Machine Learning & Artificial Intelligence for Data Scientists: Classification (Part2)

Ke Yuan
https://kyuanlab.org/
School of Computing Science

Logistic Regression

- Alternative is to directly model $P(T_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = f(\mathbf{x}_{\text{new}}; \mathbf{w})$ with some parameters $\mathbf{w}$.

- We've seen $f(\mathbf{x}_{\text{new}}; \mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}}$ before – can we use it here?
  - No – *output is unbounded and so can't be a probability.*

- But, can use $P(T_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{w}) = h(f(\mathbf{x}_{\text{new}}; \mathbf{w}))$ where $h(\cdot)$ *squashes* $f(\mathbf{x}_{\text{new}}; \mathbf{w})$ to lie between 0 and 1 – a probability.

Recap on probability

- Discrete v continuous.
- Probabilities and densities.
- Joint probabilities and densities.
- Independence.
- Conditioning.

Random variables

If I toss a coin and assign the variable X the value 1 if the coin lands heads and 0 if it lands tails, X is a random variable.

We don't know which value X will take but we do know the possible values and how likely they are.

Discrete and continuous RVs

- Random events with outcomes that we can count: Discrete.
  - Coin toss.
  - Rolling a die.
  - Next word in a document.
  - Number of emails sent in a day.

- Random events with outcomes that we cannot count Continuous.
  - Winning time in Olympic 100m.

Discrete and continuous RVs

Definitions

Random variables given capital letters - X, Y.

Lower case letters used for values they can take - x, y.

Discrete RVs

Discrete RVs defines by probabilities of different events taking place. E.g. probability of random variable X taking value x:

$$P(X = x)$$

For example, fair coin:

$$P(X = 1) = 0.5, \; P(X = 0) = 0.5$$

Die:

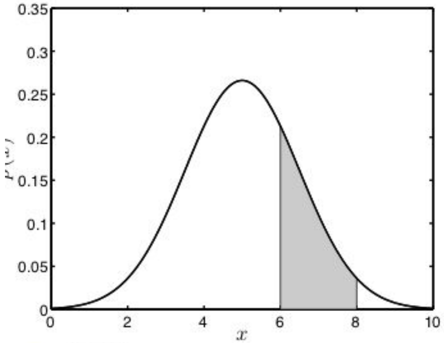
$$P(Y = y) = 1/6$$

Probabilities are constrained:

$$0 \leq P(Y = y) \leq 1, \; \sum_y P(Y = y) = 1.$$

Continuous RVs

- Don't define probabilities of particular outcomes as we can't count them!
- Instead define a density function $p(x)$:



$p(x)$ tells us how likely different values are. These are **not** probabilities!

- We can compute probabilities of ranges by computing the area under the curve:

$$P(6 \leq X \leq 8) = \int_{x=6}^{x=8} p(x)\ dx$$

- Densities are constrained:

$$p(x) \geq 0, \quad \int_{-\infty}^{\infty} p(x)\ dx = 1$$

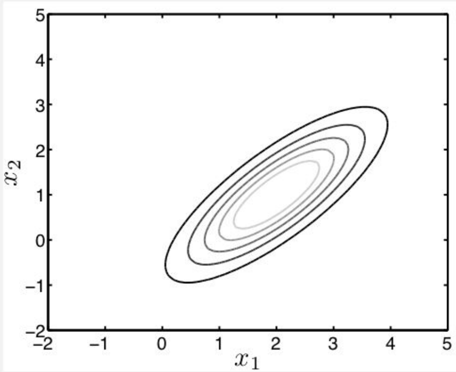
Joint probabilities and densities

Joint probabilities

For two discrete RVs, X and Y, $P(X = x, Y = y)$ is the probability that RV X has value x **and** RV Y has value y.

