2024 LLNL DSC Mini-Crash-Course

Vagelis Papalexakis
UC Riverside

M A D Lab @ UCR
Multi Aspect Data

UC Riverside
July 2024

Outline

- Data Visualization & Manipulation
- Basic Supervised Concepts
- Advanced Supervised Concepts
- Parting Thoughts

Always visualize your data!

Data set		1-3	1	2	3		4	4
Variable		x	y	y	y		x	y
Obs. no.	1	10.0	8.04	9.14	7.46		8.0	6.58
	2	8.0	6.95	8.14	6.77		8.0	5.76
	3	13.0	7.58	8.74	12.74		8.0	7.71
	4	9.0	8.81	8.77	7.11		8.0	8.84
	5	11.0	8.33	9.26	7.81		8.0	8.47
	6	14.0	9.96	8.10	8.84		8.0	7.04
	7	6.0	7.24	6.13	6.08		8.0	5.25
	8	4.0	4.26	3.10	5.39		19.0	12.50
	9	12.0	10.84	9.13	8.15		8.0	5.56
	10	7.0	4.82	7.26	6.42		8.0	7.91
	11	5.0	5.68	4.74	5.73		8.0	6.89

TABLE. Four data sets, each comprising 11 (x, y) pairs.

Number of observations (n) = 11
Mean of the x's ($\bar{x}$) = 9.0
Mean of the y's ($\bar{y}$) = 7.5
Regression coefficient (b_1) of y on x = 0.5
Equation of regression line: $y = 3 + 0.5\,x$
Sum of squares of $x - \bar{x}$ = 110.0
Regression sum of squares = 27.50 (1 d.f.)
Residual sum of squares of y = 13.75 (9 d.f.)
Estimated standard error of b_1 = 0.118
Multiple R^2 = 0.667

Four data sets with exactly the same statistics (Anscombe's quartet)

F. J. Anscombe (1973) Graphs in Statistical Analysis, The American Statistician, 27:1, 17-21

Those datasets look *very* different!

