Intro to Machine Learning

Dr. Mahmoud Nabil

Dr. Mahmoud Nabil
mnmahmoud@ncat.edu

North Carolina A & T State University

January 25, 2023

Talk Overview

Outline

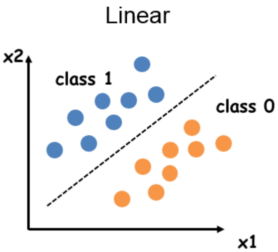
Logistic Regression Introduction

Logistic Regression

It is a statistical model used for binary classification. The inputs are the features values and the output (y) is a probability from 0 to 1.

Note that

- Logistic regression is a linear classifier.
- The equation of the decesion boundry : $0 = w_2 x_2 + w_1 x_1 + w_0$
- Class 0 condition: $0 < w_2 x_2 + w_1 x_1 + w_0$
- Class 1 condition: $0 > w_2 x_2 + w_1 x_1 + w_0$



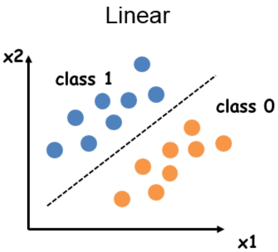
Logistic Regression Introduction

Logistic Regression

It is a statistical model used for binary classification. The inputs are the features values and the output (y) is a probability from 0 to 1.

Note that

- Logistic regression is a linear classifier.
- The equation of the decesion boundry : $0 = w_2x_2 + w_1x_1 + w_0$
- Class 0 condition: $0 < w_2x_2 + w_1x_1 + w_0$
- Class 1 condition: $0 > w_2x_2 + w_1x_1 + w_0$

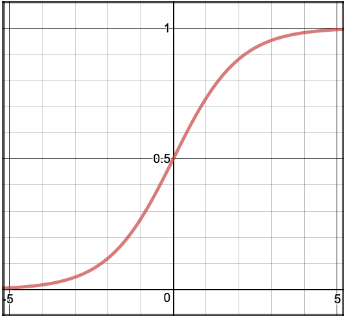


How get y' as probability given these conditions?

Sigmoid Function

- In order to map predicted values to probabilities, we use the sigmoid function.

$$sigmoid(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$



Logistic Regression

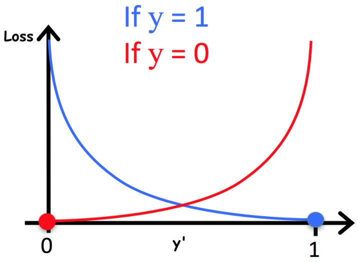
- We can set decesion boundary $z = w_2x_2 + w_1x_1 + w_0$
- Then $y' = \sigma(z) = \frac{1}{1+e^{-z}}$
- What if point (x_1, x_2) is below the decesion boundary?
- What if point (x_1, x_2) is above the decesion boundary?
- $\frac{d}{dz}\sigma(z) = \sigma(z)(1 - \sigma(z))$

Logistic Loss Function

- Since y' in logistic regression is a probability between 0 and 1.

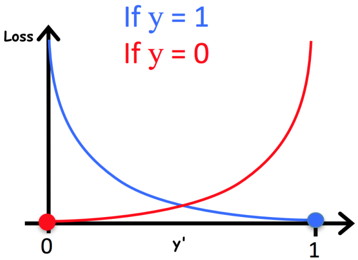
Logistic Loss Function

- Since y' in logistic regression is a probability between 0 and 1.
- Our loss can be defined with the following loss function.
  - if y = 1 : Loss = -log(y')
  - if y = 0 : Loss = -log(1-y')



Logistic Loss Function

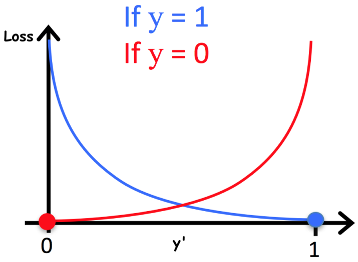
- Since y' in logistic regression is a probability between 0 and 1.
- Our loss can be defined with the following loss function.
  - if y = 1 : Loss = -log(y')
  - if y = 0 : Loss = -log(1-y')



$$\text{Loss} = \ell = \text{-ylog(y')-(1-y)log(1-y')}$$

Logistic Loss Function

- Since y' in logistic regression is a probability between 0 and 1.
- Our loss can be defined with the following loss function.
  - if y = 1 : Loss = -log(y')
  - if y = 0 : Loss = -log(1-y')