Joint densities

For two continuous RVs, x_0 and x_1, $p(x_0, x_1)$ is the joint density:



Dependence/Independence

- Let X be the random variable for the toss of a coin (1=heads, 0=tails)
- Let Y be the random variable for the rolling of a die.
- $P(X = 1, Y = 3)$ is the probability that I will roll a head **and** a 3.
- The outcome of X does not depend on Y.
- X and Y are independent.

$$P(X = x, Y = y) = P(X = x)P(Y = y)$$

Dependence/Independence

- Let X be the random variable for the event – I'm playing tennis (1=yes, 0=no)
- Let Y be the random variable for the event – It is raining (1=yes, 0=no)
- $P(X = 1, Y = 1)$ is the probability that I am playing and it is raining.
- The outcome of X **does** depend on Y.
- X and Y are dependent.

$$P(X = x, Y = y) \neq P(X = x)P(Y = y)$$

Conditioning

- Let X be the random variable for the event – I'm playing tennis (1=yes, 0=no)
- Let Y be the random variable for the event – It is raining (1=yes, 0=no)
- Because they are dependent, we can work with **conditional** probabilities.
- e.g. the probability that I am playing **given that** it is raining:

$$P(X = 1|Y = 1)$$

- Allows us to decompose the joint probability:

$$P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$$

Conditioning - continuous

Example 1:

$$p(t_n | x_n, \mathbf{w})$$

This is the density of t_n conditioned on a particular value of x and our model parameters $\mathbf{w}$.

Example 2:

$$P(9 \leq t_n \leq 10 | x_n, \mathbf{w})$$

This is the probability of t_n being between 9 and 10 conditioned on a particular value of x and our model parameters $\mathbf{w}$.

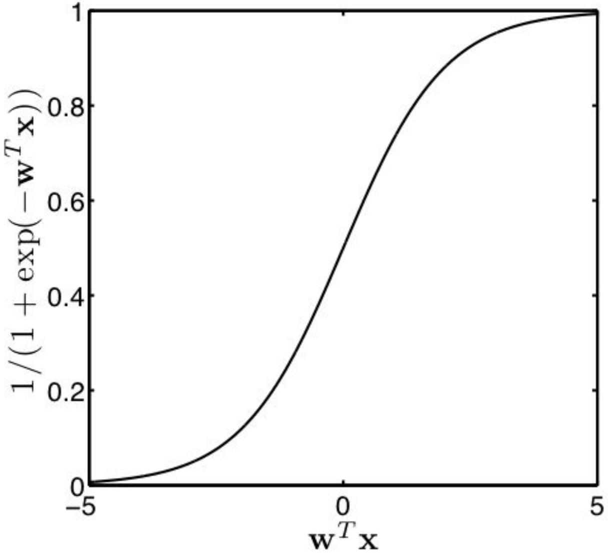
Notational nuance

Technically, we should write $p(t_n | X = x_n, W = \mathbf{w})$ but this becomes unwieldy and gets confusing (difference between X and $\mathbf{X}$?). So, we'll use $p(t_n | x_n, \mathbf{w})$.

Back to logistic regression: Sigmoid function

- For logistic regression (binary), we use the sigmoid function:

$$P(T_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{w}) = h(\mathbf{w}^{\mathsf{T}} \mathbf{x}_{\text{new}}) = \frac{1}{1 + \exp(-\mathbf{w}^{\mathsf{T}} \mathbf{x}_{\text{new}})}$$



Introducing Likelihood of a single label

$$\text{if} \quad t_n = 1, \qquad p(t_n = 1|\mathbf{x}_n, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_n)}$$

$$\text{if} \quad t_n = 0, \qquad p(t_n = 0|\mathbf{x}_n, \mathbf{w}) = 1 - p(t_n = 1|\mathbf{x}_n, \mathbf{w})$$

One formula for both scenarios

Likelihood function for nth data point

$$p(t_n|\mathbf{x}_n, \mathbf{w}) = p(t_n = 1|\mathbf{x}_n, \mathbf{w})^{t_n}(1 - p(t_n = 1|\mathbf{x}_n, \mathbf{w}))^{(1-t_n)}$$

Likelihood function for all data points

- Assuming data points are independent of each other

$$\text{Likelihood: } p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w})$$

$$\text{Log-Likelihood: } \log p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^{N} \log p(t_n|\mathbf{x}_n, \mathbf{w})$$

What about Loss?

$$\text{Log-Likelihood}(\mathbf{t}, \mathbf{X}; \mathbf{w}) = \sum_{n=1}^{N} \log p(t_n | \mathbf{x}_n, \mathbf{w})$$

$$\text{Loss}(\mathbf{t}, \mathbf{X}; \mathbf{w}) = -\text{Log-Likelihood}(\mathbf{t}, \mathbf{X}; \mathbf{w}) = -\sum_{n=1}^{N} \log p(t_n | \mathbf{x}_n, \mathbf{w})$$

