Week 5:
Support Vector Machine

Confusion Matrix for Binary Classifier

True class	Predicted class	
	Predicted Negative	Predicted Positive
True Negative (#N)	#TN	#FP
True Positive (#P)	#FN	#TP

Reduce the 4 numbers to two performance metrics

true positive rate (recall):

TP = #TP/ #P = #TP / (#TP + #FN)

false positive rate:

FP = #FP/ #N = #FP / (#FP + #TN)

Using three different threshold values

True	Predicted	
	pos	neg
pos	40	60
neg	30	70

Classifier 1
TP = 0.4
FP = 0.3

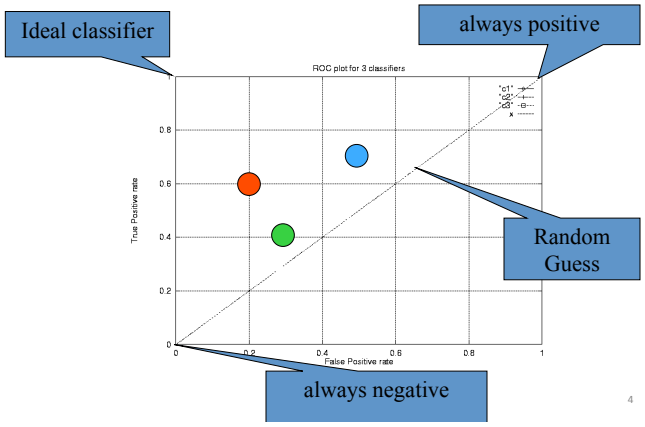
True	Predicted	
	pos	neg
pos	70	30
neg	50	50

Classifier 2
TP = 0.7
FP = 0.5

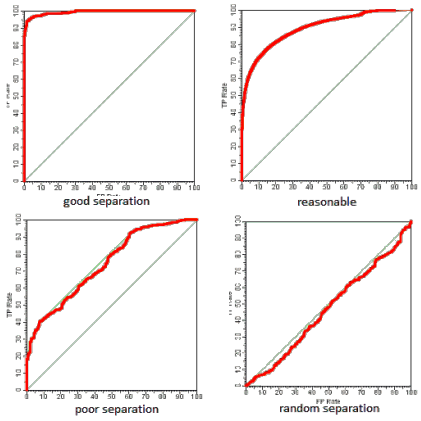
True	Predicted	
	pos	neg
pos	60	40
neg	20	80

Classifier 3
TP = 0.6
FP = 0.2

ROC (Receiver operating characteristic) plot for the 3 Classifiers

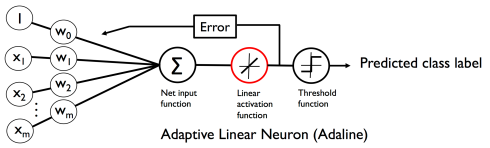


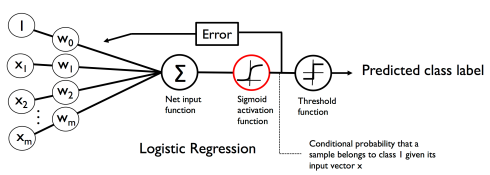
ROC & AUC



Logistic Regression Review

Adaline model uses the identify function **f(x) = x** as the activation function. But there are limitations…





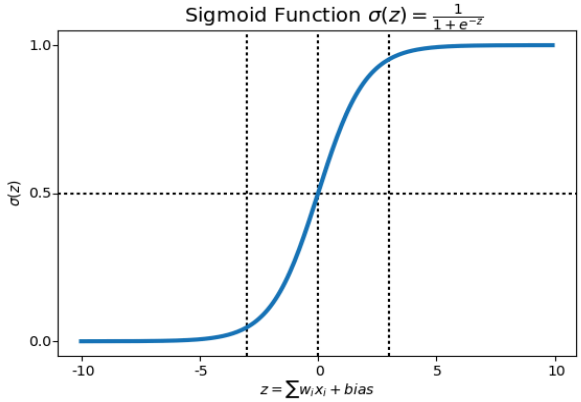


Image source: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Cost Function for Logistic Regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$P(y=1 \mid x;\theta) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Taken from Prof. Andrew Ng.'s Coursera ML course

The goal is to maximize the likelihood of predicting the expected output.

Cost Function for Logistic Regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

0 If actual y=1

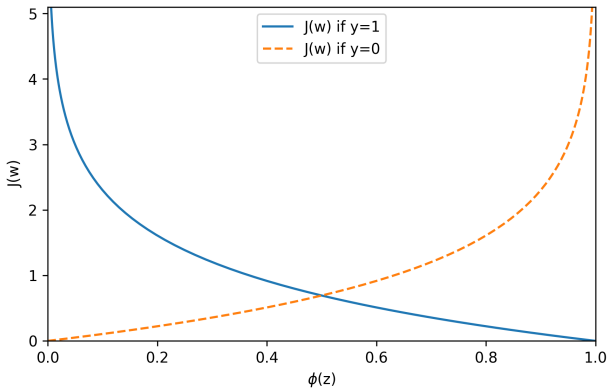
0 If actual y=0

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

Cost Function for Logistic Regression



Use Regularization to Reduce Overfitting

Add a term to the cost function to penalize large weight values of model.

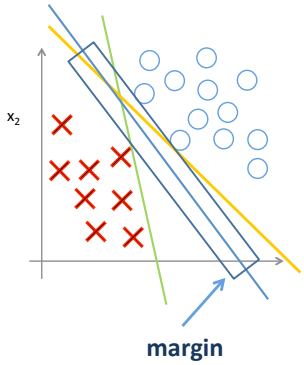
$$J(\boldsymbol{w}) = -\sum y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)$$

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left[-y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)\right] + \frac{\lambda}{2}\|w\|^2$$

The hyper-parameter C in logistic regression model is the inverse-regularization parameter:
Smaller C values specify stronger regularization and thus smaller the weight coefficients, and reduces the variance of model.