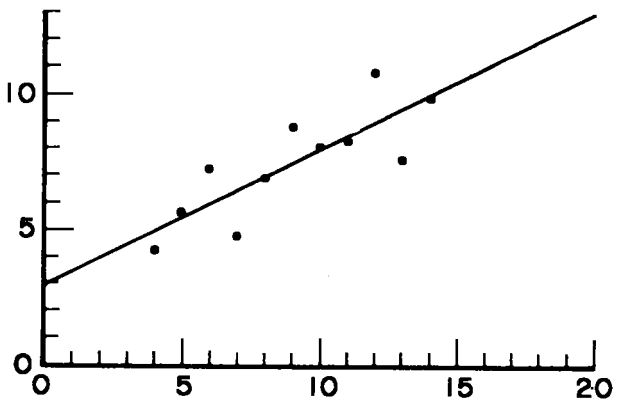


Figure 1

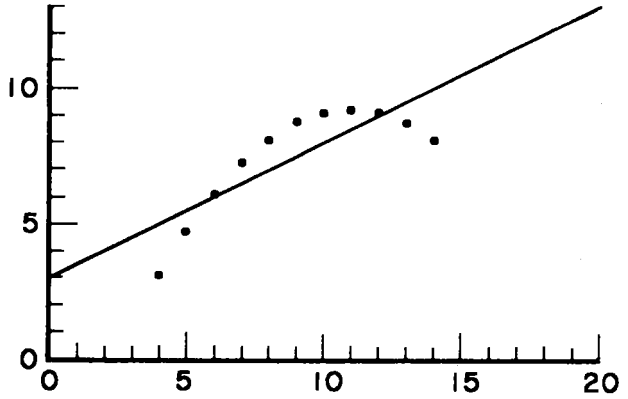


Figure 2

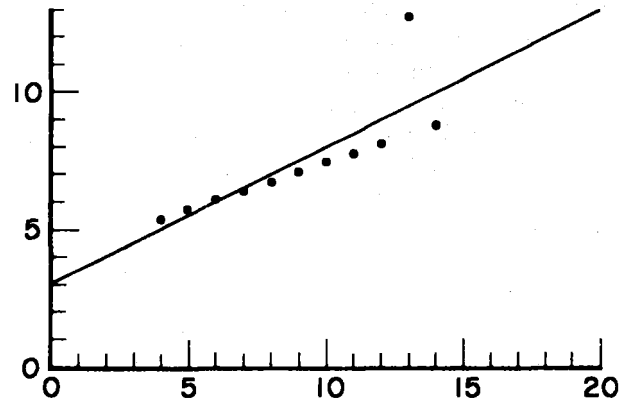


Figure 3

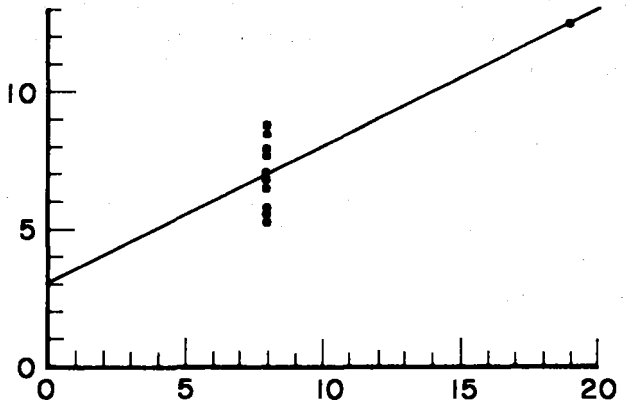
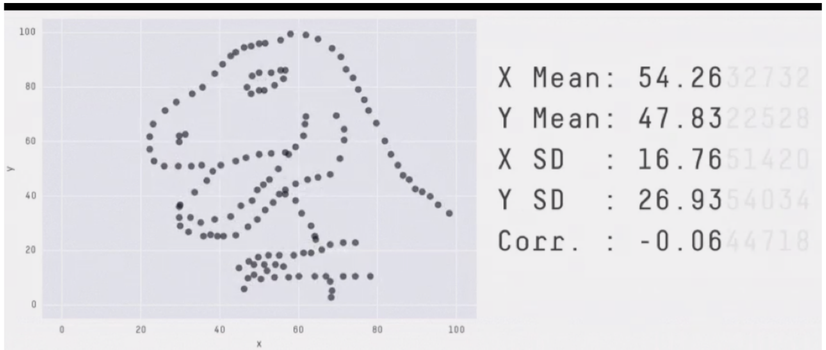
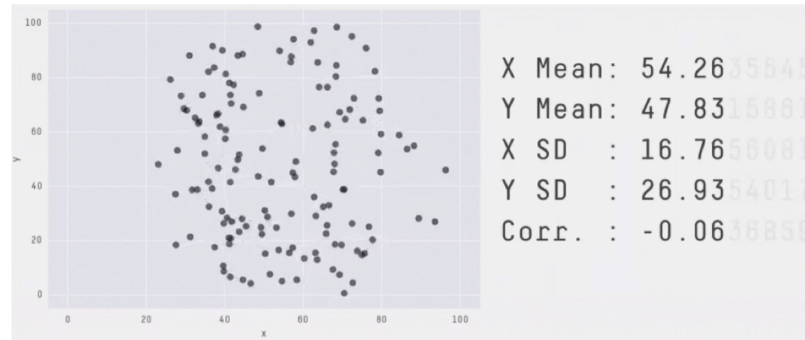


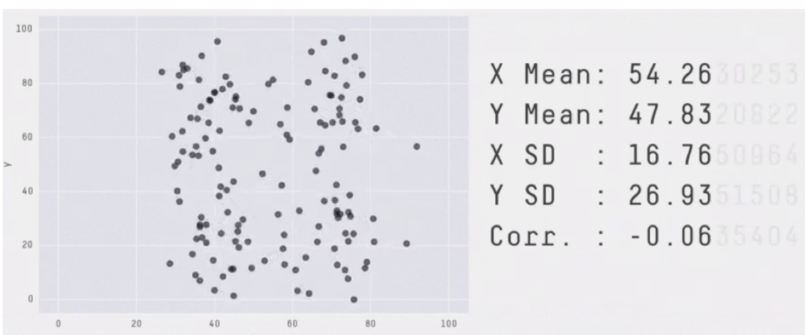
Figure 4

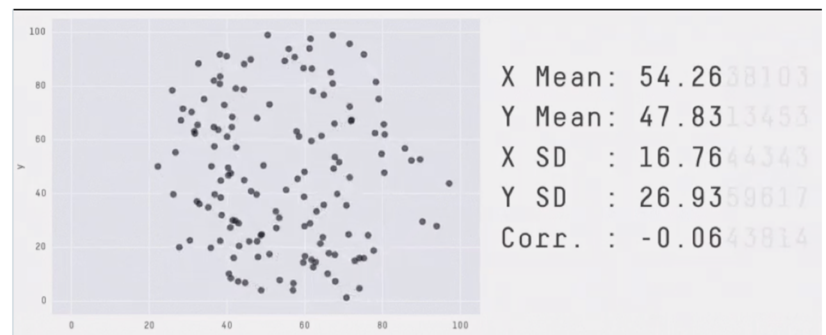
F. J. Anscombe (1973) Graphs in Statistical Analysis, The American Statistician, 27:1, 17-21

It gets even worse: Anscombosaurus





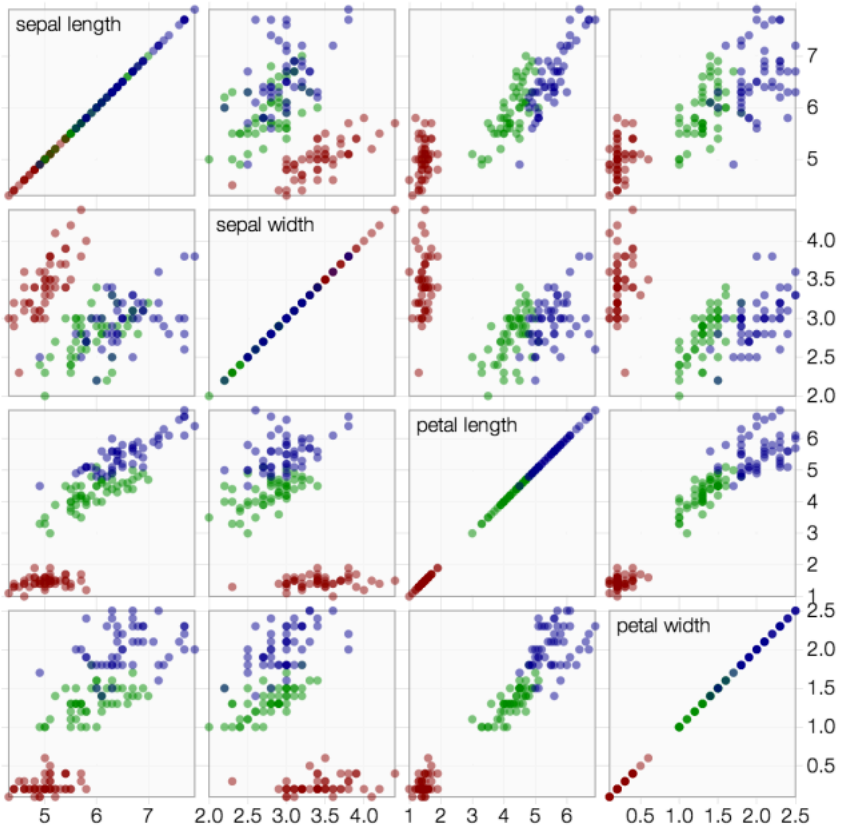




http://www.thefunctionalart.com/2016/08/download-datasaurus-never-trust-summary.html