Find the optimal parameters

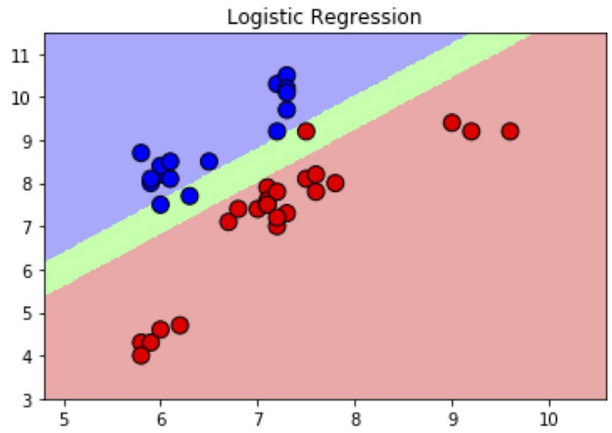
$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \operatorname{Loss}(\mathbf{t}, \mathbf{X}; \mathbf{w})$$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \operatorname{Likelihood}(\mathbf{t}, \mathbf{X}; \mathbf{w})$$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \operatorname{Log-Likelihood}(\mathbf{t}, \mathbf{X}; \mathbf{w})$$

```
In [59]: from sklearn.linear_model import LogisticRegression
clf = LogisticRegression().fit(X, t)
Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=t, cmap=cmap_bold,
            edgecolor='k', s=100)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Logistic Regression")
mean_cv_score = np.mean( cross_val_score(clf, X, t, cv=5) )
print("5-fold averae CV error:", 1-mean_cv_score)
```

```
5-fold averae CV error: 0.025000000000000022
```



Example on orange and lemon data

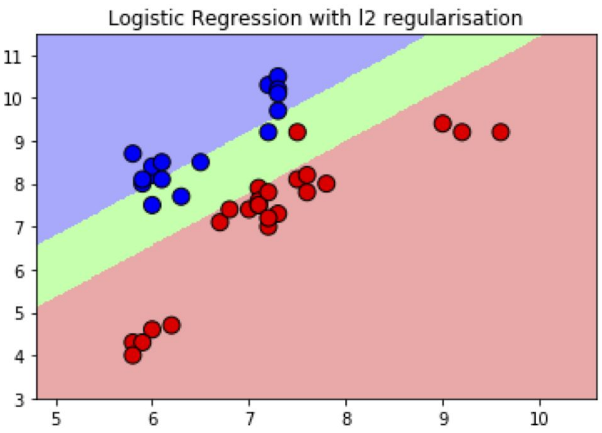
Can be regularised just the same as linear regression

$$\hat{\mathbf{w}}_{l2} = \underset{\mathbf{w}}{\operatorname{argmin}} \operatorname{Loss}(\mathbf{t}, \mathbf{X}; \mathbf{w}) + \frac{1}{C} \mathbf{w}^T \mathbf{w}$$

$$\hat{\mathbf{w}}_{l1} = \underset{\mathbf{w}}{\operatorname{argmin}} \operatorname{Loss}(\mathbf{t}, \mathbf{X}; \mathbf{w}) + \frac{1}{C} \sum_{d} |w_d|$$

In [53]:
```
parameters = {'C':cs}
logit_reg =  LogisticRegression(penalty='l2', tol=1e-5, max_iter=1e4)
clf = GridSearchCV(logit_reg, parameters, cv=5)
clf.fit(X,t)

Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
Z = Z.reshape(xx.shape) # Put the result into a color plot
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
plt.scatter(X[:, 0], X[:, 1], c=t, cmap=cmap_bold, edgecolor='k', s=100) # Plot al
so the training points
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Logistic Regression with l2 regularisation")
print("5-fold averae CV error:", 1-clf.best_score_)
```

```
5-fold averae CV error: 0.025000000000000022
```



In [49]:
```
from sklearn.svm import l1_min_c
from sklearn.model_selection import GridSearchCV
cs = l1_min_c(X, t, loss='log')*np.logspace(0, 7, 16)
print(cs)
```

```
[2.89435601e-02 8.47653998e-02 2.48247728e-01 7.27029358e-01
 2.12921058e+00 6.23570098e+00 1.82621518e+01 5.34833516e+01
 1.56633727e+02 4.58724513e+02 1.34344105e+03 3.93446133e+03
 1.15226388e+04 3.37457135e+04 9.88292004e+04 2.89435601e+05]
```

