GENERAL ASSEMBLY

DATA SCIENCE

LECTURE 14: IMBALANCED CLASSES AND EVALUATION METRICS

YUCHEN ZHAO / DAT-14

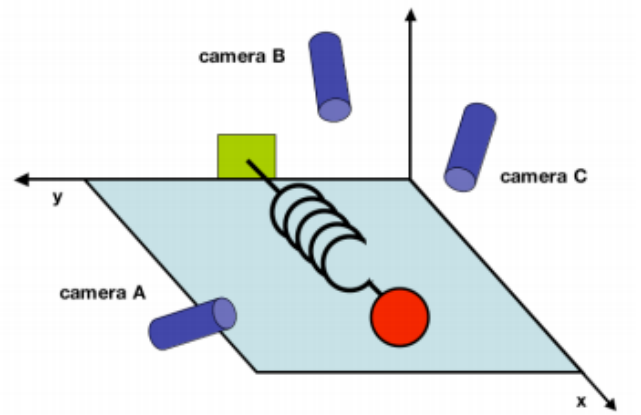
LAST TIME

I. DIMENSIONALITY REDUCTION
II. PRINCIPAL COMPONENTS ANALYSIS
III. SINGULAR VALUE DECOMPOSITION

EXERCISE:
IV. DIMENSIONALITY REDUCTION IN SCIKIT-LEARN

Another way of characterizing this is to say that high-dimensional spaces are inherently **sparse**.

EXAMPLE: 1D HARMONIC OSCILLATOR



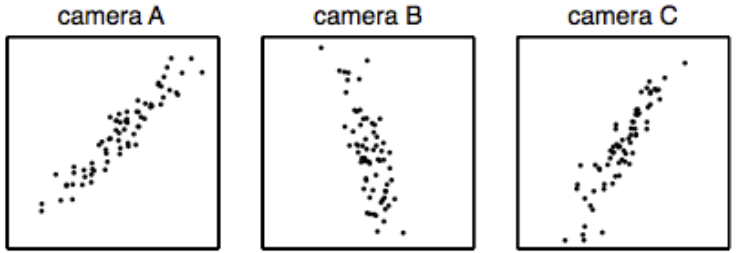


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

source: http://www.snl.salk.edu/~shlens/pca.pdf

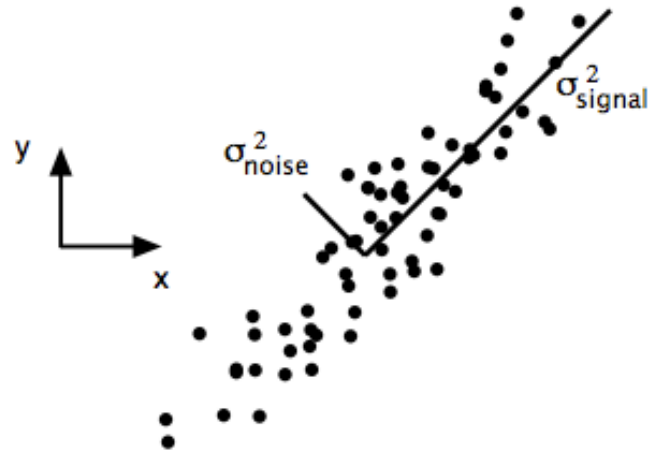


FIG. 2 Simulated data of (x, y) for camera A. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented by the two lines subtending the cloud of data. Note that the largest direction of variance does not lie along the basis of the recording (x_A, y_A) but rather along the best-fit line.

source: http://www.snl.salk.edu/~shlens/pca.pdf

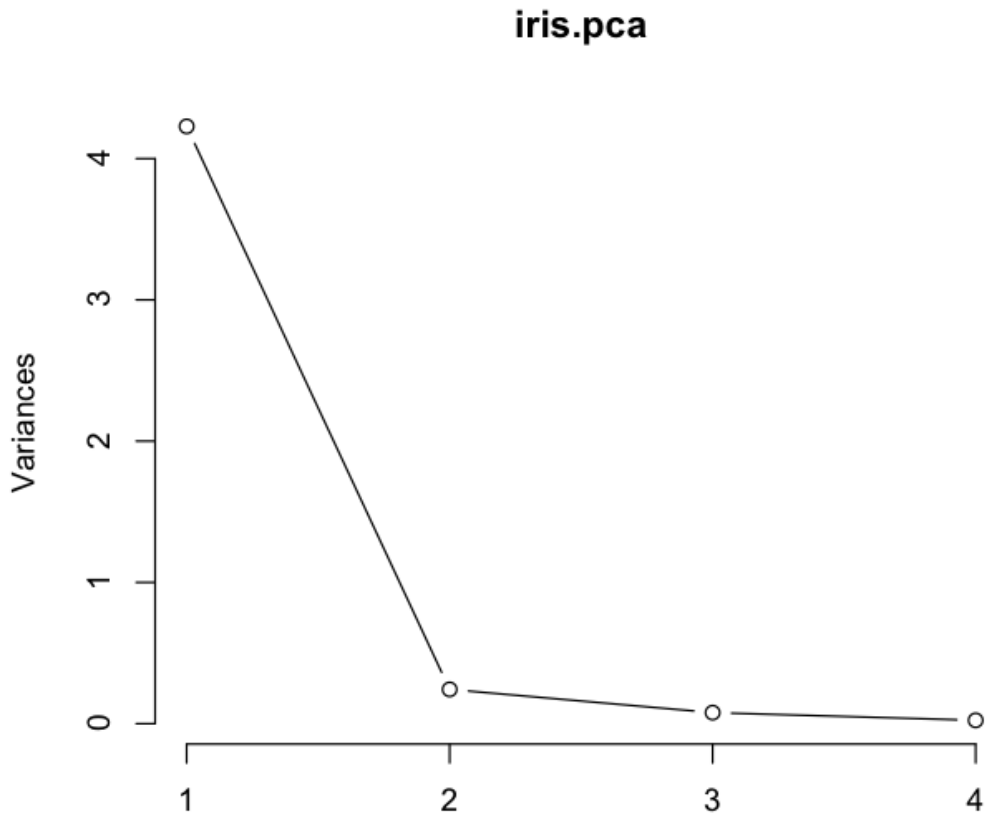
The eigenvalue decomposition of a square matrix $\mathbf{A}$ *is given by:*