https://twitter.com/maartenzam/status/770723795518812160

Scatterplot Matrices



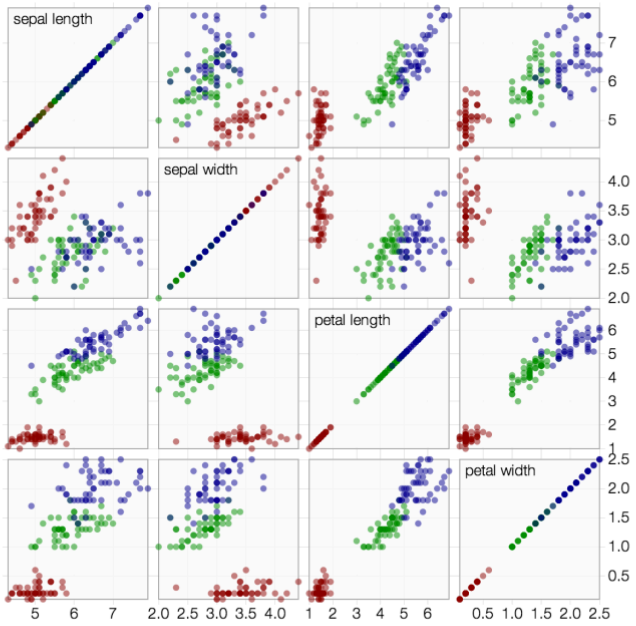
- *Iris setosa*
- *Iris versicolor*
- *Iris virginica*

Edgar Anderson's *Iris* data set
scatterplot matrix

Data: https://archive.ics.uci.edu/ml/datasets/Iris

Source: http://complexdatavisualized.com/scatterplot-matrix-mike-bostock-2013/screen-shot-2013-08-08-at-2-41-46-pm/

How do I visualize multidimensional data???



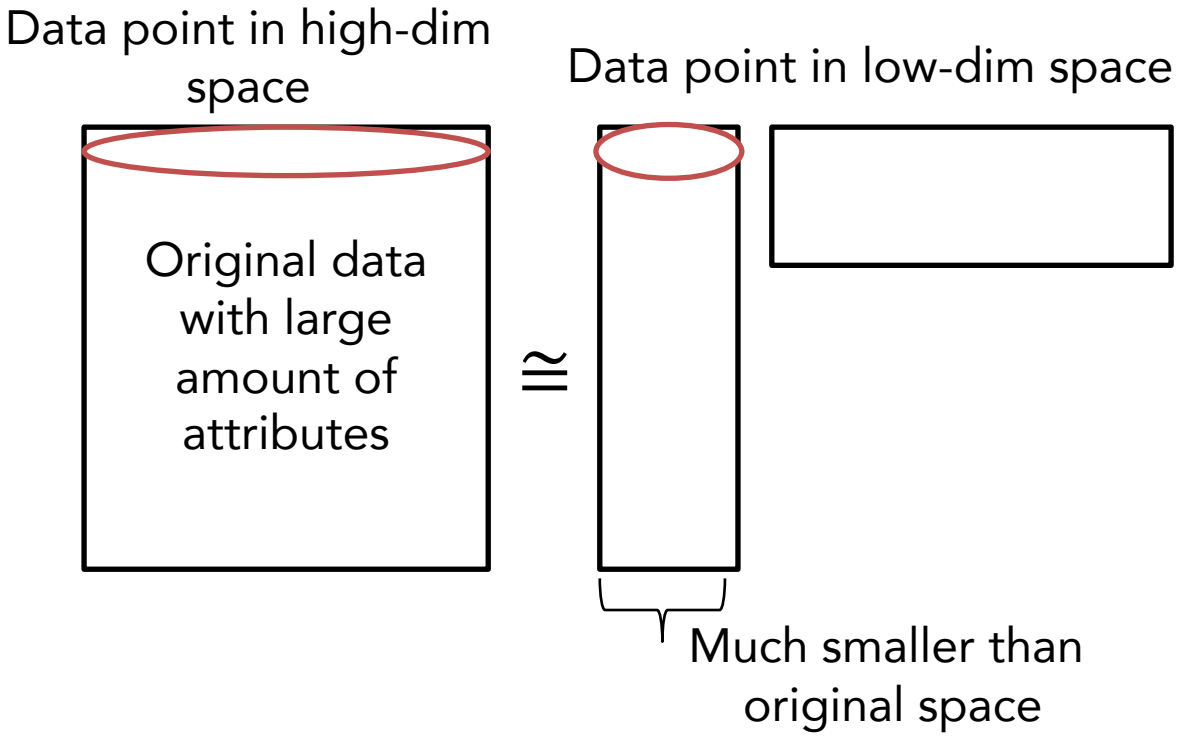
- *Iris setosa*
- *Iris versicolor*
- *Iris virginica*

Edgar Anderson's *Iris* data set
scatterplot matrix

Set of scatterplots may be very hard to inspect if #features is very large!

