Lecture 6
Linear Classification & Logistic Regression

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN (WITH MODIFICATION BY YAO WANG)

Learning Objectives

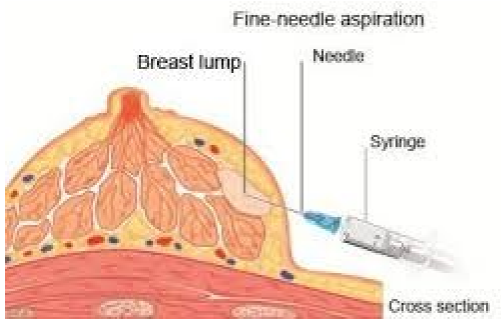
- Formulate a machine learning problem as a classification problem
  - Identify features, class variable, training data
- Visualize classification data using a scatter plot.
- Describe a linear classifier as an equation and on a plot.
  - Determine visually if data is perfect linearly separable.
- Formulate a classification problem using logistic regression
  - Binary and multi-class
  - Describe the logistic and soft-max function
  - Logistic function to approximate the probability
- Derive the loss function for ML estimation of the weights in logistic regression
- Use sklearn packages to fit logistic regression models
- Measure the accuracy of classification
- Adjust threshold of classifiers for trading off types of classification errors. Draw a ROC curve.
- Perform LASSO regularization for feature selection

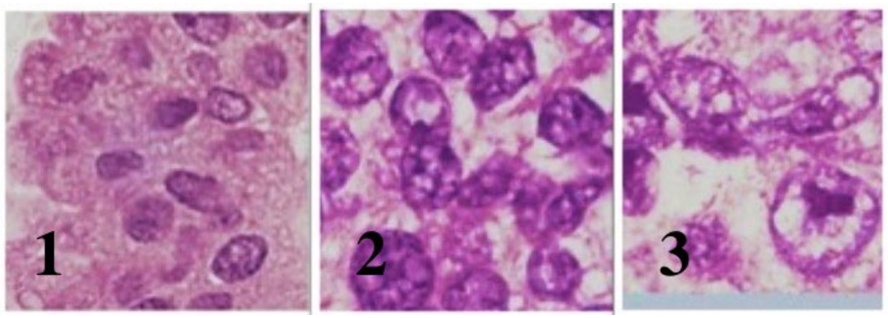
Outline

- ❑Motivating Example: Classifying a breast cancer test
- ❑Linear classifiers
- ❑Logistic regression
- ❑Fitting logistic regression models
- ❑Measuring accuracy in classification

Diagnosing Breast Cancer

- Fine needle aspiration of suspicious lumps
- Cytopathologist visually inspects cells
  - Sample is stained and viewed under microscope
- Determines if cells are benign or malignant and furthermore provides grading if malignant
- Uses many features:
  - Size and shape of cells, degree of mitosis, differentiation, …
- Diagnosis is not exact
- If uncertain, use a more comprehensive biopsy
  - Additional cost and time
  - Stress to patient
- Can machine learning provide better rules?





Grades of carcinoma cells
http://breast-cancer.ca/5a-types/

Demo on Github

- Github: https://github.com/sdrangan/introml/blob/master/logistic/breast_cancer.ipynb

Breast Cancer Diagnosis via Logistic Regression

In this demo, we will see how to visualize training data for classification, plot the logistic function and perform logistic regression. As an example, we will use the widely-used breast cancer data set. This data set is described here:

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin

Each sample is a collection of features that were manually recorded by a physician upon inspecting a sample of cells from fine needle aspiration. The goal is to detect if the cells are benign or malignant.

Loading and Visualizing the Data

We first load the packages as usual.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets, linear_model, preprocessing
%matplotlib inline
```

Next, we load the data. It is important to remove the missing values.

```
names = ['id','thick','size_unif','shape_unif','marg','cell_size','bare',
         'chrom','normal','mit','class']
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/' +
                 'breast-cancer-wisconsin/breast-cancer-wisconsin.data',
                names=names,na_values='?',header=None)
df = df.dropna()
df.head(6)
```

	id	thick	size_unif	shape_unif	marg	cell_size	bare	chrom	normal	mit	class

Data

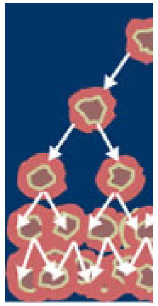
- ❑Univ. Wisconsin study, 1994
- ❑569 samples
- ❑10 visual features for each sample
- ❑Ground truth determined by biopsy

- ❑First publication: O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	442524

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

Loading The Data

```
names = ['id','thick','size_unif','shape_unif','marg','cell_size','bare',
         'chrom','normal','mit','class']
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/' +
                 'breast-cancer-wisconsin/breast-cancer-wisconsin.data',
                names=names,na_values='?',header=None)
df = df.dropna()
df.head(6)
```

	id	thick	size_unif	shape_unif	marg	cell_size	bare	chrom	normal	mit	class
0	1000025	5	1	1	1	2	1.0	3	1	1	2
1	1002945	5	4	4	5	7	10.0	3	2	1	2
2	1015425	3	1	1	1	2	2.0	3	1	1	2
3	1016277	6	8	8	1	3	4.0	3	7	1	2
4	1017023	4	1	1	3	2	1.0	3	1	1	2
5	1017122	8	10	10	8	7	10.0	9	7	1	4

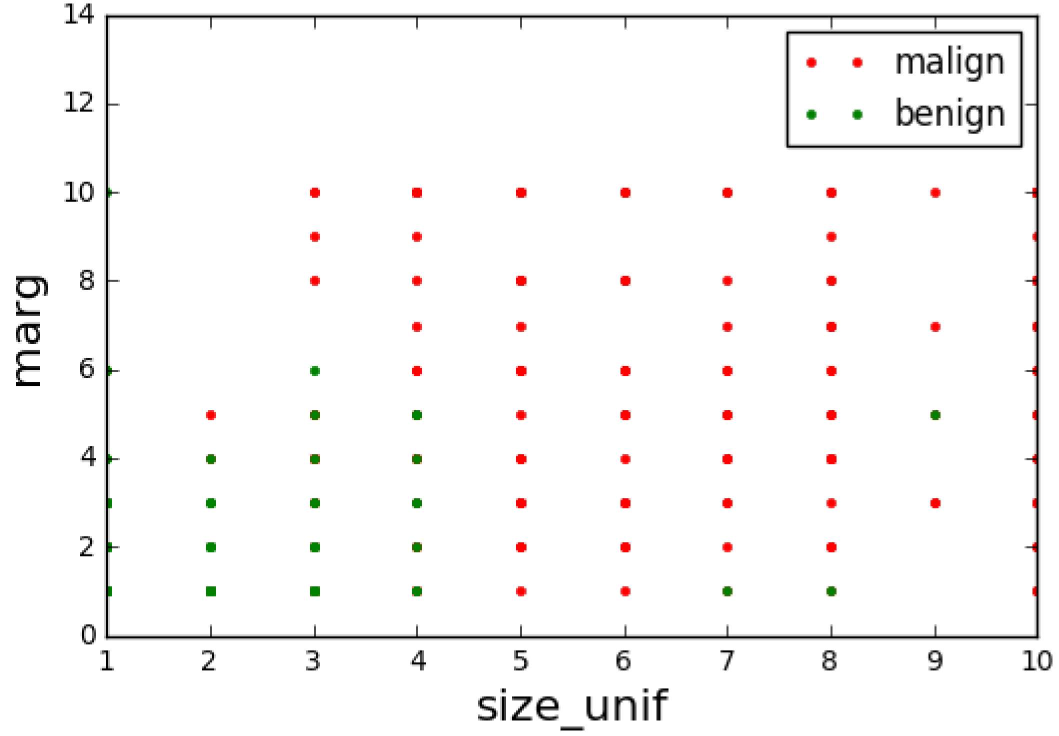
- ❑ Follow standard pandas routine
- ❑ All code in Lect06_Demo.ipynb

Drops missing samples

Class = 2 => benign
Class = 4 => malignant

See following for explanation of attributes
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names

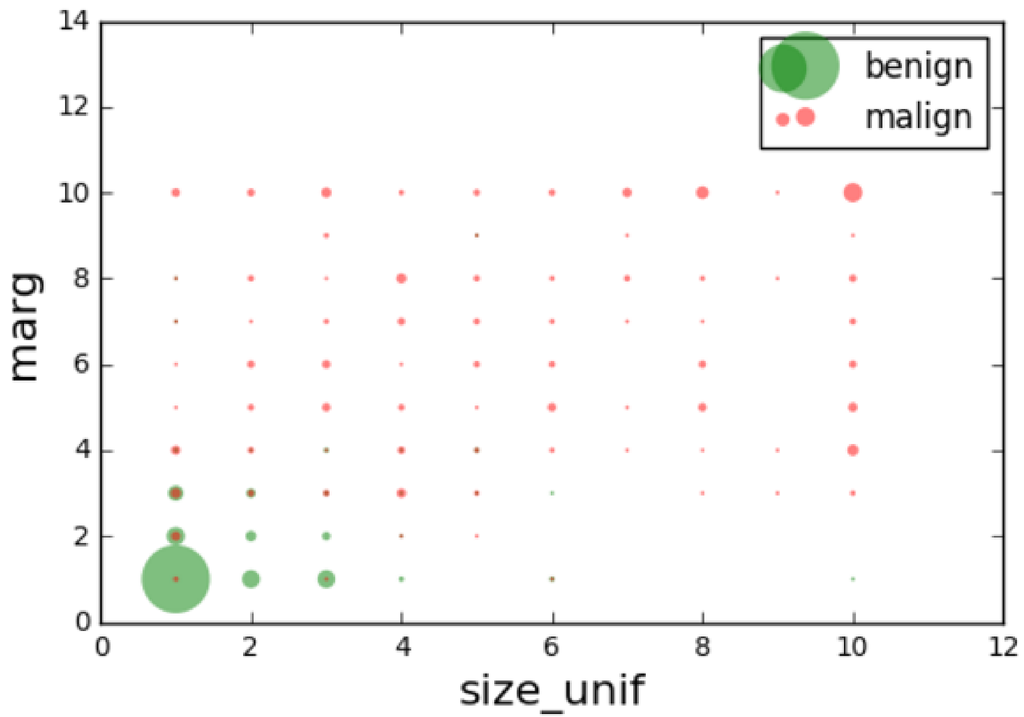
Visualizing the Data



- Scatter plot of points from each class
- Plot not informative
  - Many points overlap
  - Relative frequency at each point not visible