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$$

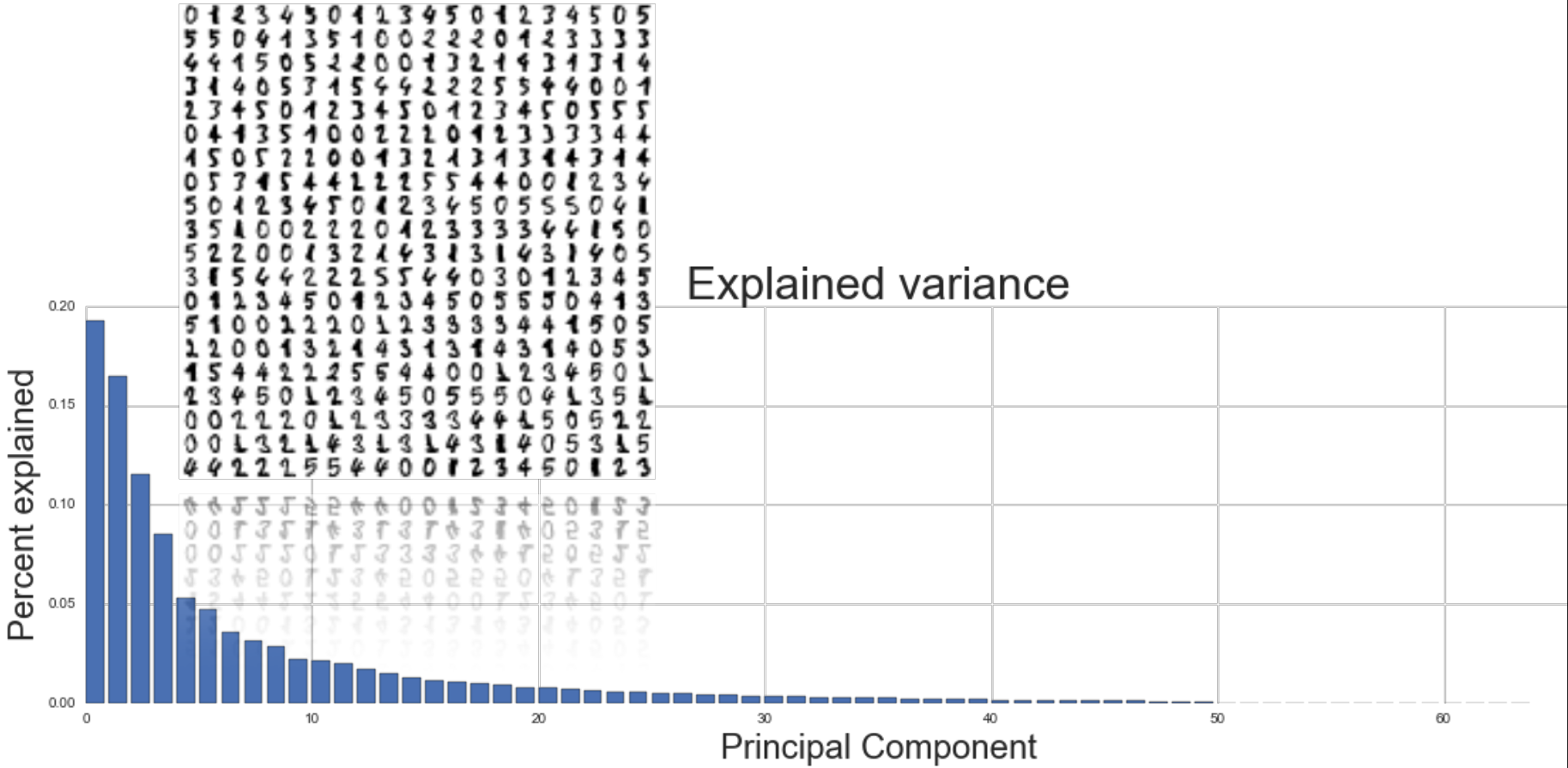
The columns of $\mathbf{Q}$ *are the* **eigenvectors** *of* $\mathbf{A}$*, and the values in* Λ *are the associated* **eigenvalues** *of* $\mathbf{A}$*.*

$$\begin{aligned} A &= \begin{bmatrix} -1/2 & 3/2 \\ 3/2 & -1/2 \end{bmatrix} \\ &= \left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right)\begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right)^T \end{aligned}$$