Principal Component Analysis (PCA)

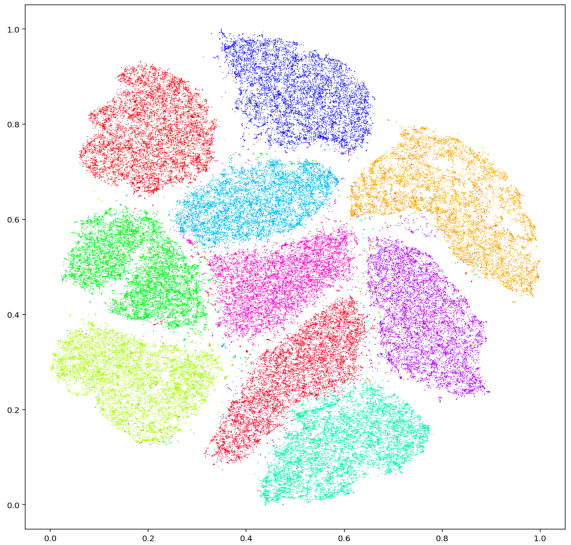
- Center (aka make zero-mean) and normalize the data
- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction.
- Works for numerical data only



t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Currently probably the most popular visualization method
- Non-linear projection of points to a low-dim (2D) space which tries to respect distances between points.
- A lot of implementations: https://lvdmaaten.github.io/tsne/