```
y = np.array(df['class'])
xnames =['size_unif','marg']
X = np.array(df[xnames])

Iben = np.where(y==2)[0]
Imal = np.where(y==4)[0]

plt.plot(X[Imal,0],X[Imal,1],'r.')
plt.plot(X[Iben,0],X[Iben,1],'g.')
plt.xlabel(xnames[0], fontsize=16)
plt.ylabel(xnames[1], fontsize=16)
plt.ylim(0,14)
plt.legend(['malign','benign'],loc='upper right')
```

Improving the Plot

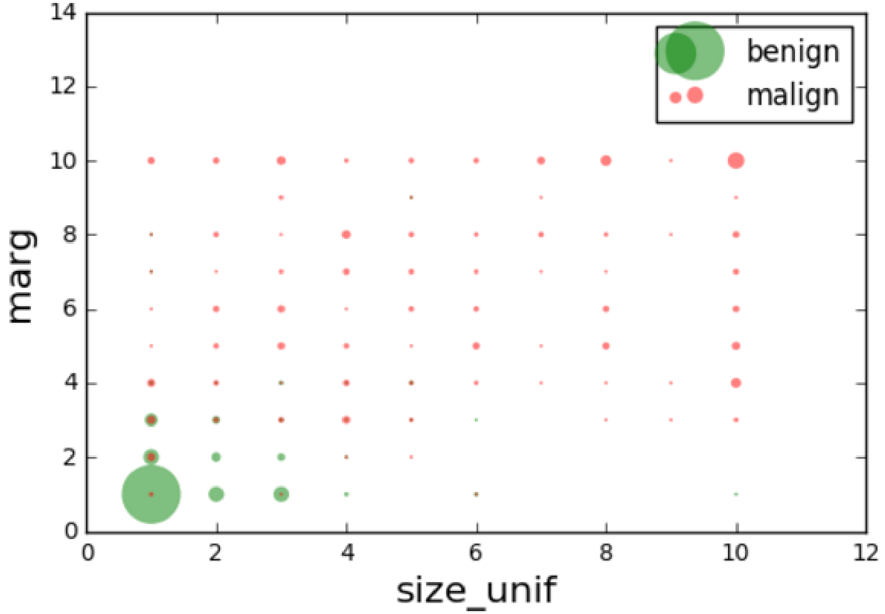


- Make circle size proportional to count
- Many gymnastics to make this plot in python

```
# Compute the bin edges for the 2d histogram
x0val = np.array(list(set(X[:,0]))).astype(float)
x1val = np.array(list(set(X[:,1]))).astype(float)
x0, x1 = np.meshgrid(x0val,x1val)
x0e= np.hstack((x0val,np.max(x0val)+1))
x1e= np.hstack((x1val,np.max(x1val)+1))

# Make a plot for each class
yval = [2,4]
color = ['g','r']
for i in range(len(yval)):
    I = np.where(y==yval[i])[0]
    cnt, x0e, x1e = np.histogram2d(X[I,0],X[I,1],[x0e,x1e])
    x0, x1 = np.meshgrid(x0val,x1val)
    plt.scatter(x0.ravel(), x1.ravel(), s=2*cnt.ravel(),alpha=0.5,
                c=color[i],edgecolors='none')
plt.ylim([0,14])
plt.legend(['benign','malign'], loc='upper right')
plt.xlabel(xnames[0], fontsize=16)
plt.ylabel(xnames[1], fontsize=16)
```

In-Class Exercise

- Get into groups
  - At least one must have a laptop with jupyter notebook
- Determine a classification rule
  - Predict class label from the two features
- Test in python
  - Make the predictions
  - Measure the accuracy

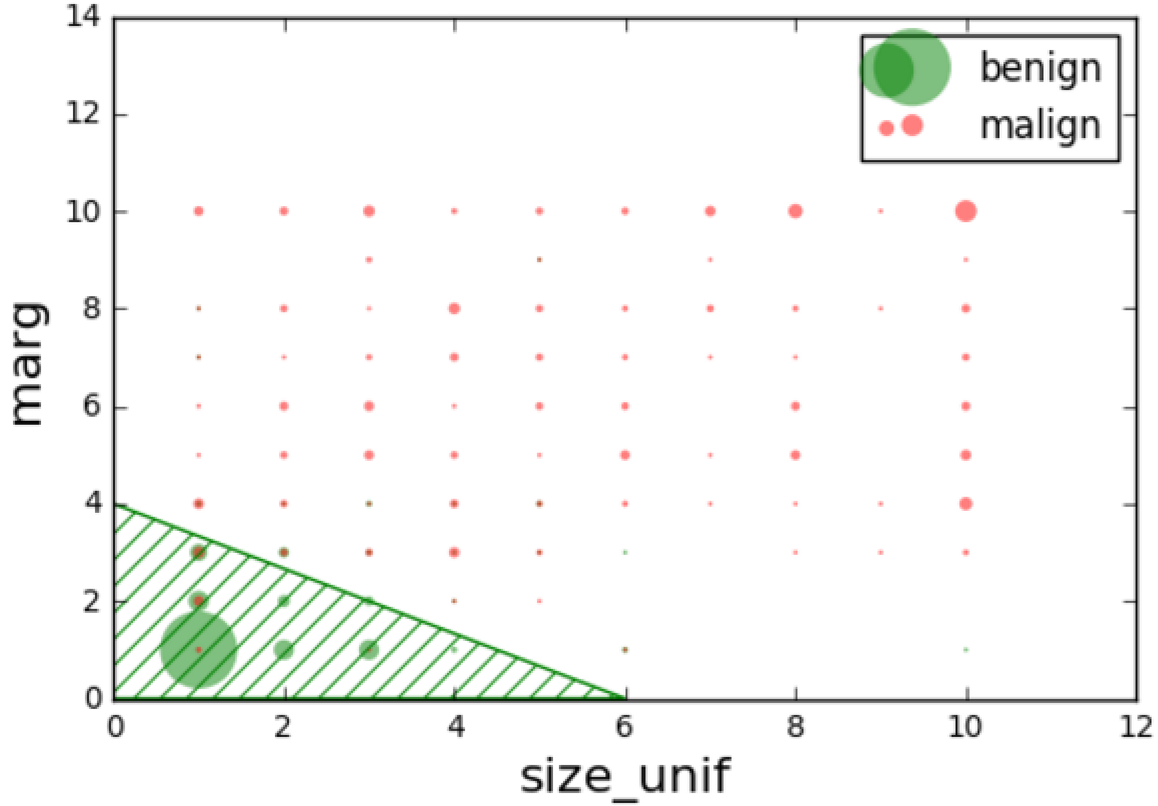


In-Class Exercise

Based on the above plot, what would be a good "classifer" using the two features. That is, write a function that makes a prediction `yhat` of the class label `y`. Code up your classifier function. Measure the accuracy of the classifier on the data. What percentage error does your classifier get?

