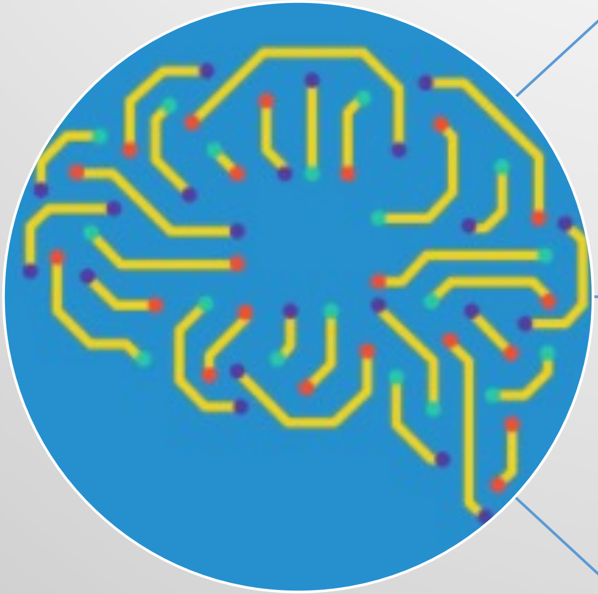
Natural Language Processing

Dr. Karen Mazidi

Part Five:
Machine Learning



Topics

- Logistic regression

Quizzes

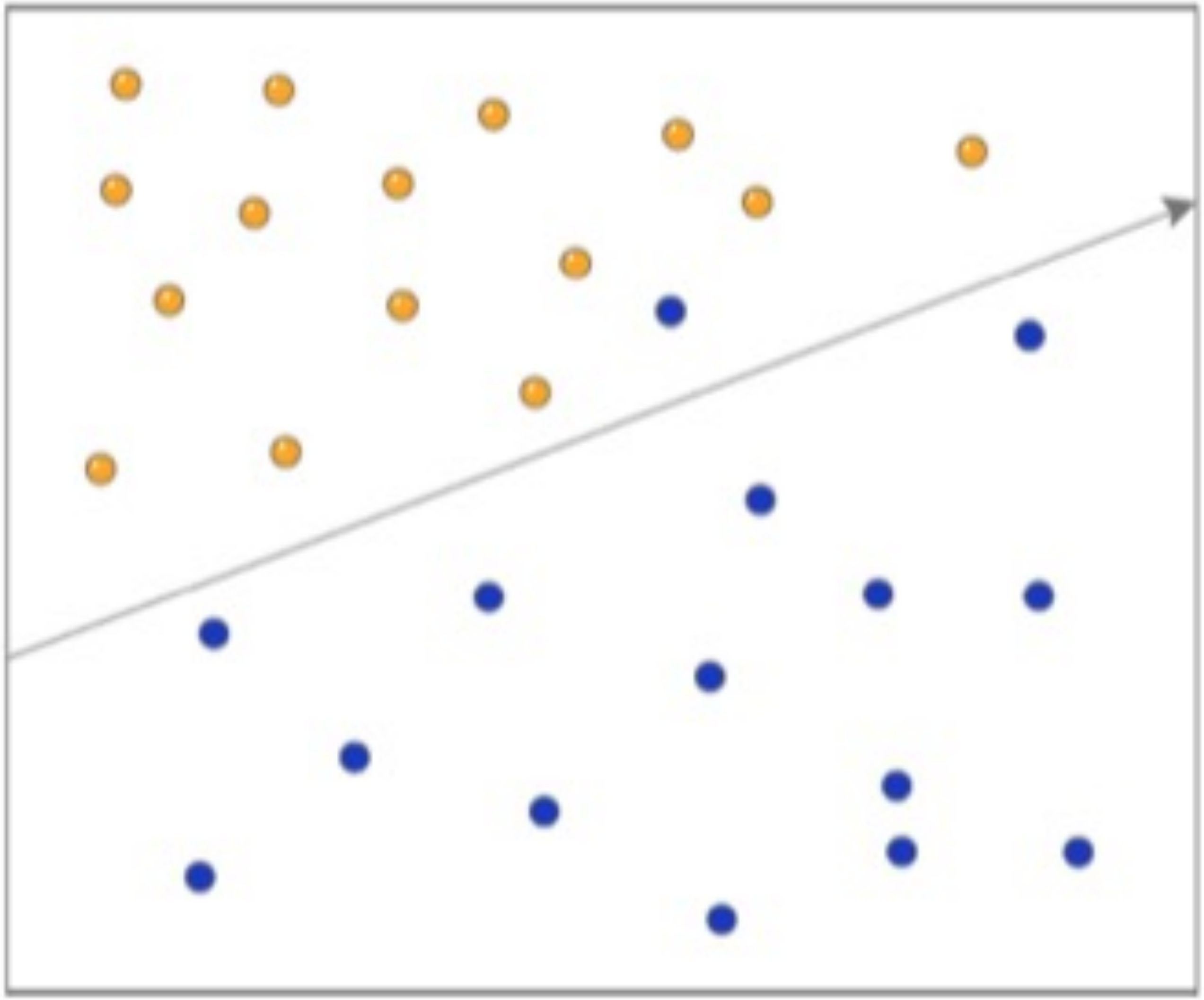
- Q: Naïve Bayes and Logistic Regression