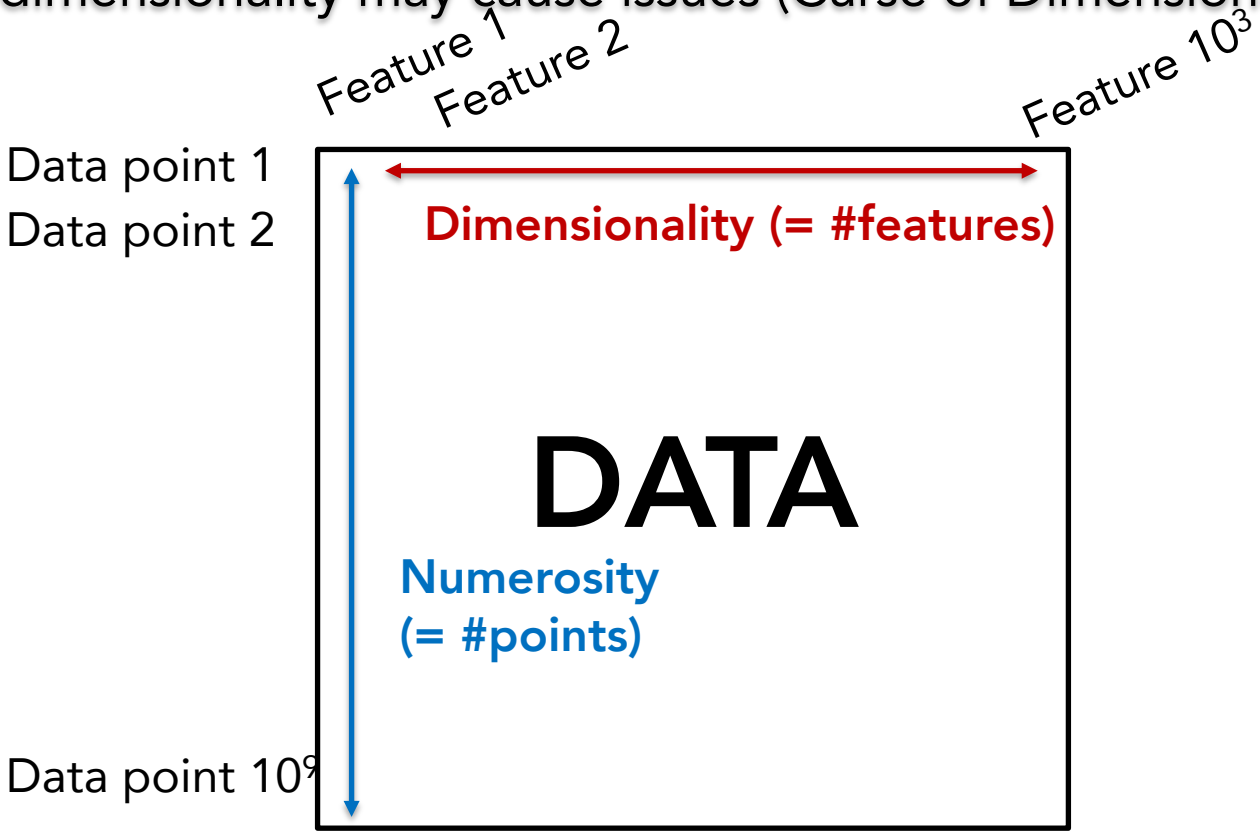
t-SNE of the MNIST dataset



Images from Wikipedia

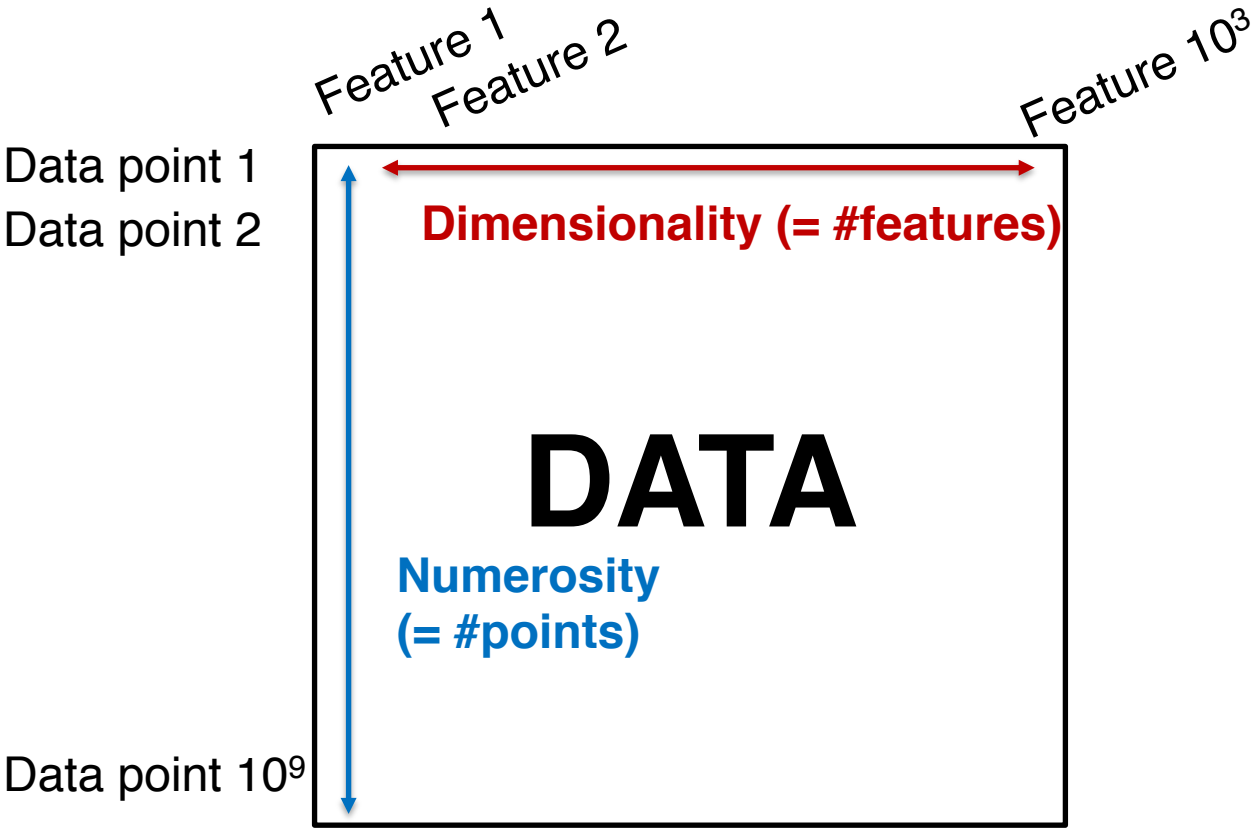
Data Reduction

- **Data reduction**: Obtain reduced representation of data that produces the same (or almost the same) analytical results
- Why data reduction? —Complex data analysis may take a very long time to run on the complete data set.
- High dimensionality may cause issues (Curse of Dimensionality)



Data Reduction Strategies

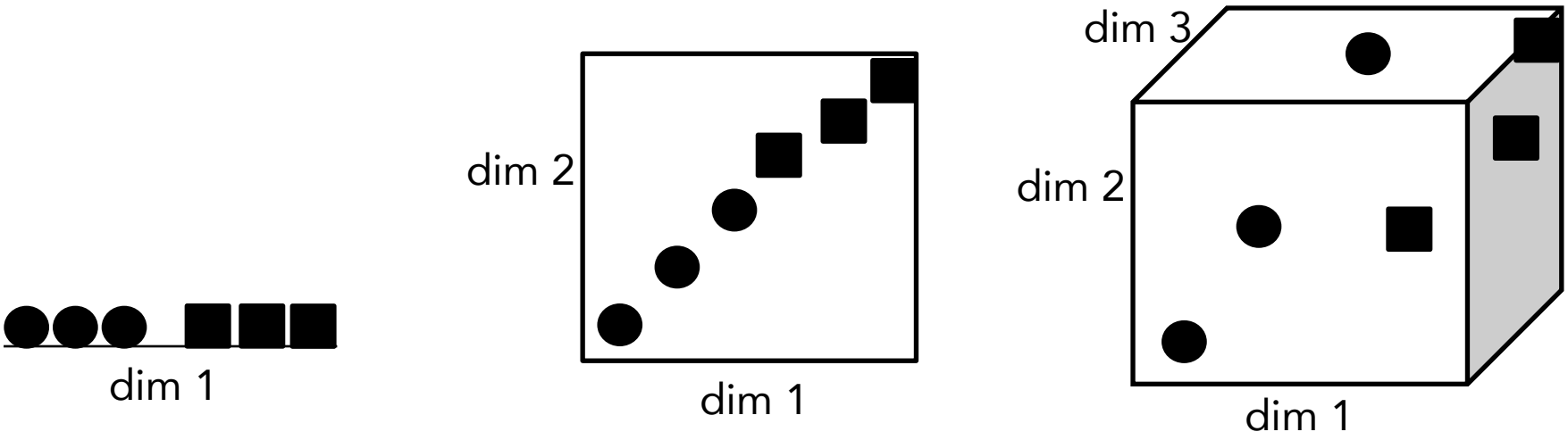
- Data reduction strategies
  - **Dimensionality reduction**, e.g., remove unimportant attributes
  - **Numerosity reduction** (some simply call it: Data Reduction)