```
# TODO
```

A Possible Classification Rule



- From inspection, benign if:

$$\text{marg} + \frac{2}{3}(\text{size\_unif}) < 4$$

- Classification rule from linear constraint
- What are other possible classification rules?
- Every rule misclassifies some points
- What is optimal?

Mangasarian's Original Paper

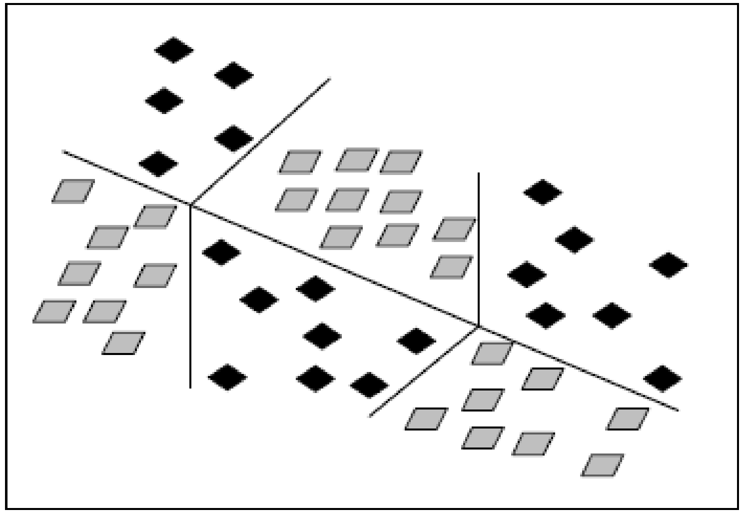


Figure 2.2 - Decision boundaries generated by MSM-T. Dark objects represent benign tumors while light object represent malignant ones.

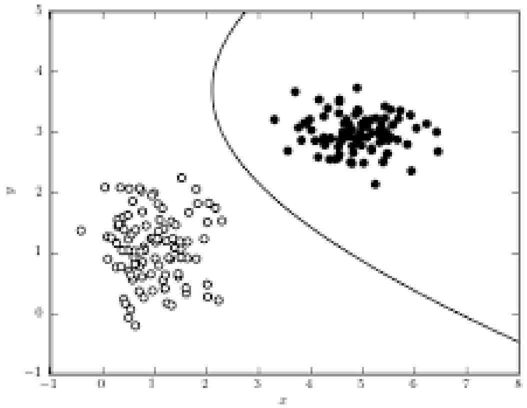
- Proposes Multisurface method – Tree (MSM-T)
  - Decision tree based on linear rules in each step
- Fig to left from
  - Pantel, "Breast Cancer Diagnosis and Prognosis," 1995
- Best methods today use neural networks
- This lecture will look at linear classifiers
  - These are much simpler
  - Do not provide same level of accuracy
- But, building block to more complex classifiers

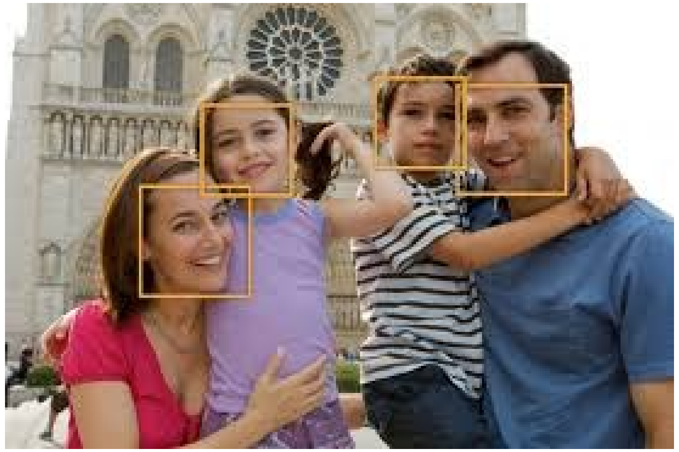
Classification

- Given features $\boldsymbol{x}$, determine its class label, $y = 1, \ldots, K$
- Binary classification: $y = 0$ or 1
- Many applications:
  - Face detection: Is a face present or not?
  - Reading a digit: Is the digit 0,1,…,9?
  - Are the cells cancerous or not?
  - Is the email spam?
- Equivalently, determine classification function:

$$\hat{y} = f(\boldsymbol{x}) \in \{1, \ldots, K\}$$

  - Like regression, but with a discrete response
  - May index $\{1, \ldots, K\}$ or $\{0, \ldots, K-1\}$





Outline

- ❑Motivating Example: Classifying a breast cancer test
- ❑Linear classifiers
- ❑Logistic regression
- ❑Fitting logistic regression models
- ❑Measuring accuracy in classification

Linear Classifier

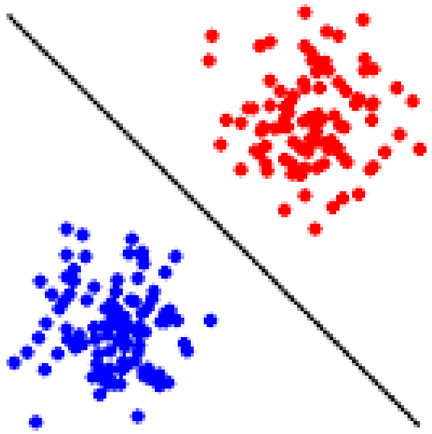
- General binary classification rule:

$$\hat{y} = f(x) = 0 \text{ or } 1$$

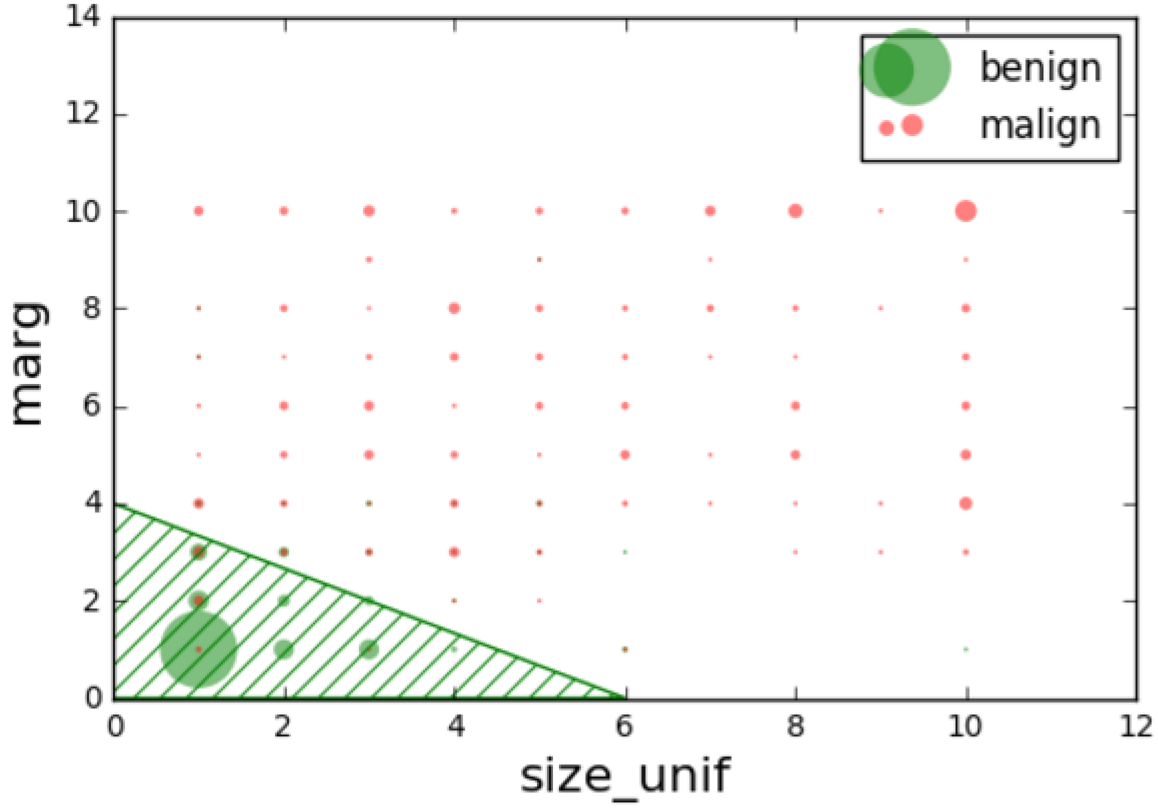
- Linear classification rule:
  - Take linear combination $z = w_0 + \sum_{j=1}^{d} w_d x_d$
  - Predict class from z

$$\hat{y} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

- Decision regions described by a half-space.
- $\boldsymbol{w} = (w_0, \dots, w_d)$ is called the weight vector



Breast Cancer example



- From inspection, benign if:

$$\text{marg} + \frac{2}{3}(\text{size\_unif}) < 4$$

- Classification rule from linear constraint
- What are other possible classification rules?
- Every rule misclassifies some points
- What is optimal?

Using Linear Regression on Two Features

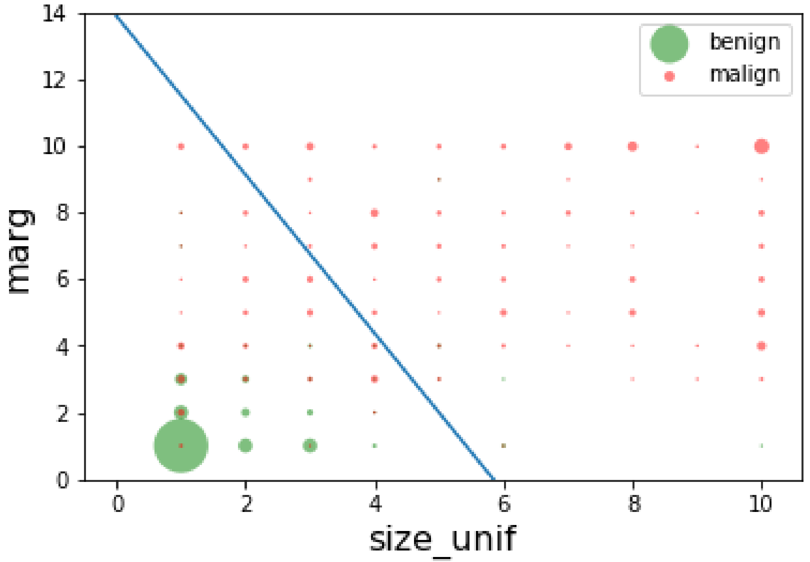
- Bad idea: Use linear regression
  - Labels are $y \in \{0,1\}$
  - Use linear model: $z = w_0 + w_1 x_1 + w_2 x_2$
  - Find linear fit so that $\sum_i (y_i - z_i)^2$ is minimized
  - Then threshold the linear fit:

$$\hat{y} = \begin{cases} 1 & z > 0.5 \\ 0 & z < 0.5 \end{cases}$$

- This yields the line:

$$w_0 + w_1 x_1 + w_2 x_2 = 0.5$$

- Why the line is not as expected?
- Should not use MSE as the optimization criterion!
  - Squared error is not related to classifier accuracy