L2 regularised Logistic Regression

Performance Evaluations

- We've seen 2 classification algorithms.
- How do we choose?
  - Which algorithm?
  - Which parameters?
- Need performance indicators.
- We'll cover:
  - 0/1 loss.
  - ROC analysis (sensitivity and specificity)
  - Confusion matrices

0/1 loss

- 0/1 loss: proportion of times classifier is wrong.
- Consider a set of predictions $t_1, \ldots, t_N$ and a set of true labels $t_1^*, \ldots, t_N^*$.
- Mean loss is defined as:

$$\frac{1}{N}\sum_{n=1}^{N}\delta(t_n \neq t_n^*)$$

- ($\delta(a)$ is 1 if a is true and 0 otherwise)
- Advantages:
  - Can do binary or multiclass classification.
  - Simple to compute.
  - Single value.

0/1 loss

Disadvantage: Doesn't take into account class imbalance:

- We're building a classifier to detect a rare disease.
- Assume only 1% of population is diseased.
- Diseased: $t = 1$
- Healthy: $t = 0$
- What if we always predict healthy? ($t = 0$)
- Accuracy 99%
- But classifier is rubbish!

Sensitivity and specificity

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- True positives (TP) – the number of objects with $t_n^* = 1$ that are classified as $t_n = 1$ (diseased people diagnosed as diseased).
- True negatives (TN) – the number of objects with $t_n^* = 0$ that are classified as $t_n = 0$ (healthy people diagnosed as healthy).
- False positives (FP) – the number of objects with $t_n^* = 0$ that are classified as $t_n = 1$ (healthy people diagnosed as diseased).
- **False negatives (FN)** – the number of objects with $t_n^* = 1$ that are classified as $t_n = 0$ (diseased people diagnosed as healthy).

Sensitivity

$$S_e = \frac{TP}{TP + FN}$$

- The proportion of diseased people that we classify as diseased.
- The higher the better.
- In our example, $S_e = 0$.

Specificity

$$S_p = \frac{TN}{TN + FP}$$

- The proportion of healthy people that we classify as healthy.
- The higher the better.
- In our example, $S_p = 1$.

Optimising sensitivity and specificity

- We would like both to be as high as possible.
- Often increasing one will decrease the other.
- Balance will depend on application:
- e.g. diagnosis:
  - We can probably tolerate a decrease in specificity (healthy people diagnosed as diseased)....
  - ...if it gives us an increase in sensitivity (getting diseased people right).

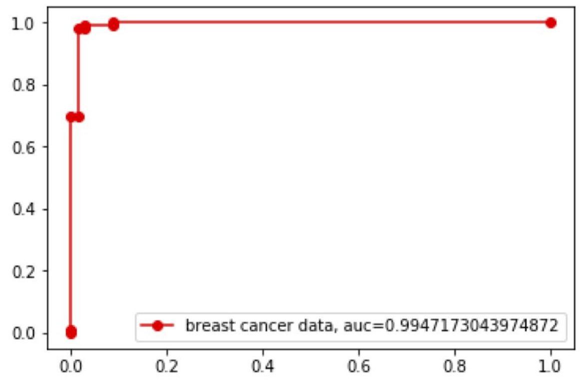
Receiver Operating Characteristic (ROC)

- Many classification algorithms involve setting a threshold.
- e.g. Logistic Regression:

$$p(t_{new} = 1 | \mathbf{x}_{new}, \mathbf{w}) > 0.5$$

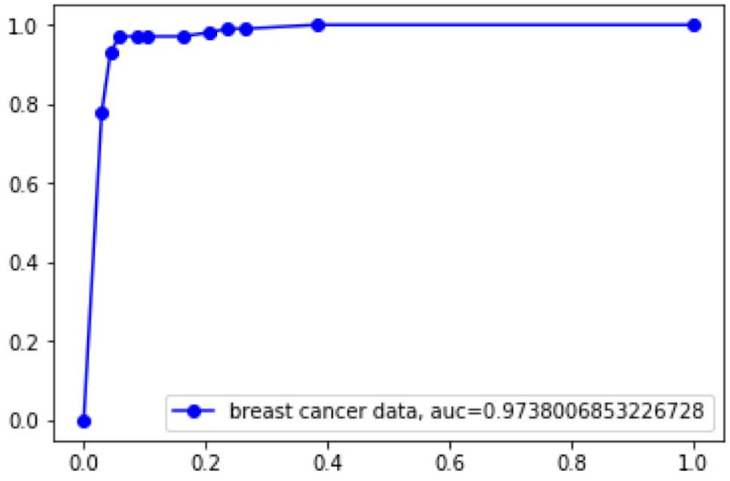
- Implies a threshold of zero (sign function)
- However, we could use any threshold we like....
- The *Receiver Operating Characteristic (ROC) curve* shows how S_e and $1 - S_p$ vary as the threshold changes.