Curse of Dimensionality

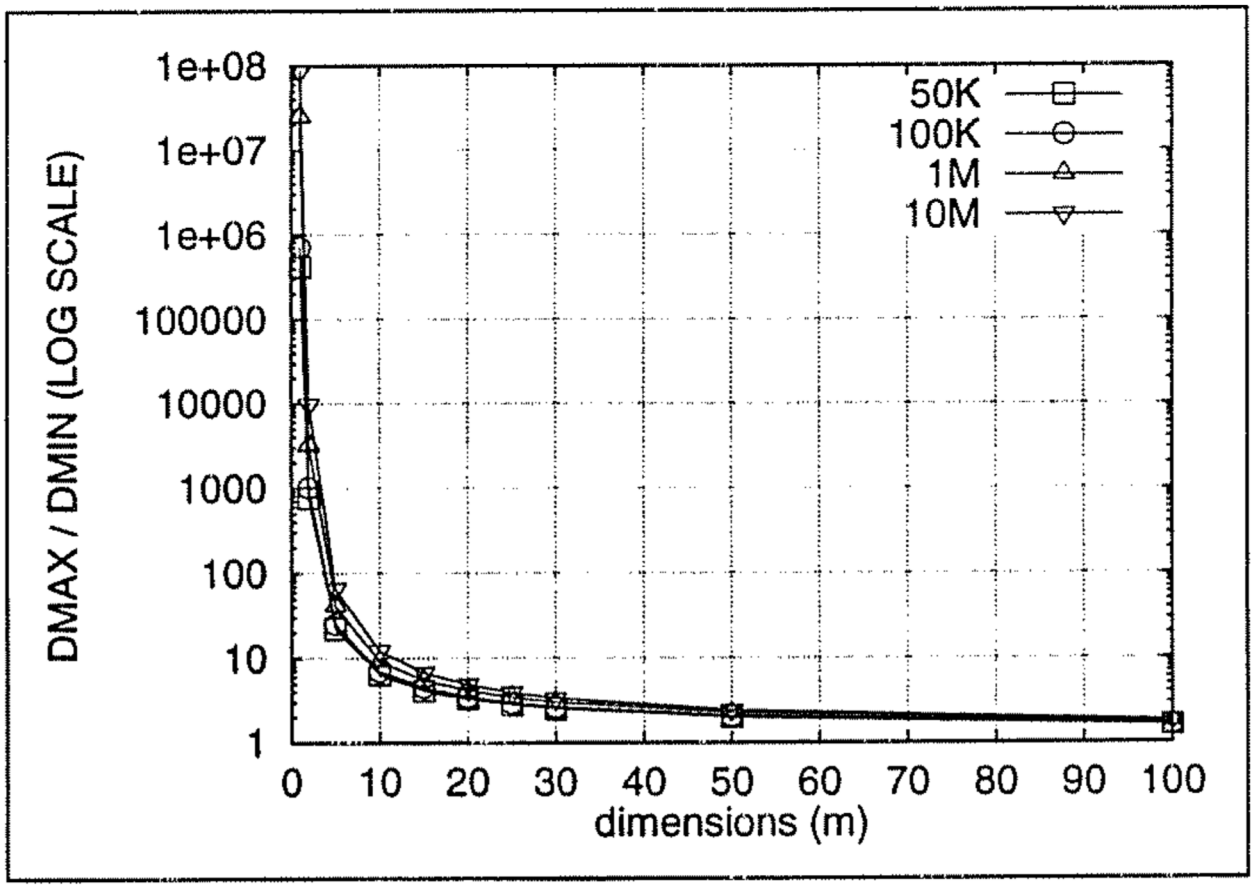
- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially

Curse of Dimensionality



As dims grow, space becomes sparser!
Need exponentially more data points to be able to “learn” something useful

Curse of Dimensionality

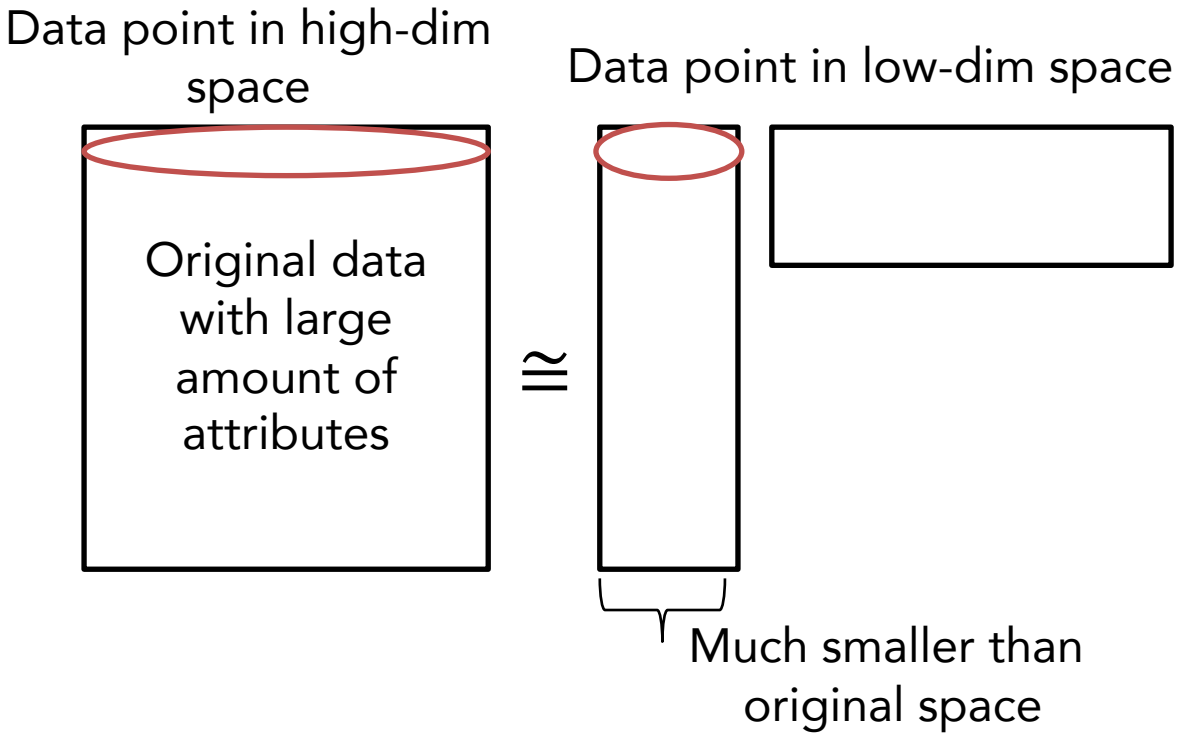


Ratio of furthest to closest point diminishes as the #dims grows!!!