```
yhat=regr.predict(X)
yhati= (yhat >=0).astype(int)
acc = np.mean(yhati == y)
print("Accuracy on training data using two features = %f" % acc)
```

```
Accuracy on training data using two features = 0.922401
```

Using linear classifier on all features

- Hard to visualize, but by setting $z = 0.5$, we get a hyper plane in high dimension

```
xnames = ['thick','size_unif','shape_unif','marg','cell_size','bare',
          'chrom','normal','mit']
X = np.array(df[xnames])

Xs = preprocessing.scale(X)
regr.fit(Xs,y1)
yhat=regr.predict(Xs)
yhati= (yhat >=0).astype(int)
acc = np.mean(yhati == y)
print("Accuracy on training data using 10 features = %f" % acc)
```

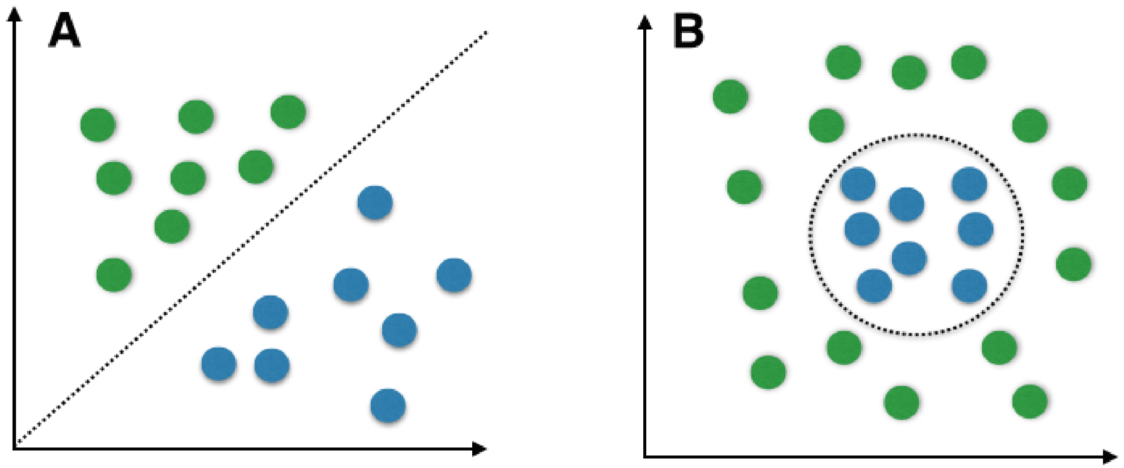
```
Accuracy on training data using 10 features = 0.960469
```

Go through the demo

- Up to linear regression

Linear vs. Non-Linear

- ❑Linear boundaries are limited
- ❑Can only describe very simple regions
- ❑But, serves as building block
  - Many classifiers use linear rules as first step
  - Neural networks, decision trees, …

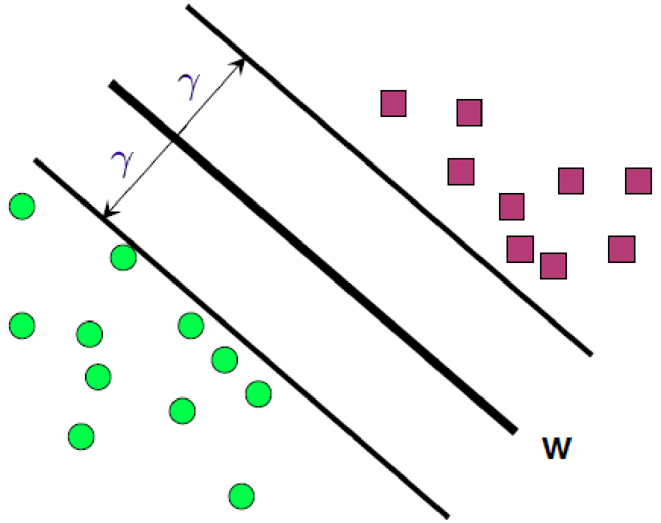


- ❑Breast cancer example:
  - Is the region linear or non-linear?

Perfect Linear Separability

- Given training data $(\boldsymbol{x}_i, y_i), i = 1, \ldots, N$
- Binary class label: $y_i = \pm 1$
- Perfectly linearly separable if there exists a $\boldsymbol{w} = (w_0, w_1, \ldots, w_d)$ s.t.
  - $w_0 + w_1 x_{i1} + \cdots w_d x_{id} > \gamma$ when $y_i = 1$
  - $w_0 + w_1 x_{i1} + \cdots w_d x_{id} < -\gamma$ when $y_i = -1$
- $\boldsymbol{w}$ is the separating hyperplane, γ is the margin
- Single equation form:

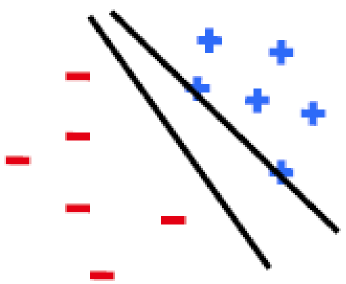
$$y_i(w_0 + w_1 x_{i1} + \cdots w_d x_{id}) > \gamma \text{ for all } i = 1, \ldots, N$$



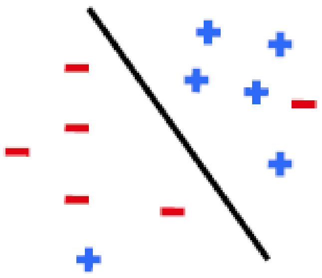
Most Data not Perfectly Separable

- Generally cannot find a separating hyperplane
- Always, some points that will be mis-classified
- Algorithms attempt to find “good” hyper-planes
  - Reduce the number of mis-classified points
  - Or, some similar metric
- Example: Look again at breast cancer data

Separable

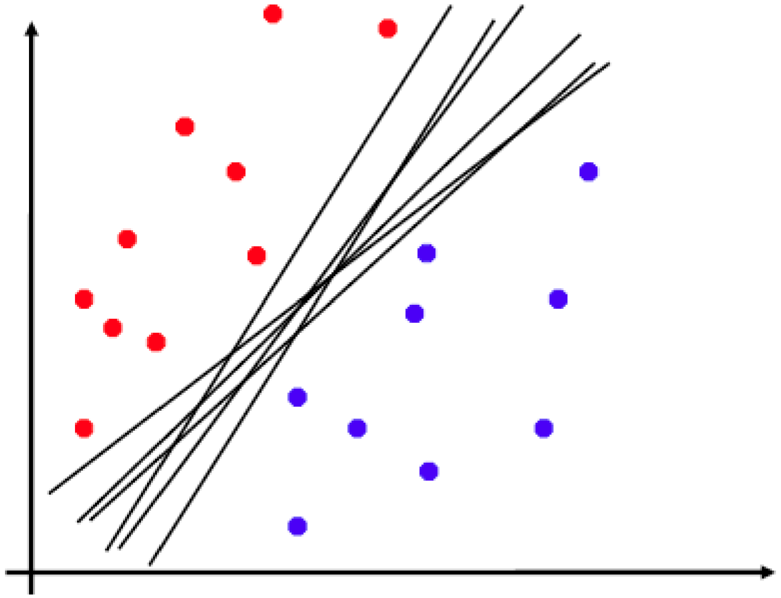


Non-Separable



Non-Uniqueness

- When one exists, separating hyper-plane is not unique
- Example:
  - If $\boldsymbol{w}$ is separating, then so is $\alpha\boldsymbol{w}$ for all $\alpha > 0$
- Fig. on right: Many separating planes
- Which one is optimal?



Outline

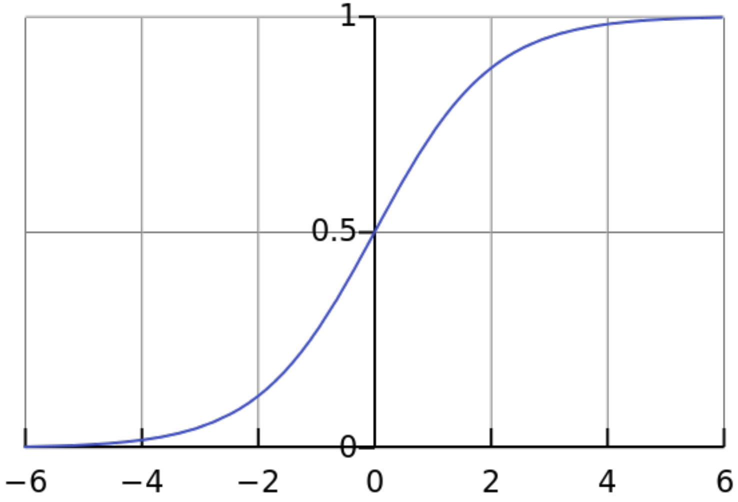
- ❑Motivating Example: Classifying a breast cancer test
- ❑Linear classifiers
- ❑Logistic regression
- ❑Fitting logistic regression models
- ❑Measuring accuracy in classification

Logistic Model for Binary Classification

- Binary classification problem: $y = 0, 1$
- Consider probabilistic model

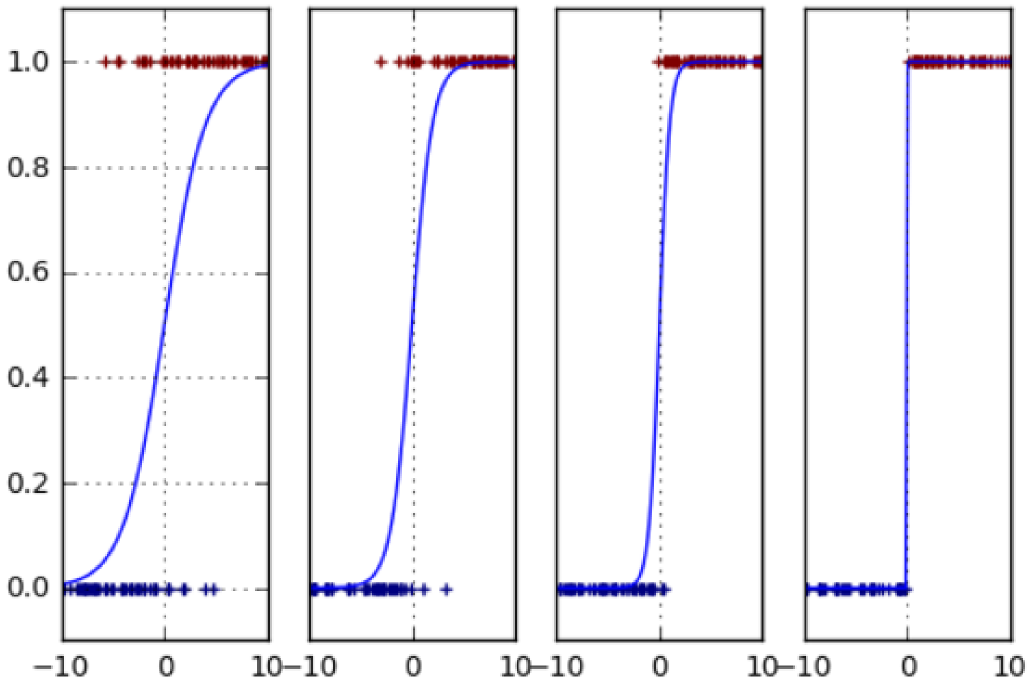
$$P(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-z}}, \qquad P(y = 0|\boldsymbol{x}) = \frac{e^{-z}}{1 + e^{-z}}$$

  - $z = w_0 + \sum_{j=1}^{k} w_k x_k$
- Logistic function: $f(z) = 1/(1 + e^{-z})$
  - Classical "S"-shape. Also called sigmoidal



- Value of f($\boldsymbol{x}$) does not perfectly predict class y.
  - Only a probability of y
  - Allows for linear classification to be imperfect.
  - Training will not require perfect separability

Logistic Model as a “Soft” Classifier



$z = 0.5x$ $\quad z = x$ $\quad z = 2x$ $\quad z = 100x$

- Plot of

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}, \qquad z = w_1 x$$

  - Markers are random samples
- Higher w_1: prob transition becomes sharper
  - Fewer samples occur across boundary
- As $w_1 \to \infty$ logistic becomes “hard” rule

$$P(y = 1|x) \approx \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

Multi-Class Logistic Regression

- Suppose $y \in 1, \ldots, K$
  - K possible classes (e.g. digits, letters, spoken words, …)
- Multi-class regression:
