Review

• Metrics for evaluating classification models
  – Accuracy, precision, recall
• Cross-validation should be used to avoid over-fitting
  – K-fold or LOOCV
• Logistic regression
  – Estimates $\Pr[Y = 1|X = x]$ using sigmoid
  – Maximum Likelihood Estimation (MLE) for objective
  – Can use gradient descent for training
  – Very interpretable
Outline

• Logistic regression
  – Gradient descent for logistic regression
• Linear Discriminant Analysis (LDA)
• Lab (logistic regression, LDA, kNN)
• Feature selection
  – Wrapper
  – Filter
  – Embedded methods
Classification

Suppose we are given a training set of N observations \( \{x^{(1)}, \ldots, x^{(n)}\} \) and \( \{y^{(1)}, \ldots, y^{(n)}\} \), \( x^{(i)} \in \mathbb{R}^d \), \( y^{(i)} \in \{-1, 1\} \)

Classification problem is to estimate \( f(x) \) from this data such that \[ f(x^{(i)}) = y^{(i)} \]
Logistic Regression

\[ h_\theta(x) = g(\theta^T x) \]
\[ g(z) = \frac{1}{1 + e^{-z}} \]

- \( \theta^T x \) should be large negative values for negative instances
- \( \theta^T x \) should be large positive values for positive instances

Probabilistic model \( h_\theta(x) = P[y = 1|x; \theta] \)

- Predict \( y = 1 \) if \( h_\theta(x) \geq 0.5 \)
- Predict \( y = 0 \) if \( h_\theta(x) < 0.5 \)

Logistic Regression is a linear classifier!
Maximum Likelihood Estimation (MLE)

Given training data $X = \{x^{(1)}, \ldots, x^{(n)}\}$ with labels $Y = \{y^{(1)}, \ldots, y^{(n)}\}$

What is the likelihood of training data for parameter $\theta$?

Define likelihood function

$$\text{Max}_\theta L(\theta) = P[Y|X; \theta]$$

Assumption: training points are independent

$$L(\theta) = \prod_{i=1}^{n} P[y^{(i)}|x^{(i)}; \theta]$$
MLE for Logistic Regression

\[ p(y|x, \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y} \]

\[
\theta_{\text{MLE}} = \arg \max_{\theta} \sum_{i=1}^{n} \log p(y^{(i)} | x^{(i)}; \theta) \\
= \arg \max_{\theta} \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x))
\]

- Substitute in model, and take negative to yield

**Logistic regression objective:**

\[
\min_{\theta} J(\theta) \\
J(\theta) = - \sum_{i=1}^{n} \left[ y^{(i)} \log h_\theta(x^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_\theta(x^{(i)})\right) \right]
\]
Gradient Descent for Logistic Regression

\[ J(\theta) = -\sum_{i=1}^{n} \left[ y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right] \]

\[ J(\theta) = -\sum_{i=1}^{n} C_i \]

Want \( \min_{\theta} J(\theta) \)

- Initialize \( \theta \)
- Repeat until convergence

\[ \theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \]

Simultaneous update for \( j = 0 \ldots d \)
Computing Gradients

- **Derivative of sigmoid**
  \[ g(z) = \frac{1}{1 + e^{-z}}; \quad g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2} = g(z)(1 - g(z)) \]

- **Derivative of hypothesis**
  \[ h_\theta(x) = g(\theta^T x) = g(\theta_j x_j + \sum_{k \neq j} \theta_k x_k) \]
  \[ \frac{\partial h_\theta(x)}{\partial \theta_j} = \frac{\partial g(\theta^T x)}{\partial \theta_j} x_j = g(\theta^T x)(1 - g(\theta^T x)) x_j \]