PRINCIPAL COMPONENT ANALYSIS

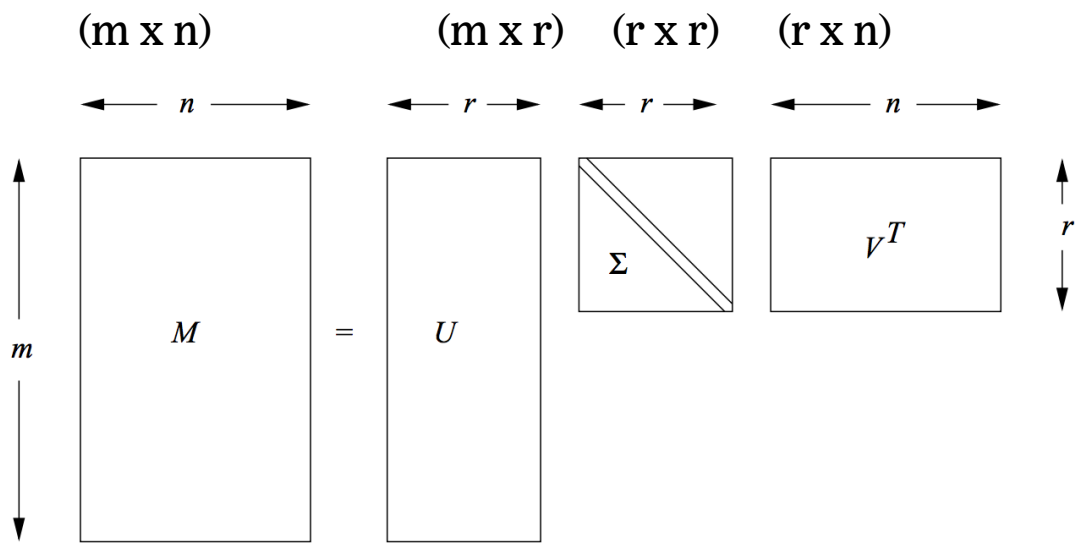


PRINCIPAL COMPONENT ANALYSIS



The singular value decomposition *of* M *is given by:*

$$\mathbf{M} = \mathbf{U}\,\Sigma\,\mathbf{V}^{\mathbf{T}}$$



source: http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

SINGULAR VALUE DECOMPOSITION - EXAMPLE

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}$$

M U Σ V^{T}

M: people -> movies

U: people -> concepts

V: concepts -> movies

Σ: the strength of each of the concepts

source: http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

SVD and PCA are both linear techniques (eg, we use a linear transformation to embed the in a lower-dimensional space).

But as we saw with SVM's, sometimes linear techniques are not sufficient.

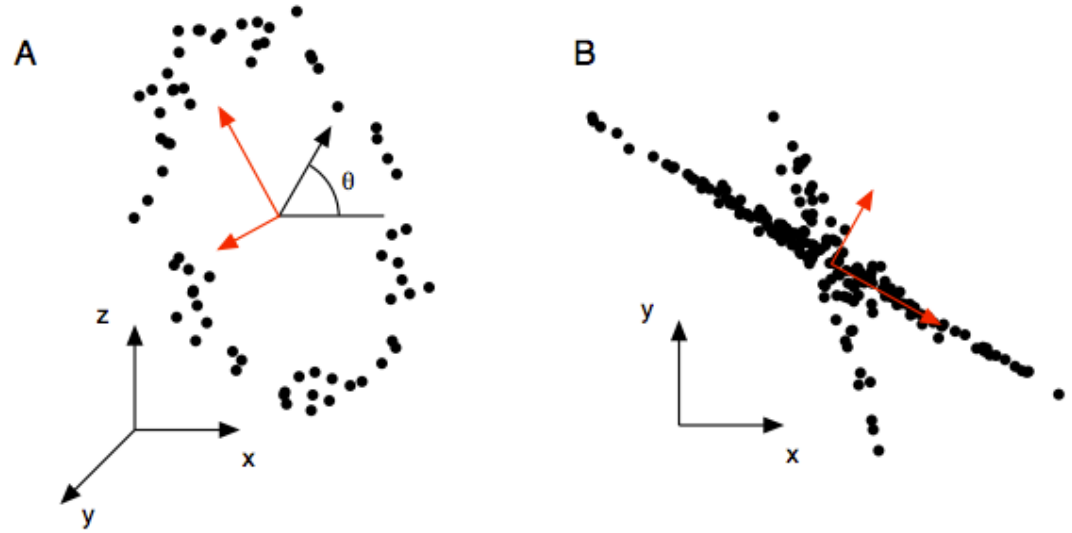
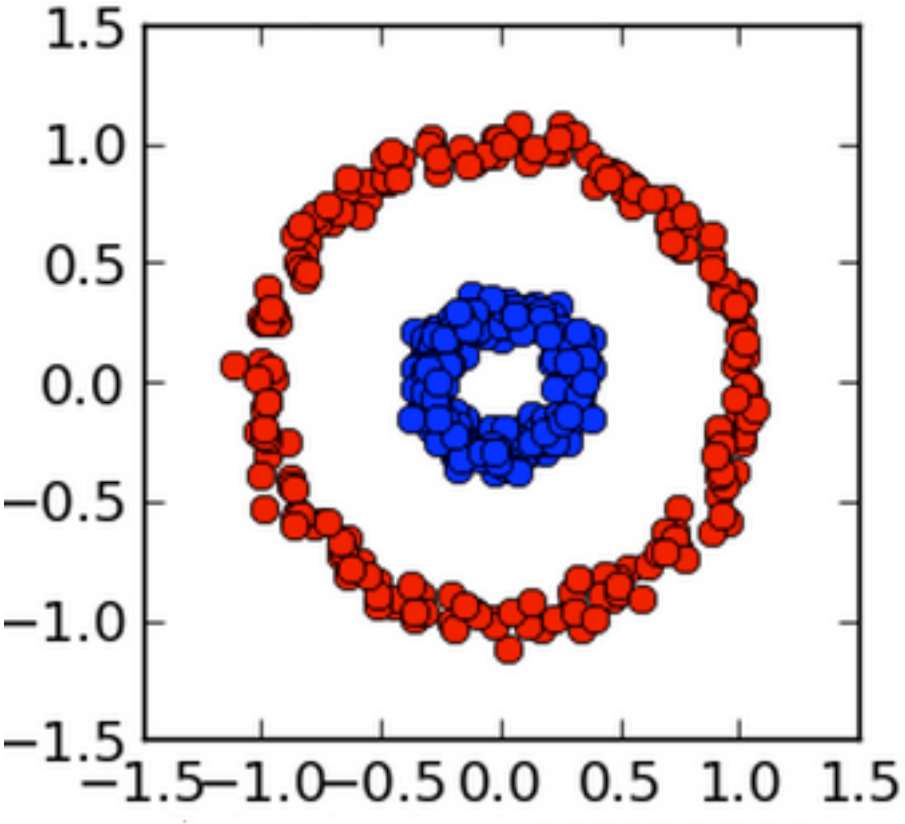