  - $\boldsymbol{W} \in R^{K \times d}, \boldsymbol{w}_0 \in R^K$ Slope matrix and bias
  - $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}_0$: Creates K linear functions
- Then, class probabilities given by:

$$P(y = k|\boldsymbol{x}) = \frac{\mathrm{e}^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

Softmax Operation

- Consider soft-max function:

$$g_k(\boldsymbol{z}) = \frac{\mathrm{e}^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

  - K inputs $\boldsymbol{z} = (\mathrm{z}_1, \ldots, z_K)$, K outputs $f(\boldsymbol{z}) = (f(\mathbf{z})_1, \ldots, f(\mathbf{z})_K)$

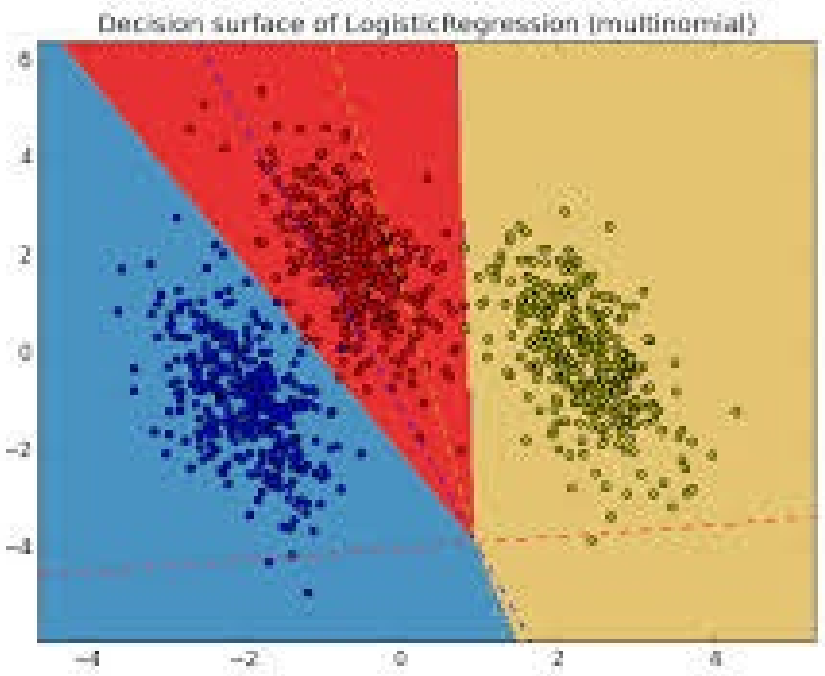
- Properties: $f(\boldsymbol{z})$ is like a PMF on the labels $[0,1, \ldots, K-1]$
  - $g_k(\boldsymbol{z}) \in [0,1]$ for each component k
  - $\sum_{k=1}^{K} g_k(\boldsymbol{z}) = 1$

- Softmax property: When $z_k \gg z_\ell$ for all $\ell \neq k$:
  - $g_k(\boldsymbol{z}) \approx 1$
  - $g_\ell(\boldsymbol{z}) \approx 0$ for all $\ell \neq k$

- Multi-class logistic regression: Assigns highest probability to class k when z_k is largest

$$z_k = \boldsymbol{w}_k^T \boldsymbol{x} + w_{0k}$$

Multi-Class Logistic Regression Decision Regions



- Each decision region defined by set of hyperplanes
- Intersection of linear constraints
- Sometimes called a polytope

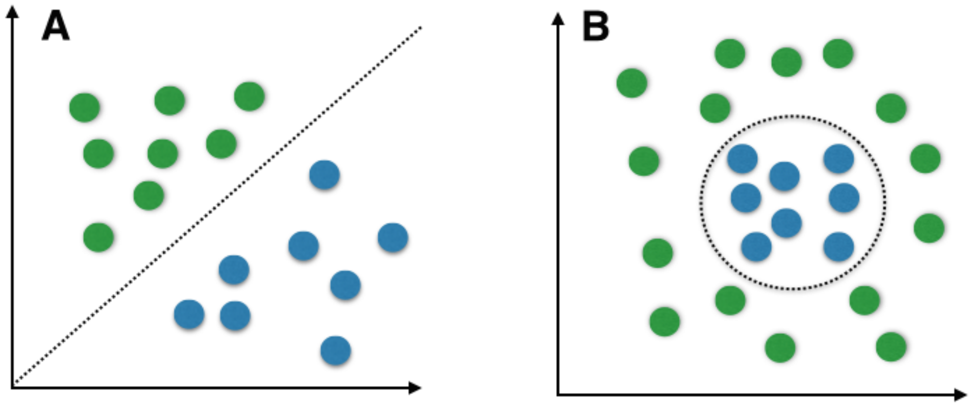
Transform Linear Models

- As in regression, logistic models can be applied to transform features
- Step 1: Map $\boldsymbol{x}$ to some transform features, $\phi(\boldsymbol{x}) = \left[\phi_1(\boldsymbol{x}), \ldots, \phi_p(\boldsymbol{x})\right]^T$ ⟵ Additional transform step
- Step 2: Linear weights: $z_k = \sum_{j=1}^{p} W_{kj}\phi_j(\boldsymbol{x})$
- Step 3: Soft-max $P(y = k|\boldsymbol{z}) = g_k(\boldsymbol{z}),\ \ g_k(\boldsymbol{z}) = \frac{e^{z_k}}{\sum_\ell e^{z_\ell}}$

- Example transforms:
  - Standard regression $\phi(\boldsymbol{x}) = \left[1, x_1, \ldots, x_j\right]^T$ (j original features, j+1 transformed features)
  - Polynomial regression: $\phi(\boldsymbol{x}) = [1, x, \ldots, x^d]^T$ (1 original feature, $d + 1$ transformed features)

Using Transformed Features

- Enables richer class boundaries
- Example: Fig B is not linearly separable
- But, consider nonlinear features
  - $\phi(\boldsymbol{x}) = [1, x_1, x_2, x_1^2, x_2^2]^T$
- Then can discriminate classes with linear function
  - $z = [-r^2, 0,0,1,1]\phi(x) = x_1^2 + x_2^2 - r^2$
- Blue when $z \leq 0$ and Green when $z > 0$



Outline

- ❑Motivating Example: Classifying a breast cancer test
- ❑Linear classifiers
- ❑Logistic regression
- ❑Fitting logistic regression models
- ❑Measuring accuracy in classification

Learning the Logistic Model Parameters

- Consider general three part logistic model:
  - Transform to features: $\boldsymbol{x} \mapsto \phi(\boldsymbol{x})$
  - Linear weights: $\boldsymbol{z} = \boldsymbol{W}\phi(\boldsymbol{x}), \quad \boldsymbol{W} \in R^{K\times p}$
  - Softmax: $P(y = k|\boldsymbol{x}) = g_k(\boldsymbol{z}) = g_k\big(\boldsymbol{W}\phi(\boldsymbol{x})\big)$

- Weight matrix $\boldsymbol{W}$ represents unknown model parameters

- Learning problem:
  - Given training data, $(\boldsymbol{x}_i, y_i), i = 1, \ldots, N$
  - Learn weight matrix $\boldsymbol{W}$
  - What loss function to minimize?

Likelihood Function

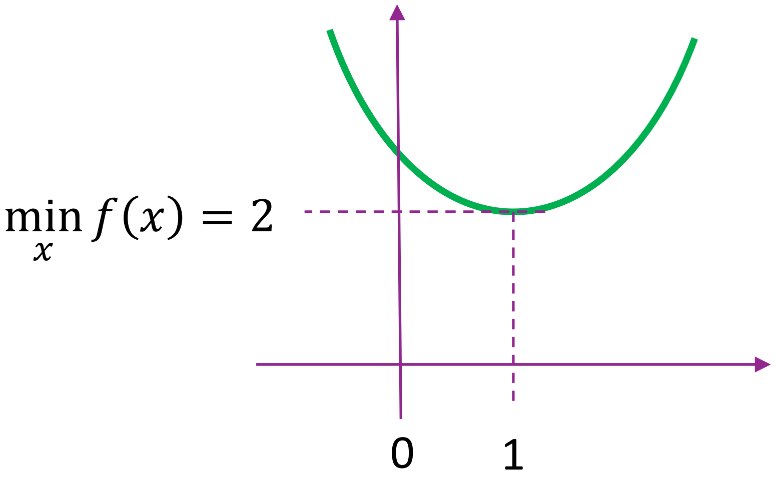
- Represent training data in vector form:
  - Data matrix: $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)^T$
  - Class label vector: $\boldsymbol{y} = (y_1, \dots, y_N)^T$
  - One component for each training sample

- Likelihood function:
  - $P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})$ = Likelihood (i.e. probability) of class labels given inputs $\boldsymbol{X}$ and weights
  - Function of training data $(\boldsymbol{X}, \boldsymbol{y})$ and parameters $\boldsymbol{W}$