Loss= ℓ = -ylog(y')-(1-y)log(1-y')

Now we can train with gradient descent to find the weights

Generalization and Gradient

- For n features: $z = \sum_{i=0}^{i=n} w_i x_i$, (w_0 is the bias)
- vector representation z = $\mathbf{w}^T\mathbf{x}$
- y' = $sigmoid(z) = \sigma(z)$
- $\ell = -ylog(y') - (1-y)log(1-y')$

Gradient Derivation

$$\frac{d\ell}{dw_i} = \frac{d\ell}{dy'}\frac{dy'}{dw_i} = \frac{d\ell}{dy'}\frac{dy'}{dz}\frac{dz}{dw_i}$$

$$=$$

Gradient Derivation

$$\frac{d\ell}{dw_i} = \frac{d\ell}{dy'}\frac{dy'}{dw_i} = \frac{d\ell}{dy'}\frac{dy'}{dz}\frac{dz}{dw_i}$$

$$= \underbrace{[\frac{-y}{\sigma(z)} + \frac{1-y}{1-\sigma(z)}]}_{\frac{d\ell}{dy'}} * \underbrace{\sigma(z)(1-\sigma(z))}_{\frac{dy'}{dz}} * \underbrace{x_i}_{\frac{dz}{dw_i}}$$

$$= \underbrace{[\frac{-y(1-\sigma(z)) + (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}]}_{\frac{d\ell}{dy'}} * \underbrace{\sigma(z)(1-\sigma(z))}_{\frac{dy'}{dz}} * \underbrace{x_i}_{\frac{dz}{dw_i}}$$

$$= \underbrace{[\frac{-y(1-\sigma(z)) + (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}]}_{\frac{d\ell}{dy'}} * \underbrace{\sigma(z)(1-\sigma(z))}_{\frac{dy'}{dz}} * \underbrace{x_i}_{\frac{dz}{dw_i}}$$

$$= (\sigma(z) - y) * x_i$$

Summary Linear vs Logistic Regression

Linear Regression

- $y' = \mathbf{w}^T\mathbf{x}$
- $\ell = (y - y')^2$
- $\frac{d\ell}{dw_i} = [2(y' - y) * x_i]$

Logistic Regression

- $z = \mathbf{w}^T\mathbf{x}$
- $y' = sigmoid(z) = \frac{1}{1+e^{-z}}$
- $\ell =$ -ylog(y')-(1-y)log(1-y')
- $\frac{d\ell}{dw_i} = (y' - y) * x_i$

Summary Linear vs Logistic Regression

Linear Regression

- $y' = \mathbf{w}^T\mathbf{x}$
- $\ell = (y - y')^2$
- $\frac{d\ell}{dw_i} = [2(y' - y) * x_i]$

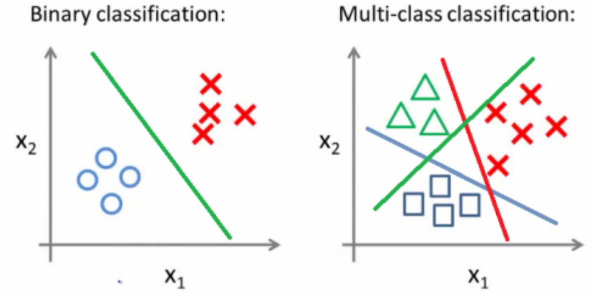
Logistic Regression

- $z = \mathbf{w}^T\mathbf{x}$
- $y' = sigmoid(z) = \frac{1}{1+e^{-z}}$
- $\ell =$ -ylog(y')-(1-y)log(1-y')
- $\frac{d\ell}{dw_i} = (y' - y) * x_i$

Loss is convex function interms of the weights (y' is function of the weights)

Outline

Multiclass Classification (One vs All)