Homework

- Homework tbd

Logistic regression

- Despite its name, performs classification not regression
- Naive Bayes and logistic regression are both considered to be linear models that create a linear decision boundary between classes
- The line is a linear combination of the X predictors



Classification

- Naive Bayes can perform multi-class classification
- Logistic regression is more suited to binary classification
- However, sklearn will perform OvA for logistic regression if the target has more than 2 classes
- OvA, One verses All, builds n classifiers for n classes
- Each classifier classes “one” class versus the rest

Example

- 20newsgroup data (see online notebook)
- 20 categories of news articles, 4 selected in the notebook
- Notebook also demos the Pipeline feature of sklearn

Code 21.0.1 — Logistic Regression. 20newsgroup data

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model.logistic import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score, log_loss

pipe1 = Pipeline([
        ('tfidf', TfidfVectorizer()),
        ('logreg', LogisticRegression(multi_class='multinomial',
                solver='lbfgs', class_weight='balanced')),
])

pipe1.fit(twenty_train.data, twenty_train.target)
```

Logistic Regression parameters

- See: sklearn.linear_model.LogisticRegression

- multi-class='multinomial' to set up the algorithm for this data
- class_weight='balanced' since the data is evenly distributed by class; this option is useful when the data set is unbalanced
- solver='lbfgs' is a good choice for multiclass problems; read about other solvers in the sklearn documentation; 'lbfgs' refers to an optimization algorithm, L-BFGS (Broyden-Fletcher-Goldfard-Shanno) that uses less computer memory.

Predict and evaluate

Code 21.0.2 — Logistic Regression. Predict and Evaluate

```
# evaluate on test data
twenty_test = fetch_20newsgroups(subset='test', categories=categories,
      shuffle=True, random_state=42)
pred = pipe1.predict(twenty_test.data)

from sklearn import metrics
print(metrics.classification_report(twenty_test.target, pred,
     target_names=twenty_test.target_names))

print("Confusion matrix:\n",
       metrics.confusion_matrix(twenty_test.target, pred))

import numpy as np
print("\nOverall accuracy: ", np.mean(pred==twenty_test.target))
```

Results

	precision	recall	f1-score	support
alt.atheism	0.95	0.81	0.87	319
comp.graphics	0.85	0.96	0.90	389
sci.med	0.93	0.88	0.90	396
soc.religion.christian	0.90	0.94	0.92	398
accuracy			0.90	1502
macro avg	0.91	0.90	0.90	1502
weighted avg	0.91	0.90	0.90	1502

```
Confusion matrix:
 [[258  13  14  34]
 [  3 374   6   6]
 [  5  41 347   3]
 [  6  11   5 376]]

Overall accuracy:  0.9021304926764314
```

Probabilities

- Extract the probabilities for each class for the first 5 test
- Notice the 3rd had no probs over .5, highest was .39

```
probs = pipe1.predict_proba(twenty_test.data)
probs[:5]