Min and Argmin

- Given a function $f(x)$
- $\min_x f(x)$
  - Minimum value of the $f(x)$
  - Point on the y-axis
- $\arg\min_x f(x)$
  - Value of x where $f(x)$ is a minimum
  - Point on the x-axis
- Similarly, define $\max_x f(x)$ and $\arg\max_x f(x)$

$f(x) = (x-1)^2 + 2$



$\arg\min_x f(x) = 1$

Maximum Likelihood Estimation

- Given training data $(\boldsymbol{X}, \boldsymbol{y})$
- Likelihood function: $P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})$
- Maximum likelihood estimation

$$\widehat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W})$$

  - Finds parameters for which observations are most likely
  - Very general method in estimation

Log Likelihood

- Assume outputs y_i are independent, depending only on $\boldsymbol{x}_i$
- Then, likelihood factors:

$$P(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{W}) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i,\boldsymbol{W})$$

- Define negative log likelihood:

$$L(\boldsymbol{W}) = -\ln P(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{W}) = -\sum_{i=1}^{N} \ln P(y_i|\boldsymbol{x}_i,\boldsymbol{W})$$

- Maximum likelihood estimator can be re-written as:

$$\widehat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{W}) = \arg\min_{\boldsymbol{W}} L(\boldsymbol{W})$$

Logistic Loss Function for binary classification

- Negative log likelihood function: $J(w) = -\sum_{i=1}^{n} \ln P(y_i | \boldsymbol{x}_i, \boldsymbol{w})$

$$P(y_i = 1 | \boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{1 + e^{-z_i}}, \qquad z_i = \boldsymbol{w}_{1:p}^T \boldsymbol{x}_i + w_0$$

- Therefore,

$$P(y_i = 1 | \boldsymbol{x}_i, \boldsymbol{w}) = \frac{e^{z_i}}{1 + e^{z_i}}, \qquad P(y_i = 0 | \boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{1 + e^{z_i}}$$

- Hence,

$$\ln P(y_i | \boldsymbol{x}_i, \boldsymbol{w}) = y_i \ln P(y_i = 1 | \boldsymbol{x}_i, \boldsymbol{w}) + (1 - y_i) \ln P(y_i = 0 | \boldsymbol{x}_i, \boldsymbol{w}) = y_i z_i - \ln[1 + e^{z_i}]$$

- Loss function = binary cross entropy:

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \ln[1 + e^{z_i}] - y_i z_i$$

One-Hot Log Likelihood for Multi-Class Classification

- To find MLE, we re-write the negative log likelihood
- Define the "one-hot" vector:

$$r_{ik} = \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases}, \qquad i = 1, \dots, N, \qquad k = 1, \dots, K$$

- Then, $\ln P(y_i | \boldsymbol{x}_i, \boldsymbol{W}) = \sum_{k=1}^{K} r_{ik} \ln P(y_i = k | \boldsymbol{x}_i, \boldsymbol{W})$
- Hence, negative log likelihood is (proof on board):

$$L(\boldsymbol{W}) = \sum_{i=1}^{N} \left[\ln \left[\sum_k e^{z_{ik}} \right] - \sum_k z_{ik} r_{ik} \right]$$

  - Sometimes called the cross-entropy

Gradient Calculations

- To minimize take partial derivatives: $\frac{\partial L(W)}{\partial W_{kj}} = 0$ for all W_{kj}
- Define transform matrix: $A_{ij} = \phi_j(\boldsymbol{x}_i)$
- Hence, $z_{ik} = \sum_{j=1}^{p} A_{ij} W_{kj}$
- Estimated class probabilities: $p_{ik} = \frac{e^{z_{ik}}}{\sum_{\ell} e^{z_{i\ell}}}$
- Gradient components are (proof on board): $\frac{\partial L(\boldsymbol{W})}{\partial W_{kj}} = \sum_{i=1}^{N} (p_{ik} - r_{ik}) A_{ij} = 0$
  - $K \times p$ equations and $K \times p$ unknowns
- Unfortunately, no closed-form solution to these equations
  - Nonlinear dependence of p_{ik} on terms in W

Numerical Optimization

- We saw that we can find minima by setting $\nabla f(x) = 0$
  - M equations and M unknowns.
  - May not have closed-form solution

- Numerical methods: Finds a sequence of estimates x^t

$$x^t \to x^*$$

  - Under some conditions, it converges to some other "good" minima
  - Run on a computer program, like python

- Next lecture: Will discuss numerical methods to perform optimization
- This lecture: Use built-in python routine

Logistic Regression in Python