Beyer et al., "When is Nearest Neighbor Meaningful?"
https://minds.wisconsin.edu/bitstream/handle/1793/60174/TR1377.pdf

Dimensionality reduction with PCA

- Center (aka make zero-mean) and normalize the data
- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction.
- Works for numerical data only



Feature Selection

- Another way to reduce dimensionality of data
- **Redundant attributes**
  - E.g., purchase price of a product and the amount of sales tax paid
- **Irrelevant attributes**
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is irrelevant to the task of predicting students' GPA

Heuristic Search in Feature Selection

- There are 2^d possible attribute combinations of d attributes
- Typical heuristic attribute selection methods:
  - Best single attribute under the attribute independence assumption: choose by significance tests
  - Best step-wise feature selection:
    - The best single-attribute is picked first
    - Then next best attribute condition to the first, ...
  - Step-wise attribute elimination:
    - Repeatedly eliminate the worst attribute

Feature Engineering

- Create new features that can capture the important information in a data set more effectively than the original ones
  - Combine features in domain knowledge driven manner
  - Transform to a new space

- Feature engineering is being replaced by **Representation Learning**
  - Deep learning can be seen as a form of automated feature engineering from raw data (e.g., pixels).
  - Time and effort that was spent in feature engineering now is spent on architecture engineering

Important distinction

- There are largely two categories of dimensionality reduction:
  - **Task specific**:
    - Select subset of features or generate new reduced feature representation that best works for a given task
  - **Task independent**:
    - Select subset of features or generate new reduced feature representation that "retains most of the information" of the data
- Need to know which category we are using every time!

Outline

- Data Visualization & Manipulation
- Basic Supervised Concepts
- Advanced Supervised Concepts
- Parting Thoughts

Regression & Classification

- **Numerical Prediction/Regression**
  - models continuous-valued functions (regression), i.e., predicts unknown or missing values
  - ***This is primarily the terminology in the Data Science literature***
  - ***In ML literature, "regression" often refers to generally modeling the relationship between independent and dependent variables (regardless of the type of the target variable)***
  - For simplicity, we will be using the distinction wrt target variable being numerical in regression.
- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

Model-based Learning

- Model construction / Training
  - Each data point is assumed to belong to a predefined class, as determined by the **class label** or **the value associated to it**
  - The set of points used for model construction is **training set**
  - The model is represented as classification rules, decision trees, or parameters of a mathematical formula

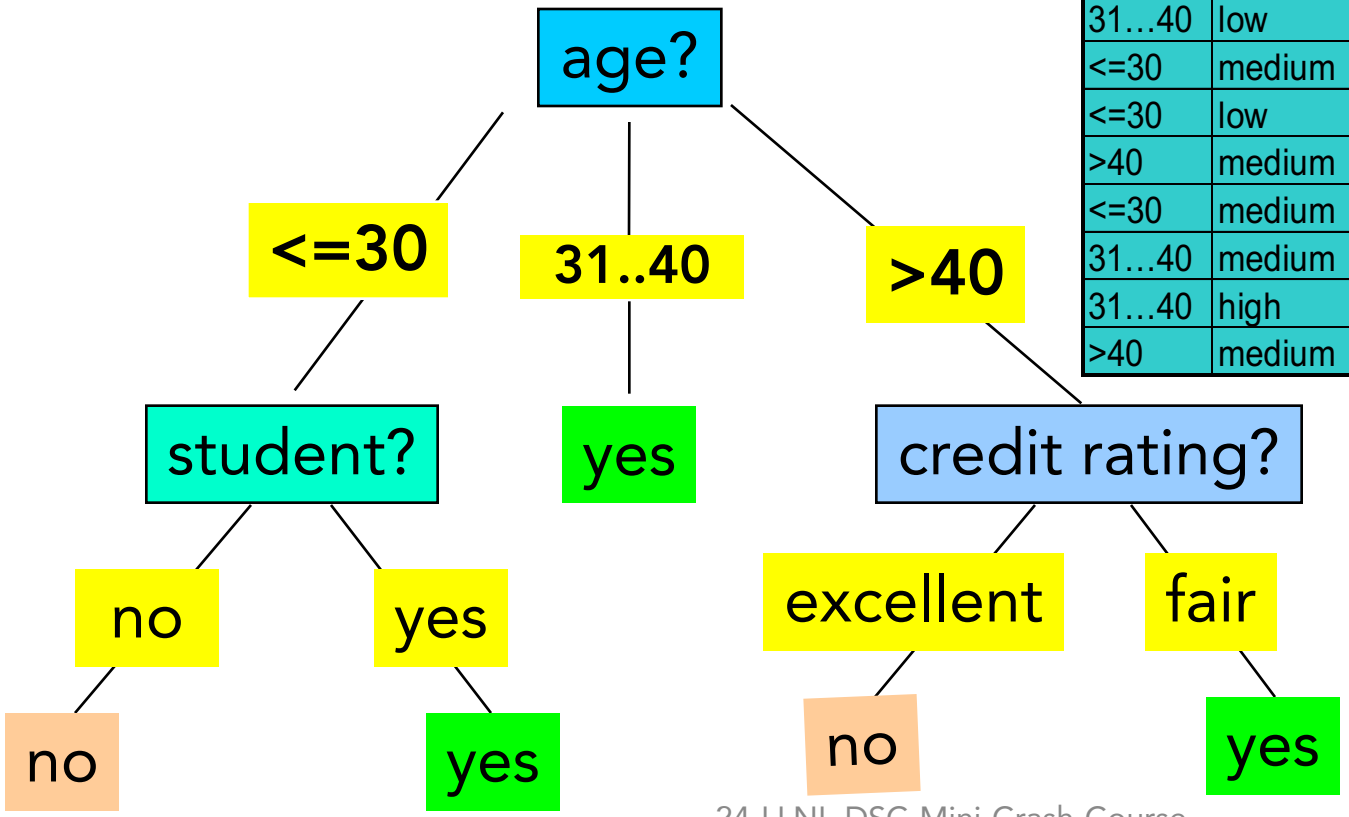
Model-based Learning

- Model usage / Testing:
  - Classifying/predicting future or unknown objects
  - **Estimate performance** of the model
    - The known label of test sample is compared with the classified result from the model
    - The predicted value is compared against the actual value
    - **Test set** is *independent* of training set (otherwise bad things can happen!!)
  - If the performance is acceptable, use the model to **unseen** data points