# output:
array([[0.14242452, 0.1643475 , 0.5691282 , 0.12409978],
       [0.0410866 , 0.03622316, 0.88370887, 0.03898138],
       [0.28678063, 0.10662445, 0.39017533, 0.21641959],
       [0.91486571, 0.02017099, 0.02211033, 0.04285297],
       [0.13633055, 0.06384863, 0.10635021, 0.6934706 ]])
```

Probability, odds, and log odds

- Played 10 games, won 7
- Odds is a ratio wins/losses, range [0, infinity)

$$odds = \frac{number\ \ of\ \ wins}{number\ \ of\ \ losses} = \frac{7}{3}$$

- Probability is a percentage of wins, range [0, 1]

$$probability = \frac{number\ \ of\ \ wins}{number\ \ of\ \ games} = \frac{7}{10}$$

Convert odds to probability

$$probability = \frac{odds}{1 + odds}$$

log odds

- the coefficient in logistic regression is the change in the log odds of y for a one-unit change in predictor x
- Log odds is log(odds)

log odds versus probability

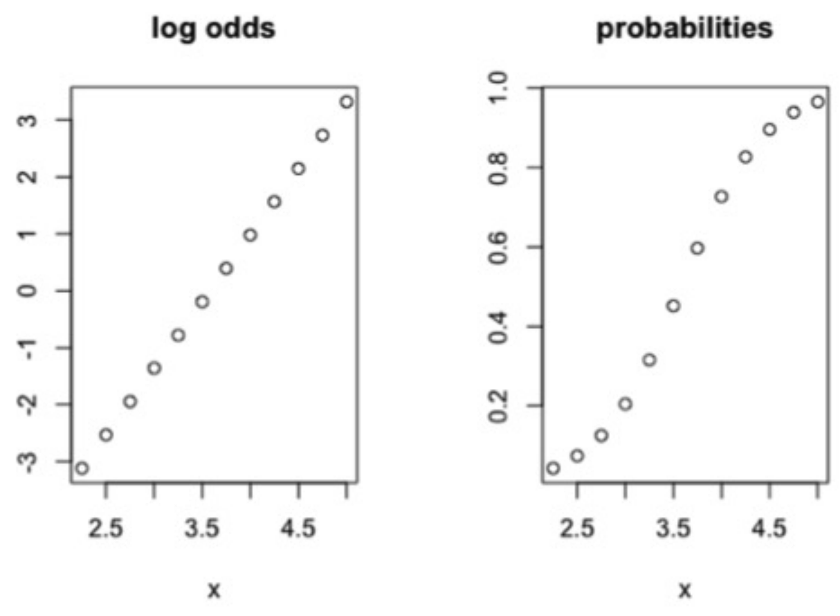


Figure 6.5: Log Odds versus Probability

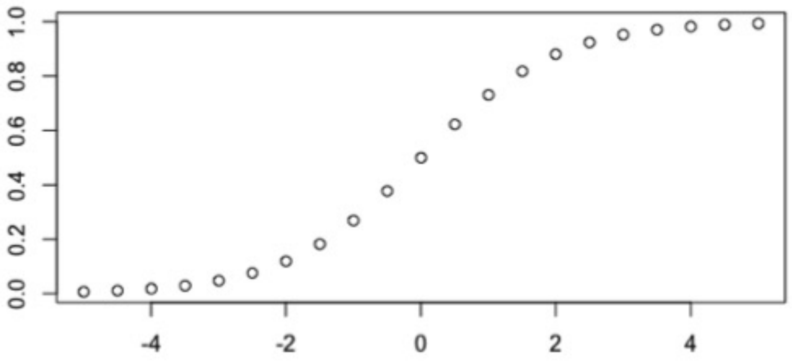
X	Log Odds	Probability
2.5	-2.53	0.07
3.0	-1.36	0.20
3.5	-0.19	0.45
4.0	0.977	0.73
4.5	2.147	0.89

Table 6.1: Log Odds and Probability for Plasma Data

The algorithm

- Linear regression had target values in infinite range +/-
- Logistic regression target is between [0, 1] to reflect the probability of the positive class
- The sigmoid, aka logistic function, does this:

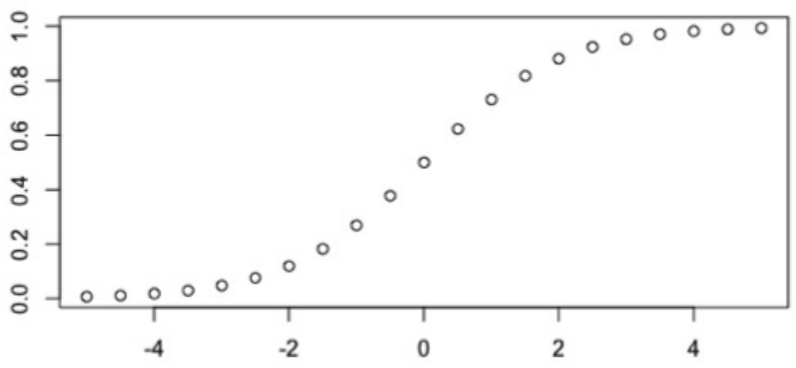
$$f(x) = \frac{1}{1+e^{-x}} \tag{6.10}$$



Predicting

- A cut-off point, like 0.5 is chosen
- Values > 0.5 are 1, others are 0