```
xnames = ['thick','size_unif','shape_unif','marg','cell_size','bare',
          'chrom','normal','mit']
X = np.array(df[xnames])
y = np.array(df['class'])
Xs = preprocessing.scale(X)
Iben = np.where(y==BEN_VAL)[0]
Imal = np.where(y==MAL_VAL)[0]
y[Iben]=0
y[Imal]=1
```

```
logreg = linear_model.LogisticRegression(C=1e5)
```

```
logreg.fit(Xs, y)
```

- Sklearn uses very efficient numerical optimization.
- Mostly internal to user
  - Don’t need to compute gradients

```
data = {'feature': xnames, 'slope': np.squeeze(logreg.coef_)}
dfslope = pd.DataFrame(data=data)
dfslope
```

	feature	slope
0	thick	1.508834
1	size_unif	-0.015979
2	shape_unif	0.957072
3	marg	0.947234
4	cell_size	0.214964
5	bare	1.395001
6	chrom	1.095654
7	normal	0.650696
8	mit	0.925912

w

http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Outline

- Motivating Example: Classifying a breast cancer test
- Linear classifiers
- Logistic regression
- Fitting logistic regression models
- Measuring accuracy in classification

Errors in Binary Classification

- Two types of errors:
  - Type I error (False positive / false alarm): Decide $\hat{y} = 1$ when $y = 0$
  - Type II error (False negative / missed detection): Decide $\hat{y} = 0$ when $y = 1$
- Implication of these errors may be different
  - Think of breast cancer diagnosis
- Accuracy of classifier can be measured by:
  - $TPR = P(\hat{y} = 1 | y = 1)$
  - $FPR = P(\hat{y} = 1 | y = 0)$
  - Accuracy=$P(\hat{y} = 1 | y = 1)$+ $P(\hat{y} = 0 | y = 0)$
    - (percentage of correct classification)

predicted → real ↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

Many Other Metrics

predicted → / real ↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN}+\text{FP}}$$

- From previous slide
  - $TPR = P(\hat{y} = 1|y = 1)$=sensitivity
  - $FPR = P(\hat{y} = 1|y = 0)$=1-specificity
- Machine learning often uses (positive=items of interests in retrieval applications)
  - Recall = Sensitivity =TP/(TP+FN) (How many positives are detected among all positive?)
  - Precision =TP/(TP+FP) (How many detected positive is actually positive?)
  - F1-score = $\frac{Precision * Recall}{(Precision+Recall)/2} = \frac{2TP}{2TP+FN+FP} = \frac{TP}{TP+\frac{FN+FP}{2}}$
  - Accuracy=(TP+TF)/(TP+FP+TN+FN) (percentage of correct classification)
- Medical tests:
  - Sensitivity = $P(\hat{y} = 1|y = 1) = TPR$
  - Specificity = $P(\hat{y} = 0|y = 0) = 1 - FPR$ =True negative rate
  - Need a good tradeoff between sensitivity and specificity

Breast Cancer

- Measure accuracy on test data
- Use 4-fold cross-validation
- Sklearn has built-in functions for CV

```
Precision =  0.9614
Recall =     0.9554
f1 =         0.9578
Accuracy =   0.9664
```

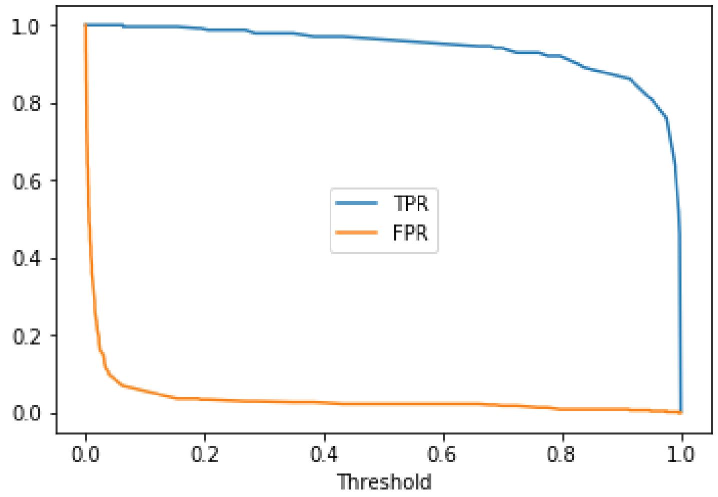
```python
from sklearn.model_selection import KFold
from sklearn.metrics import precision_recall_fscore_support
nfold = 4
kf = KFold(n_splits=nfold)
prec = []
rec = []
f1 = []
acc = []
for train, test in kf.split(Xs):
    # Get training and test data
    Xtr = Xs[train,:]
    ytr = y[train]
    Xts = Xs[test,:]
    yts = y[test]

    # Fit a model
    logreg.fit(Xtr, ytr)
    yhat = logreg.predict(Xts)

    # Measure
    preci,reci,f1i,_= precision_recall_fscore_support(yts,yhat,average='binary')
    prec.append(preci)
    rec.append(reci)
    f1.append(f1i)
    acci = np.mean(yhat == yts)
    acc.append(acci)