In [16]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.datasets import load_breast_cancer

breast_cancer = load_breast_cancer()
X = breast_cancer.data
t = breast_cancer.target


X_train, X_test, y_train, y_test = train_test_split(X,t,test_size=0.30, random_sta
te=123)
clf1 = LogisticRegression().fit(X_train, y_train)

y_pred1 = clf1.predict(X_test)
y_pred_proba1 = clf1.predict_proba(X_test)[:,1]
fpr1, tpr1, _ = metrics.roc_curve(y_test,  y_pred_proba1)
auc1 = metrics.roc_auc_score(y_test, y_pred_proba1)
plt.plot(fpr1,tpr1,'ro-',label="breast cancer data, auc="+str(auc1))
plt.legend(loc=4)
plt.show()
```



Try it on a breast cancer dataset
Plot ROC of a Logistic Regression model

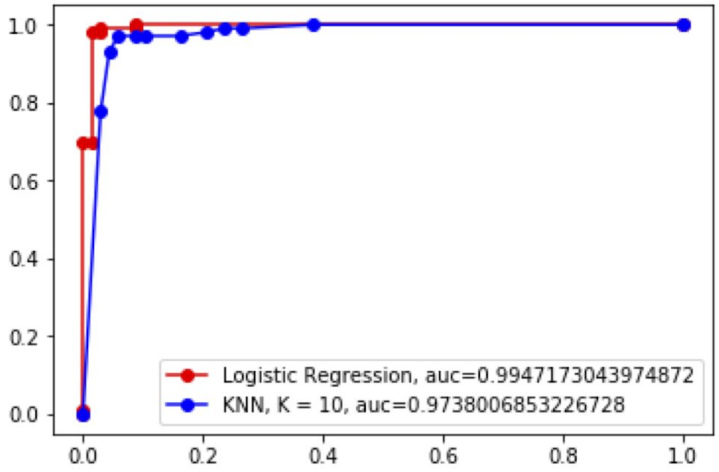
In [19]:
```
clf2 = KNeighborsClassifier(10).fit(X_train, y_train)

y_pred2 = clf2.predict(X_test)
y_pred_proba2 = clf2.predict_proba(X_test)[:,1]
fpr2, tpr2, _ = metrics.roc_curve(y_test,  y_pred_proba2)
auc2 = metrics.roc_auc_score(y_test, y_pred_proba2)
plt.plot(fpr2,tpr2,'bo-',label="breast cancer data, auc="+str(auc2))
plt.legend(loc=4)
plt.show()
```



The same data with KNN (K =10)

Overlay two ROC plots

In [21]:
```
plt.plot(fpr1,tpr1,'ro-',label="Logistic Regression, auc="+str(auc1))
plt.plot(fpr2,tpr2,'bo-',label="KNN, K = 10, auc="+str(auc2))
plt.legend(loc=4)
plt.show()
```



What is happening?

Can I have 7 volunteers?

Confusion matrix

The quantities we used to compute S_e and S_p can be neatly summarised in a table:

		True class	
		1	0
Predicted class	1	TP	FP
	0	FN	TN

- This is known as a *confusion matrix*
- It is particularly useful for multi-class classification.
- Tells us where the mistakes are being made.
- Note that normalising columns gives us S_e and S_p

Confusion matrix, example

- 20 newsgroups data.
- Thousands of documents from 20 classes (newsgroups)

True class

Predicted class	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
						⋮						
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting 'confused' between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc
  - 16: talk.religion.misc
  - 20: soc.religion.christian
- Maybe these should be just one class?
- Maybe we need more data in these classes?
- Confusion matrix helps us direct our efforts to improving the classifier.