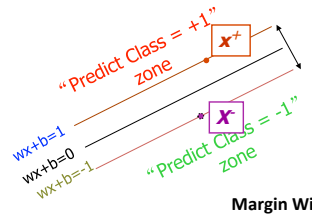
Support Vector Machine

LARGE MARGIN CLASSIFIERS



- Find a classifier (hyperplane)
- An infinite number of such hyperplanes exist.
- Find the hyperplane that maximizes the gap between **data points on the boundaries** (so-called **"support vectors"**).

Linear SVM Mathematically



What we know:

- $\boldsymbol{w}^T \cdot \boldsymbol{x}^{pos} + w_0 = +1$
- $\boldsymbol{w}^T \cdot \boldsymbol{x}^{neg} + w_0 = -1$

then $\boldsymbol{w}^T \cdot (\boldsymbol{x}^{pos} - \boldsymbol{x}^{neg}) = 2$

And we normalize by the length of vector ***w****.*

Margin Width: The distance between the positive and negative hyperplane.

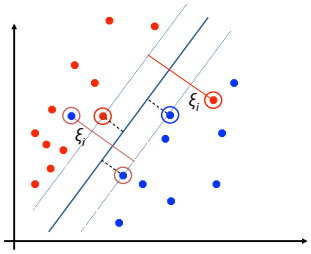
$$M = \frac{w^T(x^+ - x^-)}{\|w\|} = \frac{2}{\|w\|}$$

Minimize the reciprocal term: $\frac{1}{2}\|w\|^2$

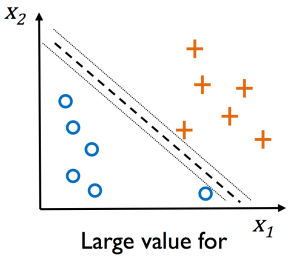
Soft Margin Classification

Allow slack variables to handle non-linearly separable data under appropriate cost penalization. (Valadimir Vapnik 1995)

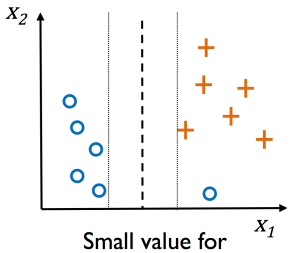
Cost function: $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_i \xi^{(i)}\right)$



C parameter



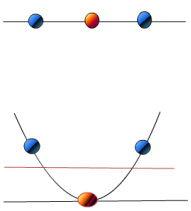
Large value for parameter C



Small value for parameter C

Small values for C lower the variance of the model and thus reducing overfitting.

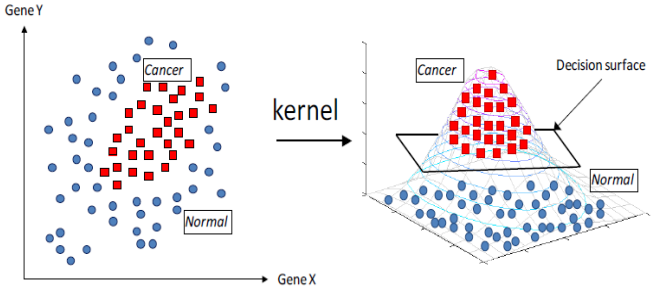
Kernel Trick

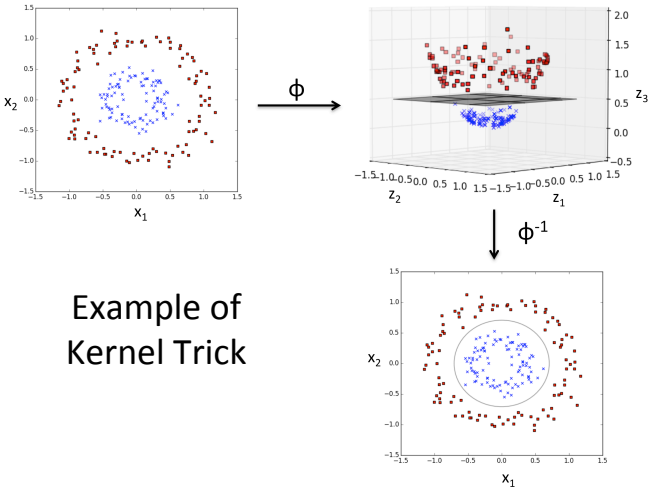


- Linear Inseparable data
- project all points up to a two dimensional space using the mapping $x \rightarrow (x, x^2)$
- We can now find a hyper plane to separate data with SVM.

The mapping $x \rightarrow (x, x^2)$ is called **KERNEL FUNCTION**

Example of Kernel Trick





Example of Kernel Trick

Kernels

Examples:

$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$ Linear kernel

$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2)$ Gaussian kernel

$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|)$ Exponential kernel

$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q$ Polynomial kernel

$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2)$ Hybrid kernel

$K(\vec{x}_i, \vec{x}_j) = \tanh(k\vec{x}_i \cdot \vec{x}_j - \delta)$ Sigmoidal

SVM disadvantages

- If the points on the boundaries are not informative (e.g., due to noise), SVMs will not do well.
- Can be computationally expensive

"However, from a practical point of view perhaps the most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements.."
Horváth (2003) in Suykens et al. p 392

"Besides the advantages of SVMs - from a practical point of view - they have some drawbacks. An important practical question that is not entirely solved, is the selection of the kernel function parameters…"
Horváth (2003) in Suykens et al.