# Take average values of the metrics
precm = np.mean(prec)
recm = np.mean(rec)
f1m = np.mean(f1)
accm= np.mean(acc)

print('Precision = {0:.4f}'.format(precm))
print('Recall =    {0:.4f}'.format(recm))
print('f1 =        {0:.4f}'.format(f1m))
print('Accuracy =  {0:.4f}'.format(accm))
```

Go through the demo

- Up to binary classification with cross validation

Hard Decisions

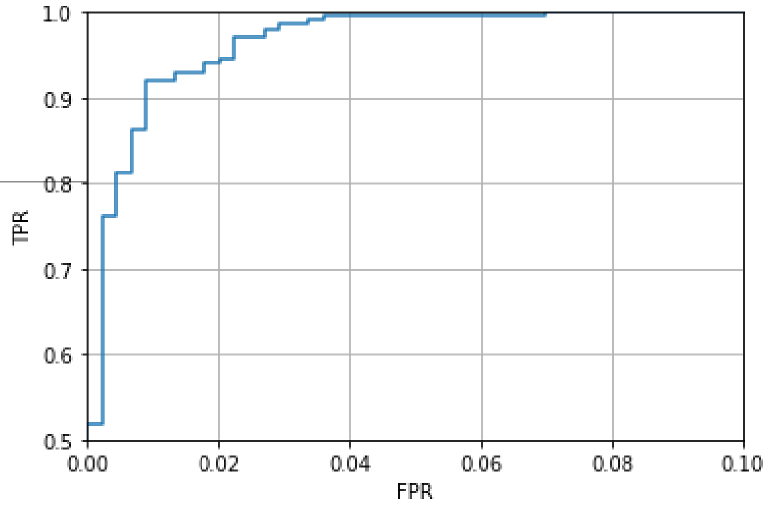
- Logistic classifier outputs a soft label: $P(y=1|x) \in [0,1]$
  - $P(y=1|x) \approx 1 \Rightarrow y=1$ more likely
  - $P(y=0|x) \approx 1 \Rightarrow y=0$ more likely
- Can obtain a hard label by thresholding:
  - Set $\hat{y}=1 \; if \;\; P(y=1|x) > t$
  - t = Threshold
- How to set threshold?
  - Set $t=\frac{1}{2} \Rightarrow$ Minimizes overall error rate
  - Increasing $t \Rightarrow$ Decreases false positives, but also reduces sensitivity
  - Decreasing $t \Rightarrow$ Increases sensitivity, but also increases false positive

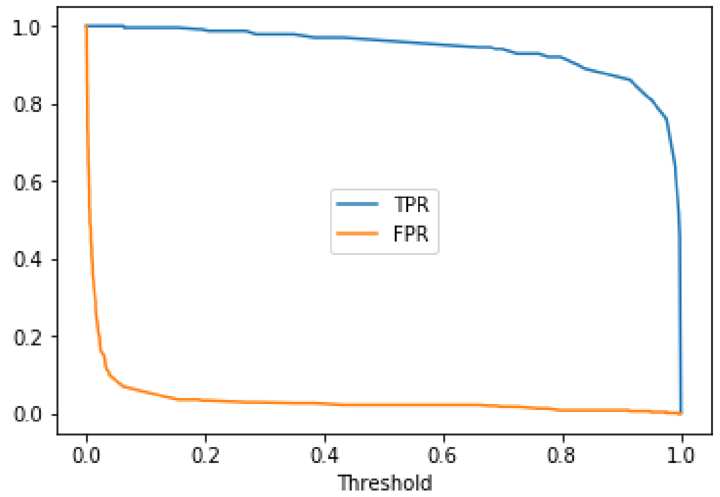


ROC Curve

- Varying threshold obtains a set of classifier
- Trades off FPR (1-specificity) and TPR (sensitivity)
- Can visualize with ROC curve
  - Receiver operating curve
  - Term from digital communications

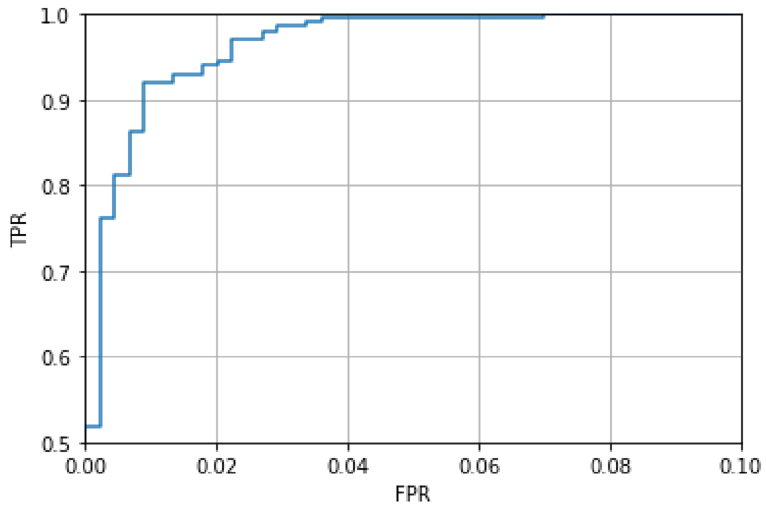
```
from sklearn import metrics
yprob = logreg.predict_proba(Xs)
fpr, tpr, thresholds = metrics.roc_curve(y,yprob[:,1])

plt.plot(fpr,tpr)
plt.grid()
plt.xlabel('FPR')
plt.ylabel('TPR')
```





Area Under the Curve (AUC)

- ❑As one may choose a particular threshold based on the desired trade-off between the TPR and FPR, it may not be appropriate to evaluate the performance of a classifier for a fixed threshold.
- ❑AUC is a measure of goodness for a classifier that is independent of the threshold.
- ❑A method with a higher AUC means that under the same FPR, it has higher PPR.
- ❑What is the highest AUC?
- ❑Should report average AUC over cross validation folds




```
uac=metrics.roc_auc_score(y,yprob[:,1])
print("UAC=%f" % uac)
```

```
UAC=0.996315
```

Go through the demo

- Up to binary classification evaluation using ROC

Multi-Class Classification in Python

- Two options
- One vs Rest (OVR)
  - Solve a binary classification problem for each class k
  - For each class k, train on modified binary labels (indicates if sample is in class or not)

$$\tilde{y}_i = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k \end{cases}$$

  - Predict based on classifier that yields highest score
- Multinomial
  - Directly solve weights for all classes using the multi-class cross entropy

Metrics for Multiclass Classification

- Using a $K \times K$ confusion matrix
- Should normalize the matrix:
  - Sum over each row =1
- Can compute accuracy:
  - Per class: This is the diagonal entry
  - Average: The average of the diagonal entries

Pred--> Real↓	1	2	…	K
1				
2				
…				
K				

LASSO Regularization for Logistic Regression

- Similar to linear regression, we can use LASSO regularization with logistic regression
  - Forces the weighting coefficients to be sparse.
- Add L1 penalty $L(\boldsymbol{W}) = \sum_{i=1}^{N} [\ln[\sum_k e^{z_{ik}}] - z_{ik} r_{ik}] + \lambda \|W\|_1$
- The regularization level λ should be chosen via cross validation as before
- Sklearn implementation:

```
logreg = linear_model.LogisticRegression(penalty='l1')
logreg.C = c
```

  - Default use l2 penalty, to reduce the magnitude of weights
- C is the inverse of regularization strength $(C = 1/\lambda)$; must be a positive float.
  - Should use a large C is you do not want to apply regularization
- Go through the LASSO part of the demo

Go through the demo

- Go though the last part with LASSO regression

What You Should Know

- Formulate a machine learning problem as a classification problem
  - Identify features, class variable, training data
- Visualize classification data using a scatter plot.
- Describe a linear classifier as an equation and on a plot.
  - Determine visually if data is perfect linearly separable.
- Formulate a classification problem using logistic regression
  - Binary and multi-class
  - Describe the logistic and soft-max function
  - Understand the idea of using the logistic function to approximate the probability
- Derive the loss function for ML estimation of the weights in logistic regression
- Use sklearn packages to fit logistic regression models
- Measure the accuracy of classification: precision, recall, accuracy
- Adjust threshold of classifiers for trading off types of classification errors. Draw a ROC curve and determine AUC
- Perform LASSO regularization for feature selection