- **Derivation of \( C_i \)**
  \[ \frac{\partial C_i}{\partial \theta_j} = y^{(i)} \frac{1}{h_\theta(x^i)} g(\theta^T x^{(i)}) \left( 1 - g(\theta^T x^{(i)}) \right) x_j^{(i)} - \]
  \[ (1 - y^{(i)}) \frac{1}{1 - h_\theta(x^i)} g(\theta^T x^{(i)}) \left( 1 - g(\theta^T x^{(i)}) \right) x_j^{(i)} \]
  \[ = \left( y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \]
Gradient Descent for Logistic Regression

\[
J(\theta) = - \sum_{i=1}^{n} \left[ y^{(i)} \log h_{\theta}(x^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\theta}(x^{(i)})\right) \right].
\]

Want \ \min_{\theta} J(\theta)

- Initialize \ \theta
- Repeat until convergence

\[
\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left( h_{\theta} \left( x^{(i)} \right) - y^{(i)} \right)
\]

\[
\theta_j \leftarrow \theta_j - \alpha \left[ \sum_{i=1}^{n} \left( h_{\theta} \left( x^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \right]
\]

(simultaneous update for \ j = 0 \ldots d)
Gradient Descent for Logistic Regression

Want \( \min_{\theta} J(\theta) \)

- Initialize \( \theta \)
- Repeat until convergence (simultaneous update for \( j = 0 \ldots d \))

\[
\begin{align*}
\theta_0 &\leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \\
\theta_j &\leftarrow \theta_j - \alpha \left[ \sum_{i=1}^{n} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \right]
\end{align*}
\]

This looks IDENTICAL to Linear Regression!

- However, the form of the model is very different:

\[
h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}
\]
LDA

• Classify to one of $k$ classes

• Logistic regression computes directly
  – $P[Y = 1 | X = x]$
  – Assume sigmoid function

• LDA uses Bayes Theorem to estimate it

\[
P[Y = k | X = x] = \frac{P[X = x | Y = k]P[Y = k]}{P[X = x]} \]

– Let $\pi_k = P[Y = k]$ be the prior probability of class $k$ and $f_k(x) = P[X = x | Y = k]$
LDA

Assume $f_k(x)$ is Gaussian!

Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp \left( -\frac{1}{2\sigma^2_k} (x - \mu_k)^2 \right)$$

$$p_k(x) = \frac{\pi_k}{\sum_{l=1}^K \pi_l} \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{1}{2\sigma^2} (x - \mu_k)^2 \right)$$

Assumption: $\sigma_1 = \ldots \sigma_k = \sigma$
LDA decision boundary

Pick class $k$ to maximize

$$
\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)
$$

Example: $k = 2, \pi_1 = \pi_2$
Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2}$

Bayes decision boundary  Estimated decision boundary
LDA in practice

Given training data \((x^{(i)}, y^{(i)}), i = 1, \ldots, n, y^{(i)} \in \{1, \ldots, K\}\)

1. Estimate mean and variance

\[
\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i = k} x^{(i)} \\
\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i = k} (x^{(i)} - \hat{\mu}_k)^2
\]

2. Estimate prior

\[
\hat{\pi}_k = \frac{n_k}{n}.
\]

Given testing point \(x\), predict \(k\) that maximizes:

\[
\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)
\]
LDA vs Logistic Regression

• Logistic regression computes directly $\Pr[Y = 1|X = x]$ by assuming sigmoid function
  – Uses Maximum Likelihood Estimation

• LDA uses Bayes Theorem to estimate it
  – Estimates mean, co-variance, and prior from training data
  – Assumes Gaussian distribution for $f_k(x) = \Pr[X = x|Y = k]$