For Three classes

$$\text{Result Class} = \underset{k \in \{1,2,3\}}{\text{argmax}}\ f_k(x)$$

From Linear to non-Linear (Feature Crosses)

- Our basic equation for one feature linear regression is

$$y' = w_1 x_1 + w_0$$

- Having the following equation allow us to fit quadratic input x_1

$$y' = w_2(x_1)^2 + w_1 x_1 + w_0$$

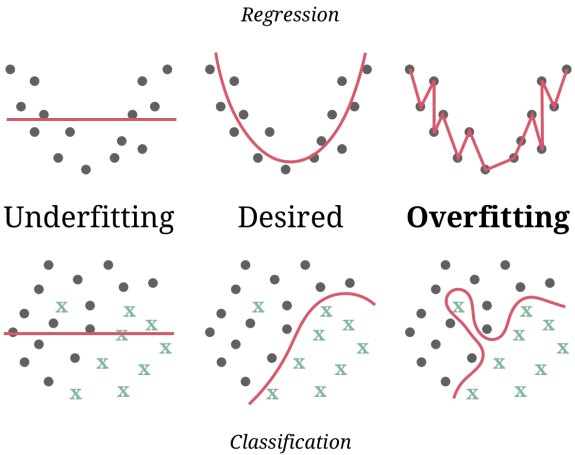
- The same is applicable for the Logistic Regression we can learn non linear decesion boundries
- By this trick we can increase the model complexity.

From Linear to non-Linear (Feature Crosses)

- Our basic equation for one feature linear regression is
$$y' = w_1x_1 + w_0$$
- Having the following equation allow us to fit quadratic input x_1
$$y' = w_2(x_1)^2 + w_1x_1 + w_0$$
- The same is applicable for the Logistic Regression we can learn non linear decesion boundries
- By this trick we can increase the model complexity.

But how this effect the learning?

Overfitting and Underfitting (1/2)



Note

A good model (best fit) should be able to generalize to new (unseen) data. How?

Overfitting and Underfitting (2/2)

- **Over-fitting:**
  - Model too complex (flexible)
  - Fits "noise" in the training data
  - High error is expected on the test data.
- **Under-fitting:**
  - Model too simplistic (too rigid)
  - Not powerful enough to capture salient patterns in training data and test data.

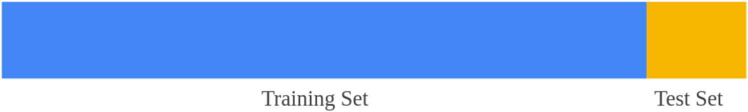
Note

A good model (best fit) should be able to generalize to new (unseen) data. How?

Training and Test Sets

To measure how our model generalize, we split our data to

- **Training set** a subset to train a model.
- **Test set** a subset to evaluate the trained model Estimeate Generalization.



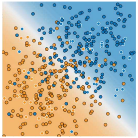
Training and Test Sets

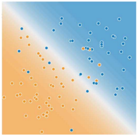
To measure how our model generalize, we split our data to

- **Training set** a subset to train a model.
- **Test set** a subset to evaluate the trained model Estimeate Generalization.

The test should:

- be large enough to yield statistically meaningful results.
- be representative of the data set as a whole.

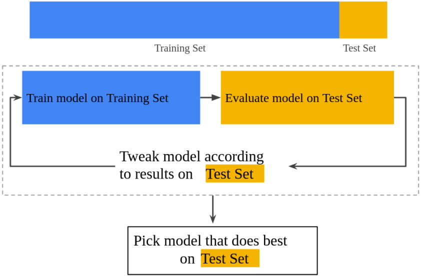

Training Data


Test Data

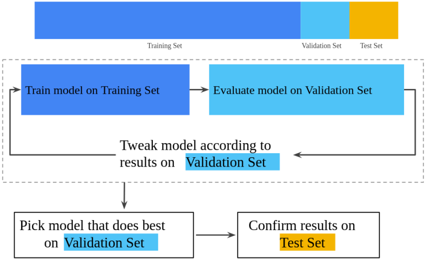
Validation Set

What if we have several model to compare and pick only one?