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ, a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Some methods for nonlinear dimensional reduction include:

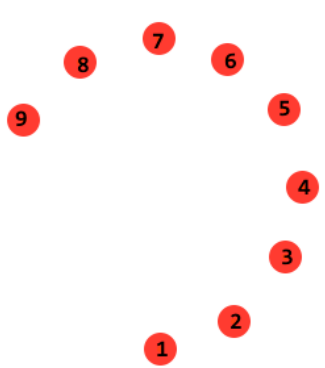
multidimensional scaling: *low-dim embedding that preserves pairwise distances*

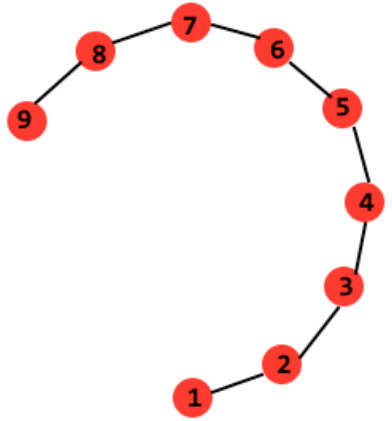
locally linear embedding: *approximates local structure of data*

Some methods for nonlinear dimensional reduction (or manifold learning) include:

kernel PCA: *exploits PCA dependence on inner product (same logic as SVM)*

isomap: *define neighbors and find interpoint distances*



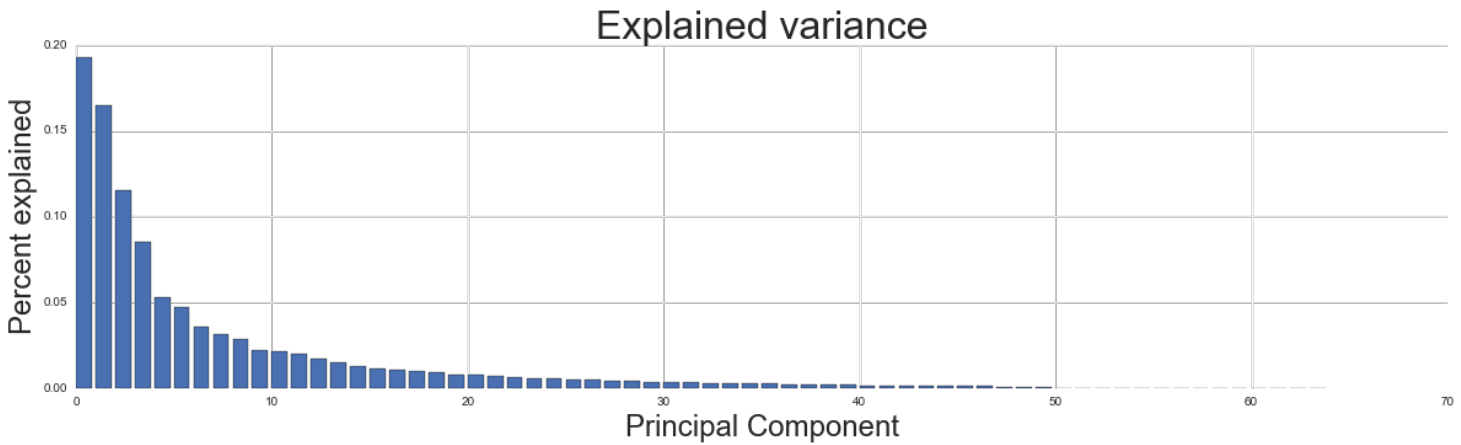


source: http://blog.shriphani.com/2014/11/12/the-isomap-algorithm/

In any case, the key difficulties with dimensionality reduction are:

