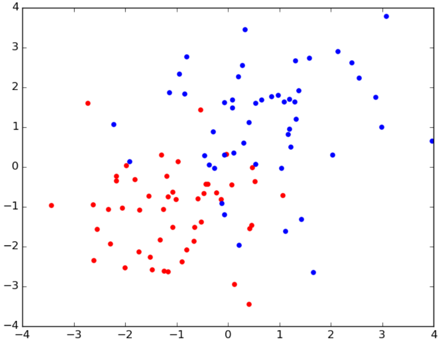


Linear regression

Example:

- 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 the category of new points.

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 $x^T\beta \in \mathbb{R}$.

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 $x^T\beta \in \mathbb{R}$.
- 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.)

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

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

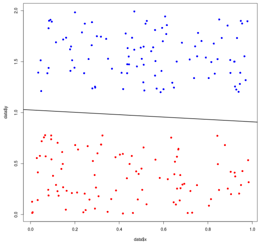
Remarks:

1. Linear regression is not always appropriate for categorical data.
2. For example, coding (e.g. $1, 2, 3, \ldots$) often implies an ordering.

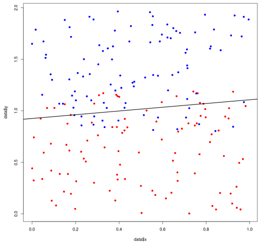
Linear regression (cont.)

Example with simulated data (exercise):

```
eps = 0.2
ydata1 = runif(100, 0,1 + eps)
ind1 = matrix(0, nrow=100,ncol=1)
ydata2 = runif(100,1-eps, 2)
ind2 = matrix(1, nrow=100,ncol=1)

ydata = c(ydata1,ydata2)
xdata = runif(200,0,1)
ind = c(ind1, ind2)

data = data.frame(x = xdata, y = ydata, cat = ind)

plot(data$x, data$y, col=c("red", "blue")[data$cat+1])
model = lm(cat ~ x + y, data=data)
coef = model$coefficients
abline((0.5-coef[1])/coef[3], -1*coef[2]/coef[3], lwd=2)
```



(a) eps = -0.2

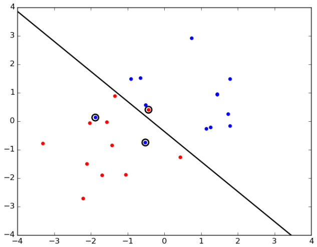


(b) eps = 0.2

Test error

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

```
library(caTools)

sample = sample.split(data$x, SplitRatio = .75)
train = subset(data, sample == TRUE)
test = subset(data, sample == FALSE)

model = lm(cat ~ x + y, data=train)
yhat = as.numeric(predict(model, test) > 0.5)
error = test$cat != yhat
error_rate = sum(error)/length(error)*100
```



Exercise: Compute the test error as a function of eps in the previous example, for multiple train/test sets.

Note: we can also use a more general loss function $(L(i,j))_{i,j=1}^{k}$.

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

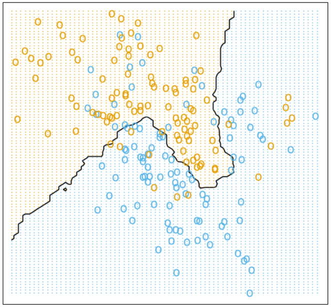
Nearest neighbors: use closest observations in the training set to make predictions.

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

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

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

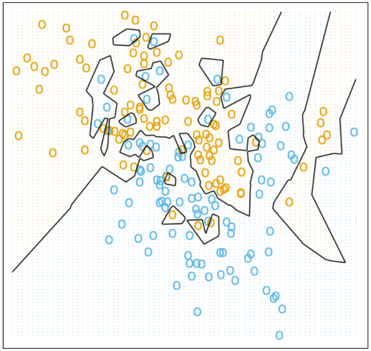


ESL, Fig. 2.2: 15 Nearest Neighbor classifier

Nearest neighbors

Reducing the number of neighbors leads to:

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

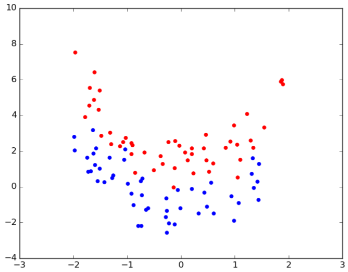


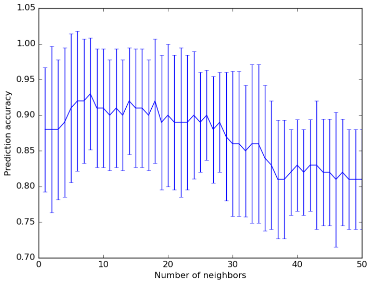
ESL, Fig. 2.3, 1 Nearest classifier

Example

Note: Variables should usually be scaled before using k-NN.

```
train.X = scale(train[,c("x","y")])
train.Y = train$cat
test.X = scale(test[,c("x","y")])

knn_pred = knn(train.X, test.X, train.Y, k=1)
error = test$cat != knn_pred
error_rate_knn = sum(error)/length(error)*100
```





```
error_knn = rep(0,10)
for(i in 1:10){
  knn_pred = knn(train.X, test.X, train.Y, k=i)
  error = test$cat != knn_pred
  error_knn[i] = sum(error)/length(error)*100
}
```

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).