- Adding or removing features
- Trying different model complexities (linear, quadratic, etc)
- ...



More chances to Overfit.



Less chances to Overfit.

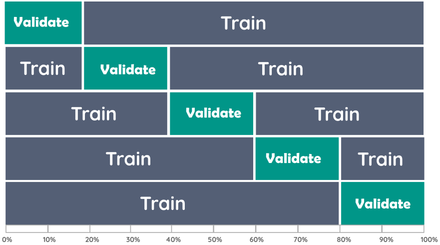
K-Cross Validation

Why?

- We can be exposed to the test set only once.
- We need to estimate future error as accurately as possible.

Ex.

- Randomly split the training into k sets.
- Validate on one in each turn (train on 4 others)
- Average the results over 5 folds



5-fold cross validation

Training vs. Generalization Error (1/3)

Training Error: It measures how we are performing on the training set (same as loss).

$$E_{train} = \frac{1}{|D_{train}|} \sum_{(\mathbf{x},y)\in D_{train}} error(f(\mathbf{x}), y)$$

Generalization Error:

- How well we will do on any kind future data from the same distribution.

$$E_{gen} = \int_{(\mathbf{x},y)\in D} error(f(\mathbf{x}), y) \underbrace{p(\mathbf{x}, y)}_{\text{How often we see (x,y) pair}} d\mathbf{x}$$

Training vs. Generalization Error (1/3)

Training Error: It measures how we are performing on the training set (same as loss).

$$E_{train} = \frac{1}{|D_{train}|} \sum_{(\mathbf{x},y)\in D_{train}} error(f(\mathbf{x}), y)$$

Generalization Error:

- How well we will do on any kind future data from the same distribution.

$$E_{gen} = \int_{(\mathbf{x},y)\in D} error(f(\mathbf{x}), y) \underbrace{p(\mathbf{x}, y)}_{\text{How often we see (x,y) pair}} d\mathbf{x}$$

Can never compute generalization error practically

Training vs. Generalization Error (2/3)

Test Error:

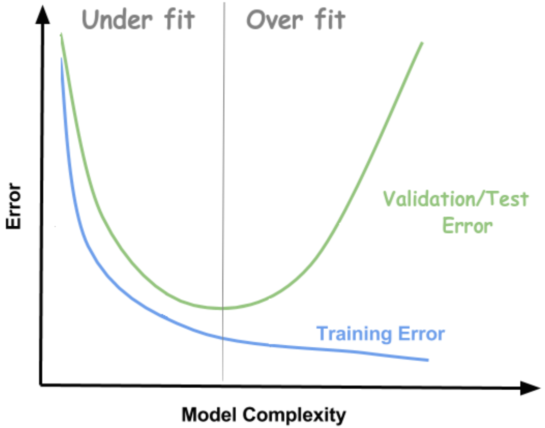
- Introduced to estimate the generalization error.
- That is why we should be exposed to test set only once.

$$E_{test} = \frac{1}{|D_{test}|} \sum_{(\mathbf{x},y)\in D_{test}} error(f(\mathbf{x}), y)$$

- How close E_{gen} to E_{test}? depends on $|D_{test}|$.

$$\lim_{|D_{test}|\to\infty} E_{test} \approx E_{gen}$$

Training vs. Generalization Error (3/3)

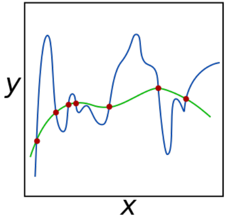


Regularization

- Sparse feature vectors often contain many dimensions. Feature cross results in even more dimensions (more model complexity).
- Can we force the weights for the meaningless features to drop to 0.

L2 and L1 penalize weights differently:

- L_2 penalizes $weight^2$. (L2 has not a discontinuity at 0)
  - $\ell_{new} = \ell_{old} + \text{p} \sum_{i=0}^{i=n} w_i^2$
  - $\frac{\ell_{new}}{dw_i} = \frac{\ell_{old}}{dw_i} + 2\text{p}\ w_i$
- L_1 penalizes $|weight|$. (L1 has a discontinuity at 0)
  - $\ell_{new} = \ell_{old} + \text{p} \sum_{i=0}^{i=n} |w_i|$
  - $\frac{\ell_{new}}{dw_i} = \frac{\ell_{old}}{dw_i} + \text{p}\ sign(w_i)$



p is the tuning parameter which decides how much we want to penalize w_i.