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6.10}$$



Logistic regression is a linear model

- because the log odds is a linear function of the parameters

$$log \frac{p(x)}{1-p(x)} = w_o + w_1 x \tag{6.11}$$

Solving for p gives us the logistic function:

$$p(x) = \frac{e^{-(w_0+w_1 x)}}{1+e^{-(w_0+w_1 x)}} = \frac{1}{1+e^{-(w_0+w_1 x)}} \tag{6.12}$$

Likelihood v. probability

- Probability $P(O \mid \theta)$
  - O is observed outcomes
  - Theta describes the underlying model, like 70%
- What if you don’t know theta?
- Likelihood $L(\theta \mid O)$
- Two ways of describing same phenomenon
- P in range [0, 1]
- L in range [0, inf)

Loss function for logistic regression

- Start with the likelihood

$$L(w_0, w_1) = \prod_{i=1}^{n} f(x_i)^{y_i} \, (1 - f(x_i))^{1-y_i}$$

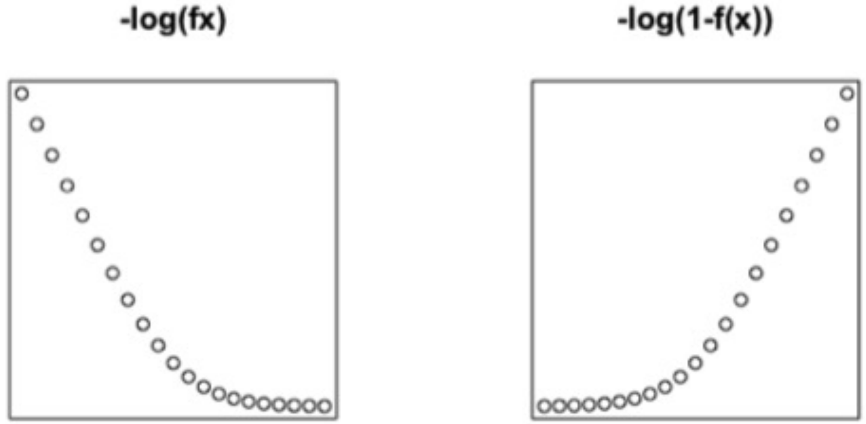
- One term above always reduces to 1
- The log likelihood is a simpler computation:

$$\ell = \sum_{i=1}^{n} y_i \, log f(x_i) + (1 - y_i) \, log(1 - f(x_i))$$

log likelihood

- For a single instance: $\ell = y\ log f(x) + (1-y)\ log(1-f(x))$
- gives a convex loss function

$$\mathcal{L} = -log(f(x))\ if\ y = 1 \qquad \mathcal{L} = -log(1-f(x))\ if\ y = 0 \qquad (6.17)$$



Loss function

$$\mathcal{L} = -\left[\sum_{i=i}^{N} y_i log(f(x_i)) \quad + \quad (1-y_i) log(1-f(x_i))\right] \tag{6.18}$$

where f(x) =

$$f(x) = \frac{1}{1+e^{-(w^T x)}} \tag{6.19}$$

- solve using an optimization method like gradient descent

Naïve Bayes v. Logistic Regression

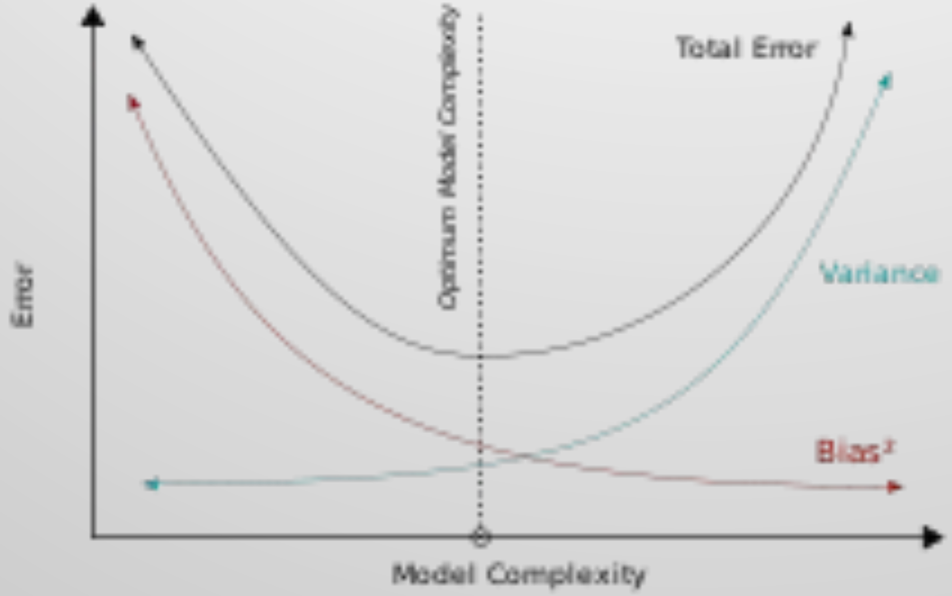
- See ‘sarcasm’ notebook online