- *time/space complexity*
- *randomness (eg different results for different runs)*
- *selecting the number of dimensions in the lower-dim subspace*

Furthermore, there's an obvious (bias/variance) tradeoff between the number of subspace dimensions and the size of approximation error.



I. IMBALANCED CLASSES

II. EVALUATION

III. LAB

I. IMBALANCED CLASSES

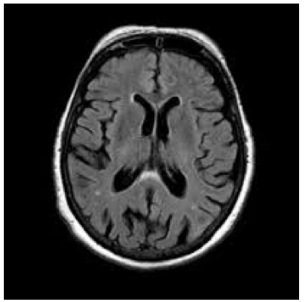
IMBALANCED CLASSES

Your are a data scientist for a project called CancerScreen.

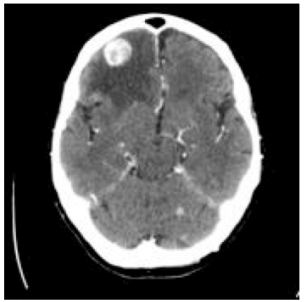
You are tasked with creating a new classifier that classifies radiology images of brains as having cancer or not having cancer.

IMBALANCED CLASSES

In this project, all the images that have been tagged as cancerous will go to a trained physician for further review



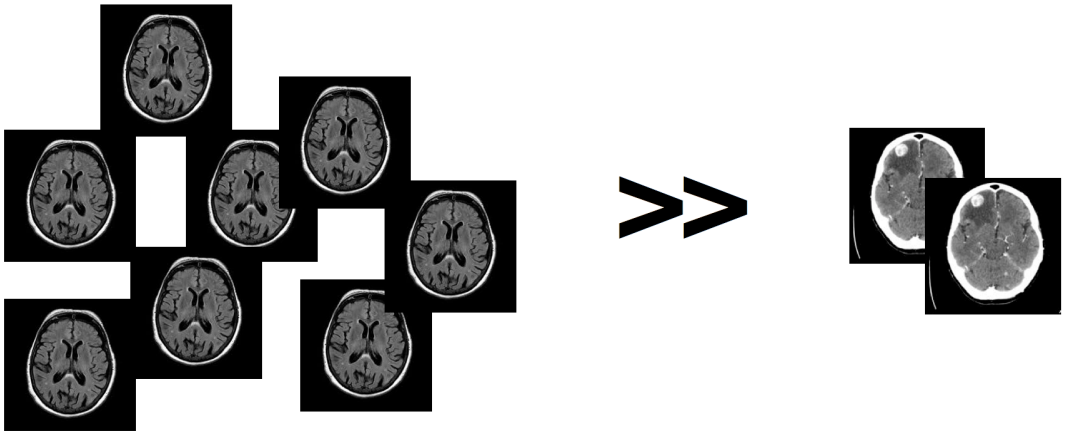
No Cancer



Cancer

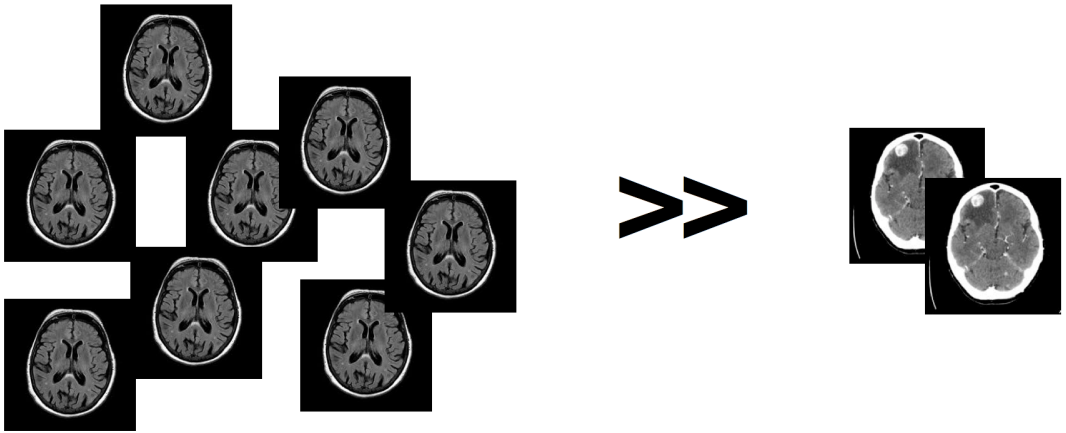
First Issue: *There are a lot more healthier brains than cancerous brains.*

This imbalance will confuse many classifiers as they will only perform well on the dominant class and poorly on the minority class.