• Which one is better?
  – LDA can be sensitive to outliers
  – LDA works well for Gaussian distribution
  – Logistic regression is more complex to solve, but more expressive
```r
> library(ISLR)
> fix(Smarket)
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Lag1</th>
<th>Lag2</th>
<th>Lag3</th>
<th>Lag4</th>
<th>Lag5</th>
<th>Volume</th>
<th>Today</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>-1.055</td>
<td>5.01</td>
<td>1.1913</td>
<td>0.959</td>
<td>Up</td>
</tr>
<tr>
<td>2</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>-1.055</td>
<td>1.2965</td>
<td>1.032</td>
<td>Up</td>
</tr>
<tr>
<td>3</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>1.4112</td>
<td>-0.623</td>
<td>Down</td>
</tr>
<tr>
<td>4</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>1.276</td>
<td>0.614</td>
<td>Up</td>
</tr>
<tr>
<td>5</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>1.2057</td>
<td>0.213</td>
<td>Up</td>
</tr>
<tr>
<td>6</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>1.3491</td>
<td>1.392</td>
<td>Up</td>
</tr>
<tr>
<td>7</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>1.445</td>
<td>-0.403</td>
<td>Down</td>
</tr>
<tr>
<td>8</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.4078</td>
<td>0.027</td>
<td>Up</td>
</tr>
<tr>
<td>9</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>1.164</td>
<td>1.303</td>
<td>Up</td>
</tr>
<tr>
<td>10</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>1.2326</td>
<td>0.287</td>
<td>Up</td>
</tr>
<tr>
<td>11</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>1.309</td>
<td>-0.498</td>
<td>Down</td>
</tr>
<tr>
<td>12</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.258</td>
<td>-0.189</td>
<td>Down</td>
</tr>
<tr>
<td>13</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>1.098</td>
<td>0.68</td>
<td>Up</td>
</tr>
<tr>
<td>14</td>
<td>0.68</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>1.0531</td>
<td>0.701</td>
<td>Up</td>
</tr>
<tr>
<td>15</td>
<td>0.701</td>
<td>0.68</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.1498</td>
<td>-0.562</td>
<td>Down</td>
</tr>
</tbody>
</table>
Lab Logistic Regression

Train on data before 2005

```r
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252 9
> Direction.2005=Direction[!train]
> glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket,family=binomial,subset=train)
> summary(glm.fits)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial, data = Smarket, subset = train)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.302   -1.190   1.079    1.160    1.350

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.191213   0.333690   0.573   0.567
Lag1       -0.054178   0.051785  -1.046   0.295
Lag2       -0.045805   0.051797  -0.884   0.377
Lag3        0.007200   0.051644   0.139   0.889
Lag4        0.006441   0.051706   0.125   0.901
Lag5       -0.004223   0.051138  -0.083   0.934
Volume     -0.116257   0.239618  -0.485   0.628
```
Lab Logistic Regression

Test on data in 2005

```r
> glm.probs = predict(glm.fits, Smarket.2005, type = "response")
> glm.pred = rep("Down", nrow(Smarket.2005))
> glm.pred[glm.probs > .5] = "Up"
> head(Smarket.2005)

     Year Lag1  Lag2  Lag3  Lag4  Lag5  Volume  Today  Direction
   999 2005 -0.134  0.008 -0.007  0.715  0.431  0.7869   Down
 1000 2005 -0.812 -0.134  0.008 -0.007  0.715  1.5108  -1.167   Down
 1001 2005 -1.167 -0.812 -0.134  0.008 -0.007  1.7210  -0.363   Down
 1002 2005 -0.363 -1.167 -0.812 -0.134  0.008  1.7389   0.351    Up
 1003 2005  0.351 -0.363 -1.167 -0.812 -0.134  1.5691  -0.143   Down
 1004 2005 -0.143  0.351 -0.363 -1.167 -0.812  1.4779   0.342    Up
> head(glm.probs)

     999 1000 1001 1002 1003 1004
0.5282195 0.5156688 0.5226521 0.5138543 0.4983345 0.5010912
> head(glm.pred)
[1] "Up" "Up" "Up" "Up" "Down" "Up"
> table(glm.pred, Direction.2005)

 glm.pred Direction.2005
     Down      Up
   Down   77   97
    Up    34   44
> mean(glm.pred == Direction.2005)
[1] 0.4801587
```
```
library(MASS)
> lda.fit=lda(Direction~Lag1+Lag2,data=Smarket,subset=train)
> lda.fit
Call:
  lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
  Down  Up
0.491984 0.508016