- Data: combination of headlines from The Onion and a real news source, evenly divided
- Both NB and LogReg got about 85% accuracy
- Classification is hard on small text items, there’s not much there for the classifier to learn
- Both classifiers have high bias, low variance, with NB having higher bias

Naïve Bayes v. Logistic Regression

- NB is considered a generative classifier because it learns the parameters P(Y) as well as P(X|Y), which generated the data
- Logistic Regression is considered a discriminative classifier because it directly learns P(Y|X) from the data

Bias-variance tradeoff

- Bias is the tendency of an algorithm to make assumptions about the shape of the data
- Variance is the sensitivity of an algorithm to noise in the data



overfitting and underfitting

- A model that is too simple will have higher bias and lower variance (underfit)
- A model that is overly complex will have lower bias and higher variance (overfit)

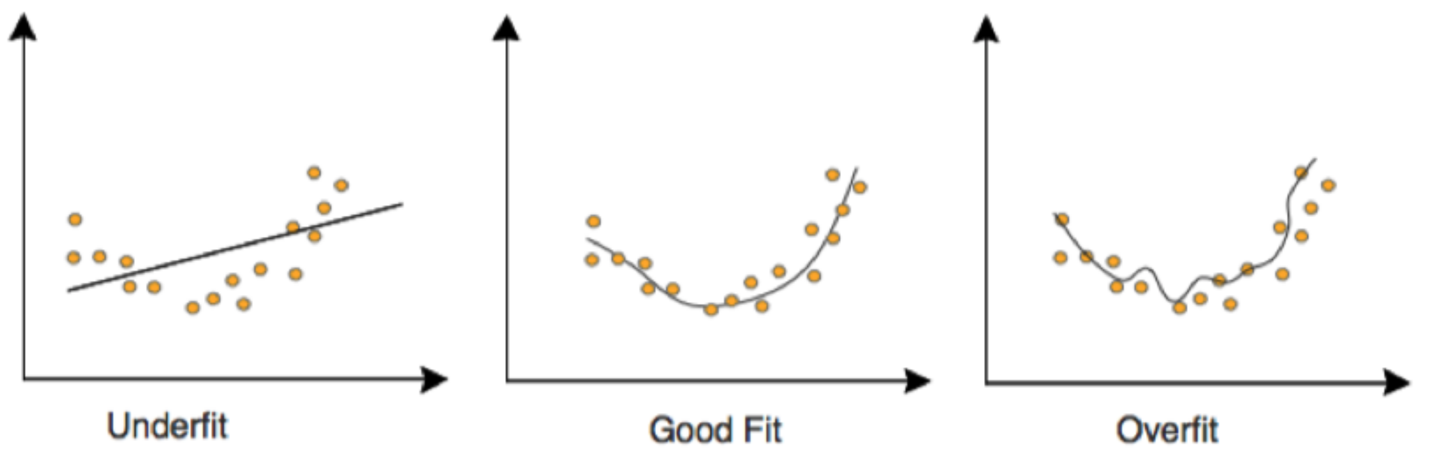


Figure 21.5: Overfitting versus Underfitting

Code Examples

See Part 5 Chapter 21

- logistic regression on the 20 news data
- logistic regression on the spam data
- logistic regression on the sarcasm data

Essential points to note

- Logistic regression performs well when the classes are linearly separable
- Logistic regression has high bias, but not as much as Naïve Bayes
- Logistic regression will often outperform Naïve Bayes on larger data sets

To Do

- Quiz on Naïve Bayes and Logistic Regression
- Homework: tbd

Next topic

Neural Networks