Outline

Scaling feature values

Scaling

Scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range (for example, 0 to 1 or -1 to +1).

Why?

1. Helps gradient descent converge more quickly.
2. Helps to decrease the possibility of weights over/under-flow.
3. Helps the model learn appropriate weights for each feature. Without feature scaling, the model will pay too much attention to the features having a wider range.

Min Max Scaler

Transform features by scaling each feature(f) to a given range [Min, Max].

$$u = \frac{f - f.min}{f.max - f.min}$$

$$f_{scaled} = u * (Max - Min) + Min$$

Z-score Scaler

Another popular scaling tactic is to calculate the Z score of each feature (f). The Z score relates the number of standard deviations away from the mean. In other words:

$$f_{scaled} = (f - f.mean)/f.std\_dev$$

Issues in the Datasets

In real-life, many examples in data sets are unreliable due to one or more of the following:

- **Omitted values.** For instance, a person forgot to enter a value for a house's age.
  - Categorical ("N/A") , "fill-in mean", remove ,other methods rule-learners, decision trees
- **Duplicate examples.** For example, a server mistakenly uploaded the same logs twice.
  - Remove duplicates
- **Bad labels.** For instance, a person mislabeled a picture of cat as a dog.
  - Clustering can help
- **Bad feature values/Outliers.** For example, someone typed in an extra digit, or a thermometer was left out in the sun.
  - Histogram can help.

How to Handle?

Always try to visualize the features. Histograms are also great mechanism

for visualizing your data in the aggregate. In addition, getting statistics like the following can help:

- Maximum and minimum
- Mean and median
- Standard deviation

Outline

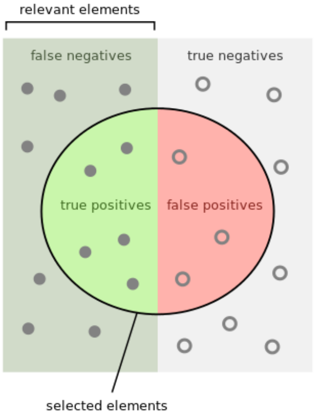
Thresholding

- In order to map a logistic regression value to a binary category, we must define a classification threshold.
- Classification threshold is problem-dependent.
- It does not have to be 0.5.
- Classification metrics are used to define the classification threshold.

Confusion Matrix

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

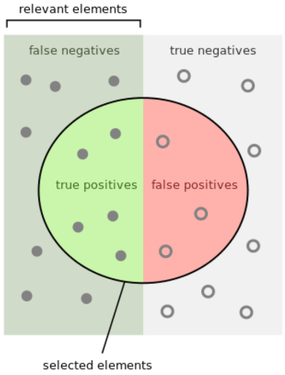


We want large diagonal, small FP, FN

Accuracy and Error

- **Accuracy** is the total correct prediction

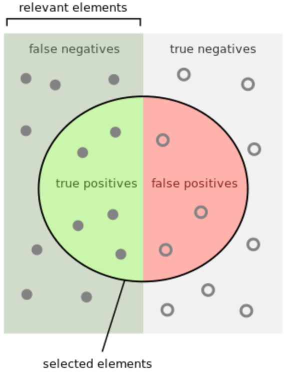
	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	



Accuracy and Error

- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$

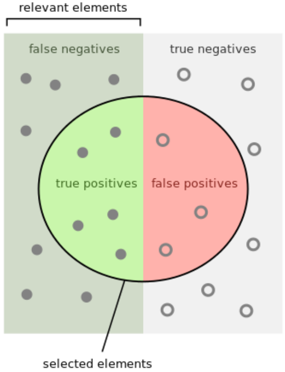
	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	