Group means:
  Lag1   Lag2
Down 0.04279022 0.03389409
Up -0.03954635 -0.03132544

Coefficients of linear discriminants:
  LD1
Lag1 -0.6420190
Lag2 -0.5135293

> lda.pred=predict(lda.fit, Smarket.2005)
> lda.class=lda.pred$class
> table(lda.class,Direction.2005)
       Direction.2005
  Down  Up
Down  35  35
 Up  76 106

> mean(lda.class==Direction.2005)
[1] 0.5595238
```
Lab kNN

```R
> library(class)
> train.X=cbind(Lag1,Lag2)[train,]
> test.X=cbind(Lag1,Lag2)[!train,]
> train.Direction=Direction[train]
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Direction,k=1)
> table(knn.pred,Direction.2005)
   Direction.2005
knn.pred Down Up
   Down  43  58
   Up    68  83
> mean(knn.pred==Direction.2005)
[1] 0.5
> knn.pred=knn(train.X,test.X,train.Direction,k=3)
> table(knn.pred,Direction.2005)
   Direction.2005
knn.pred Down Up
   Down  48  54
   Up    63  87
> mean(knn.pred==Direction.2005)
[1] 0.5357143
> knn.pred=knn(train.X,test.X,train.Direction,k=7)
> table(knn.pred,Direction.2005)
   Direction.2005
knn.pred Down Up
   Down  41  65
   Up    70  76
> mean(knn.pred==Direction.2005)
[1] 0.4642857
```
Linear models

- Perceptron

\[ h(x) = \text{sign}(\theta^T x) \]

- Logistic regression

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

- LDA

\[ \max_k \delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \]
Outline

• Logistic regression
  – Gradient descent for logistic regression
• Linear Discriminant Analysis (LDA)
• Lab (logistic regression, LDA, kNN)
• Feature selection
  – Wrapper
  – Filter
  – Embedded methods
Supervised Learning

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled → Normalization → Feature Selection → Classification Regression

Testing

New data → Learning model → Predictions

Unlabeled → Healthy Sick Classification → Price Risk score Regression
Example: Email Classification

- **Input**: a email message
- **Output**: is the email...
  - spam,
  - work-related,
  - personal, ...

**PERSONAL**
Bag-of-Words

- **Input:** $x$ (email-valued)
- **Feature vector:**
  $$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}, \text{ e.g. } f_1(x) = \begin{cases} 1 & \text{if the email contains} \\ 0 & \text{otherwise} \end{cases}$$

- **Learn one weight vector for each class:**
  $$w_y \in \mathbb{R}^n, \quad y \in \{\text{SPAM, WORK, PERS}\}$$

Could also use frequency
- $f_i(x)$ is the number of times word $i$ appears in $x$
• Large number of words in dictionary (>50,000)
• Very sparse representation (many features set at 0)
Feature selection

• Feature Selection
  • Process for choosing an optimal subset of features according to a certain criteria

• Why we need Feature Selection:
  1. To improve performance (in terms of speed, predictive power, simplicity of the model).
  2. To visualize the data for model selection.
  3. To reduce dimensionality and remove noise.
Feature Search Space

Search Space:

Complete Set of Features

Empty Set of Features

Exponentially large!
Methods for Feature Selection

• **Wrappers**
  – Select subset of features that gives best prediction accuracy (using cross-validation)
  – Model-specific

• **Filters**
  – Compute some statistical metrics (correlation coefficient, mutual information)
  – Select features with statistics higher than threshold

• **Embedded methods**
  – Feature selection done as part of training
  – Example: Regularization (Lasso, L1 regularization)
Feature Engineering

• Feature engineering is crucial to getting good results

• Strategy: overshoot and regularize
  – Define as many features as you can
  – Use regularization for models that support it
  – Use other feature selection methods (e.g., filters) otherwise

• Do cross-validation to evaluate selected features on multiple runs

• When feature selection is frozen, evaluate on test set
Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag

- Thanks!