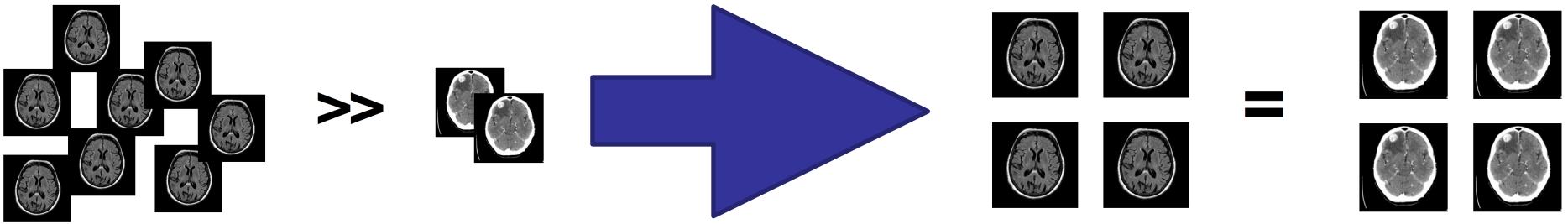
First Issue: *There are a lot more healthier brains than cancerous brains.*

This situation shows up frequently in many fields ex. fraud detection, medical diagnosis, etc.

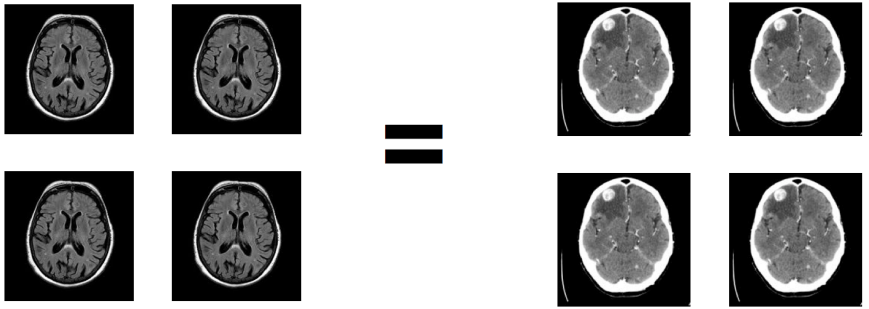


***Solution**:*

Balance your classes and train on this balanced dataset.



IMBALANCED CLASSES



This can be done by:

1. ***Undersampling*** *the dominant class - remove some the majority class so it has less weight*
2. ***Oversampling*** *the minority class - add more of the minority class so it has more weight.*
3. ***Hybrid*** *- doing both*

Undersampling:

Randomly remove elements from the majority class.

Drawback:

Removing data points could lose important information

***Oversampling**:*

Duplicate elements of your minority class

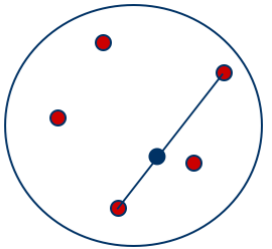
Drawback:

Just replicating randomly minority classes could cause overfit

Smote: *Synthetic Minority Over-sampling Technique*

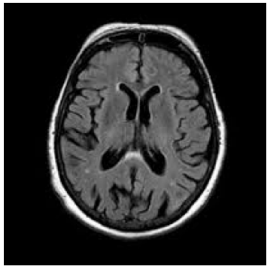
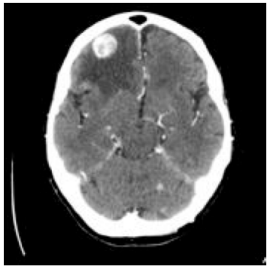
For each minority Sample:

- *Find its k-nearest minority neighbors*
- *Randomly select j of these neighbors*
- *Randomly generate synthetic samples along the lines joining the minority sample and its j selected neighbors*



II. EVALUATION

Second Problem: Not all errors are equal ...

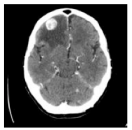
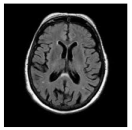
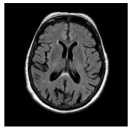
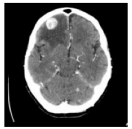
Error 1		Classifier Label: Cancerous	Permissable, because a physician will review it
Error 2		Classifier Label: Non-Cancerous	Not permissable, because this data will be discarded

Most comparisons of machine learning algorithms use classification accuracy.

Problems with this approach:

- *May be different costs associated with Error 1 and Error 2*
- *Training data may not reflect true class distribution*

So we need a more sophisticated model of Error Rate:

TP: An Example that is **positive** and is classified as **positive**		Label: Positive
TN: An Example that is **negative** and is classified as **negative**		Label: Negative
FP: An Example that is **negative** and is classified as **positive**		Label: Positive
FN: An Example that is **positive** and is classified as **negative**		Label: Negative

Metrics (Accuracy):

		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)
	Test outcome negative	**False negative** (Type II error)	**True negative**

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Metrics (Precision & Recall):

		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)
	Test outcome negative	**False negative** (Type II error)	**True negative**