Accuracy and Error

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

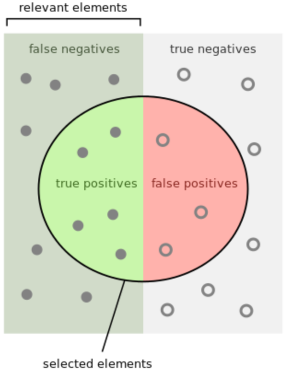
- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- **Error** is the total false prediction



Accuracy and Error

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

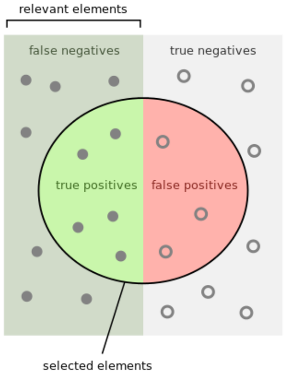
- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- **Error** is the total false prediction
  - $\frac{FP+FN}{TP+TN+FP+FN} = 1 \text{ - Accuracy}$



Accuracy and Error

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

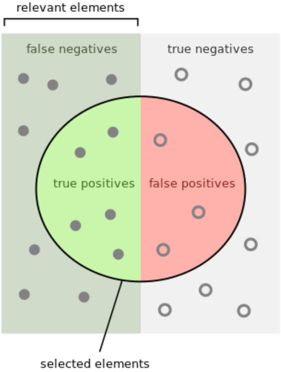
- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- **Error** is the total false prediction
  - $\frac{FP+FN}{TP+TN+FP+FN} = 1$ - Accuracy
- **Problem:** cannot handle unbalanced classes



Accuracy and Error

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

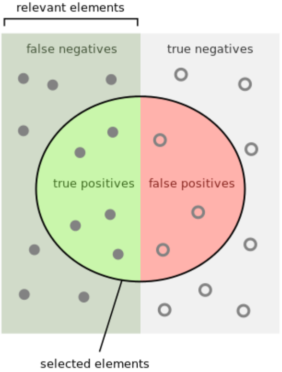
- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- **Error** is the total false prediction
  - $\frac{FP+FN}{TP+TN+FP+FN} = 1 \text{ - Accuracy}$
- **Problem:** cannot handle unbalanced classes
  - Predict whether an earthquake is about to happen



Accuracy and Error

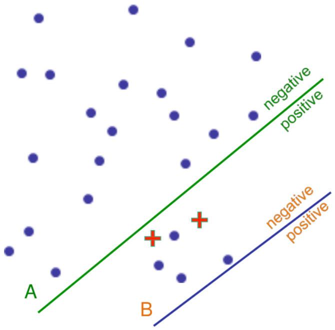
	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

- **Accuracy** is the total correct prediction
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- **Error** is the total false prediction
  - $\frac{FP+FN}{TP+TN+FP+FN} = 1$ - Accuracy
- **Problem:** cannot handle unbalanced classes
  - Predict whether an earthquake is about to happen
  - Happen very rarely, very good accuracy if always predict “No”.



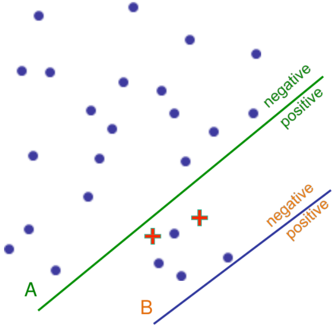
Problem with Accuracy

- You're predicting cancer possiblity (+) vs. not (•)



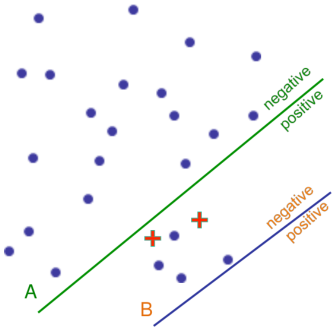
Problem with Accuracy

- You're predicting cancer possiblity (+) vs. not (•)
- Accuracy will prefer classifier B (fewer errors)



Problem with Accuracy

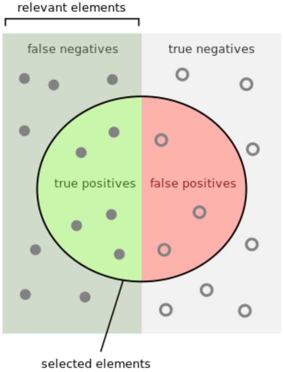
- You're predicting cancer possiblity (+) vs. not (•)
- Accuracy will prefer classifier B (fewer errors)
- Classifier A is better though.



Metrics (1/3)

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

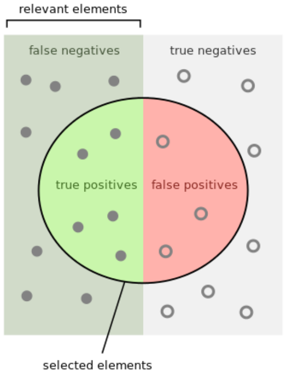
- **Recall** How many (+) we hit? (Recall = Sensitivity = True pos rate = hit rate)
  - $\frac{TP}{P} = \frac{TP}{TP+FN}$
- **Miss Rate** How many (+) we miss? (Miss rate = False neg rate = false rejection = type II error rate)
  - $1 - hitrate = \frac{FN}{P} = \frac{FN}{TP+FN}$



Metrics (2/3)

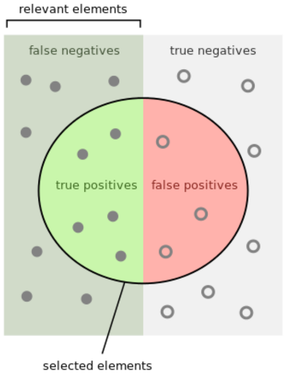
	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

- **Specificity** How many (-) we hit? (Specificity = True neg rate)
  - $\frac{TN}{N} = \frac{TN}{FP+TN}$
- **False Alarm** How many (-) we miss OR How many (+) we falsely accepted? (False alarm = False pos rate = false acceptance = = type I error rate) How many irrelevant items are selected?
  - $1 - Specificity = \frac{FP}{FP+TN}$



Metrics (2/3)

	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

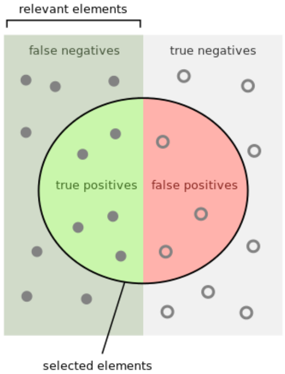


- **Specificity** How many (-) we hit? (Specificity = True neg rate)
  - $\frac{TN}{N} = \frac{TN}{FP+TN}$
- **False Alarm** How many (-) we miss OR How many (+) we falsely accepted? (False alarm = False pos rate = false acceptance = = type I error rate) How many irrelevant items are selected?
  - $1 - Specificity = \frac{FP}{FP+TN}$

Metrics (3/3)

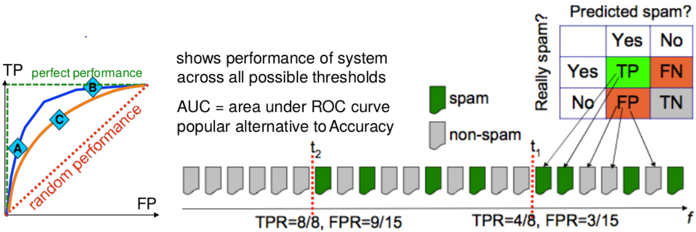
	Actually Positive (1)	Actually Negative (0)	
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	P'
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)	N'
	P	N	

- **Prcesion** How many of our (+) decesions is correct?
  - $\frac{TP}{P'} = \frac{TP}{TP+FP}$
- **F1 measure** Harmonic mean of precesion and the recall
  - $2\frac{PER*REC}{PER+REC}$



ROC curves

- Plot TPR vs. FPR as classification threshold (t) varies from 0 to 1
- RoC summarizes all the confusion matrices for all possible thresholds.
- Each point on the RoC is for a different classification threshold.
- (1,1) point is all (+) threshold.
- (0,0) point is all (-) threshold.
- Benefits:
    - Make us make a decesion for classification threshold.
    - Make us compare different classifier using AUC.



Questions