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

PRECISION/RECALL – ANOTHER MEASURE OF PERFORMANCE

Counts: Lions: 4, Tigers: 5, House Cats: 4

Total: 13

PRECISION/RECALL - ANOTHER MEASURE OF PERFORMANCE

We've trained our classifier on tigers and ask it to find all the tigers in this dataset. Here's what it returns:

Classified as Tigers:





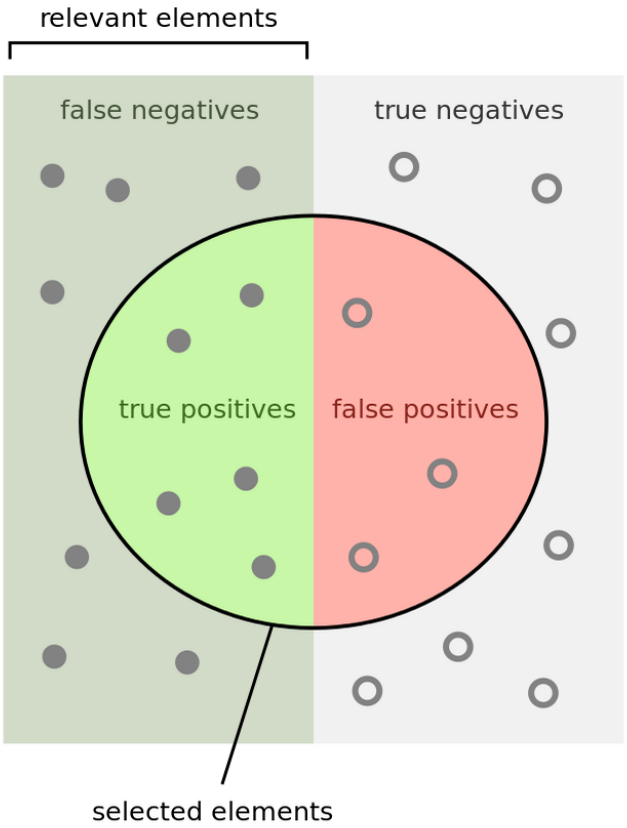

Precision is the percentage of True Positives in your set of results

= 4 / 6 = .66

Recall is True Postives / Total Positives.
Same as TPR

= 4 / 5 = .8

PRECISION/RECALL – ANOTHER MEASURE OF PERFORMANCE



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

source: http://en.wikipedia.org/wiki/Precision_and_recall

Metrics (F score):

		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)
	Test outcome negative	**False negative** (Type II error)	**True negative**

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

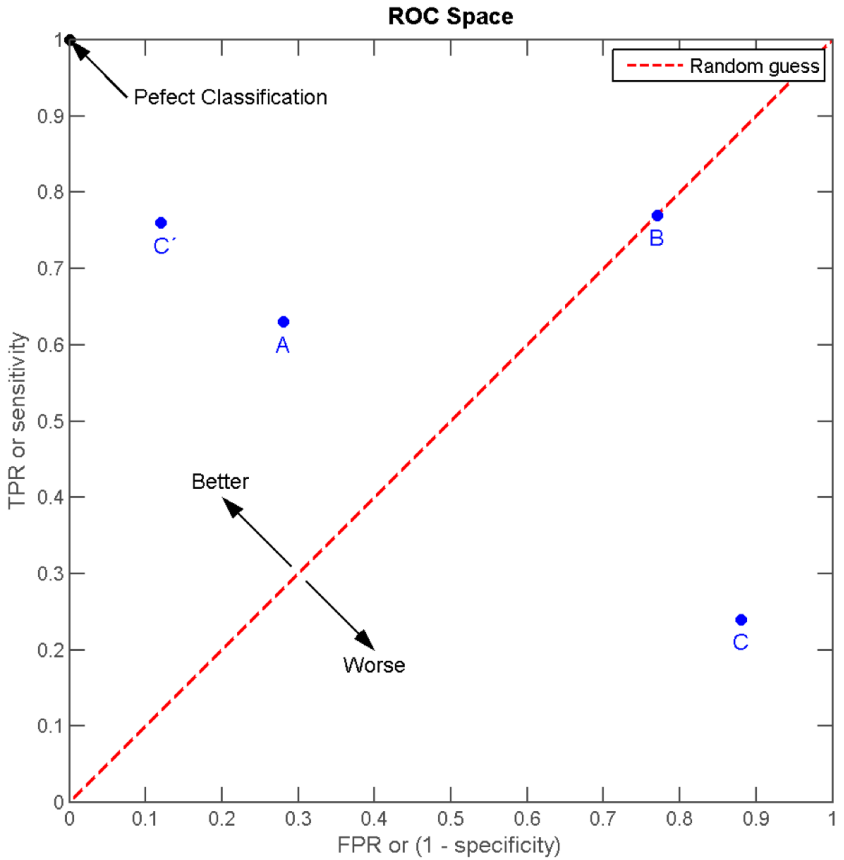
Metrics (true positive rate & false positive rate):

		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)
	Test outcome negative	**False negative** (Type II error)	**True negative**

$$TPR = TP/P = TP/(TP + FN)$$

$$FPR = FP/N = FP/(FP + TN)$$

ROC (RECEIVER OPERATING CHARACTERISTIC)



TP Rate = True Positives / All positives

FP Rate = False Positives / All Negatives

ROC (RECEIVER OPERATING CHARACTERISTIC)

A

TP=63	FP=28	91
FN=37	TN=72	109
100	100	200

TPR = 0.63

FPR = 0.28

ACC = 0.68

B

TP=77	FP=77	154
FN=23	TN=23	46
100	100	200

TPR = 0.77

FPR = 0.77

ACC = 0.50

C

TP=24	FP=88	112
FN=76	TN=12	88
100	100	200

TPR = 0.24

FPR = 0.88

ACC = 0.18

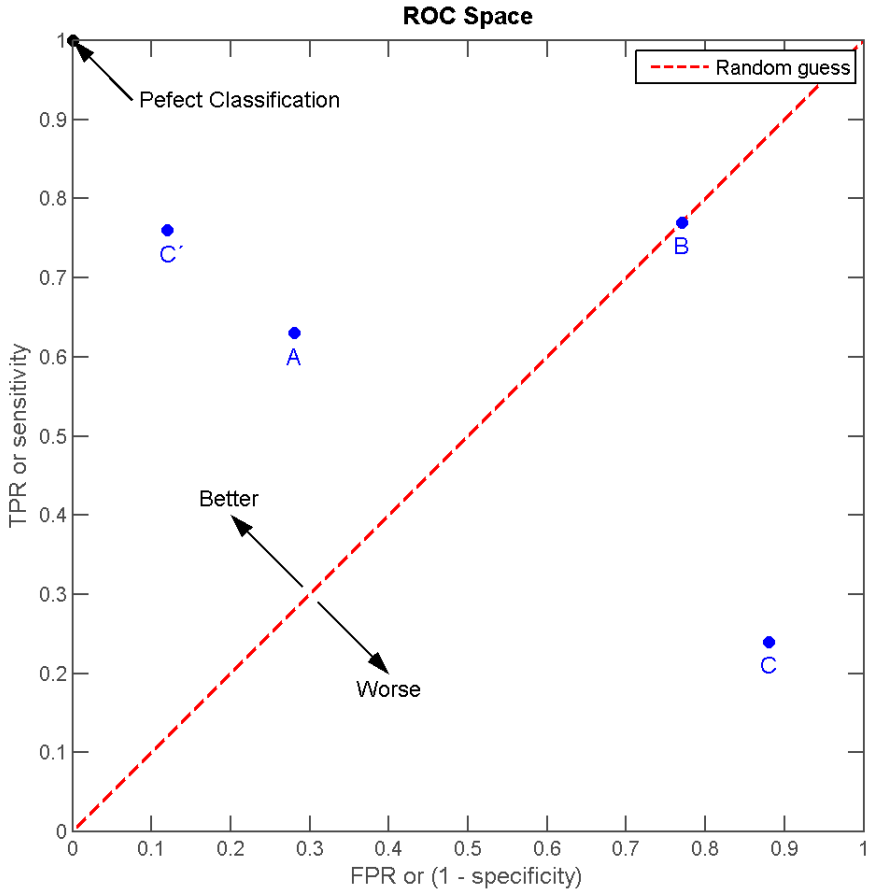
C'

TP=76	FP=12	88
FN=24	TN=88	112
100	100	200

TPR = 0.76

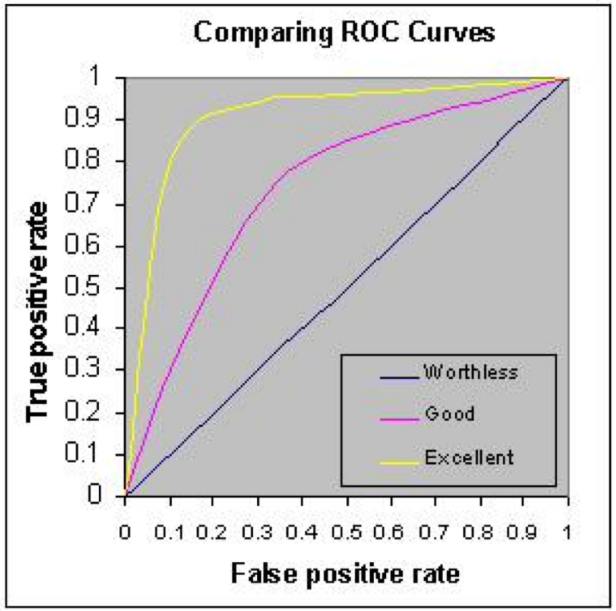
FPR = 0.12

ACC = 0.82



source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC Curves show the relationship between the TP Rate and the FP Rate as we vary the decision threshold for the classifier



Evaluating A Classifier using ROC:

- *We evaluate a classifier by measuring the area under the curve for its ROC curve.*
- *The Greater area under the curve, the more effective the classifier.*
- *Then for our chosen classifier, we pick an appropriate decision threshold.*
- *In general, we pick the decision threshold that gets us closest to the upper left corner.*

Review for Imbalanced Classes

- *Balance your dataset so that the number of elements in each class are equal*
- *Train different classifiers on this balanced data*
- *For each classifier, evaluate the performance*
    - *if you prefer ROC, create an ROC curve and compute the Area under the Curve (AUC)*
- *For the classifier with the greatest AUC, pick the appropriate decision threshold given the specifics of your problem*

III. LAB