Decision Tree Training: An Example

- Training data set: Buys_computer
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



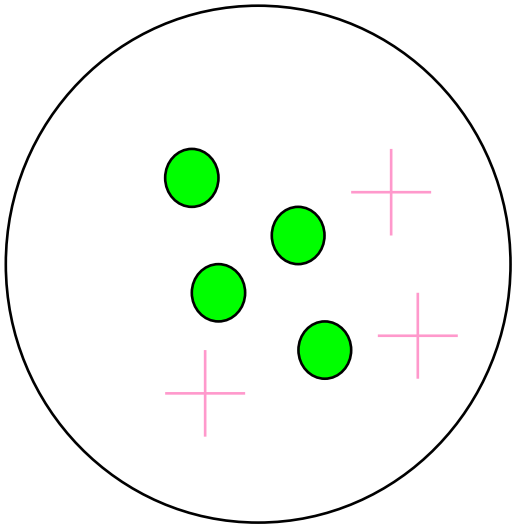
Algorithm for Decision Tree Training

- Basic (greedy) algorithm
  - Tree is constructed in a **top-down recursive divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Most commercial trees use variations of this algorithm

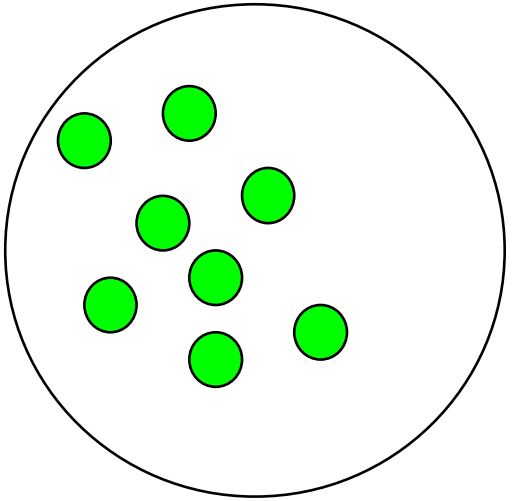
Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  1. All samples for a given node belong to same class
  2. There are no remaining attributes for further partitioning – use **majority vote** for the leaf
  3. There are no samples left

How to split?

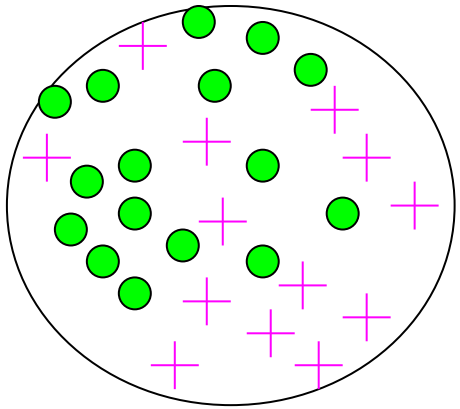


Need to create a “pure” set of data points

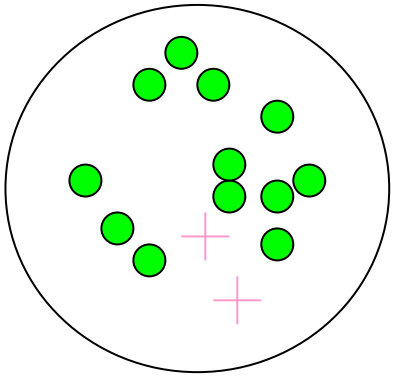


Source: http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

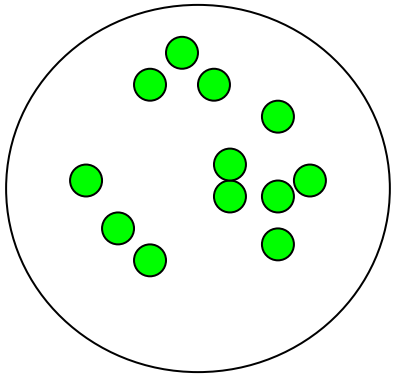
Levels of “impurity"



Very Impure



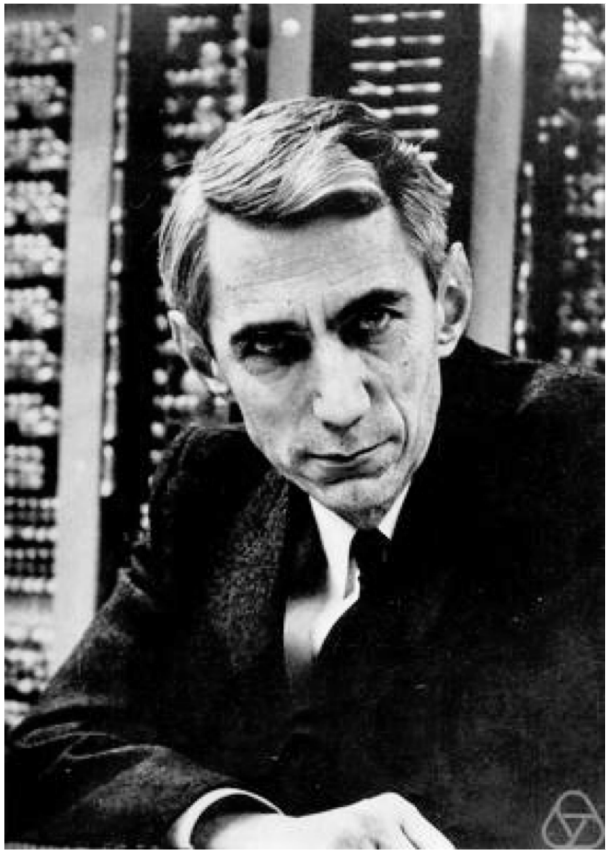
Medium Impure



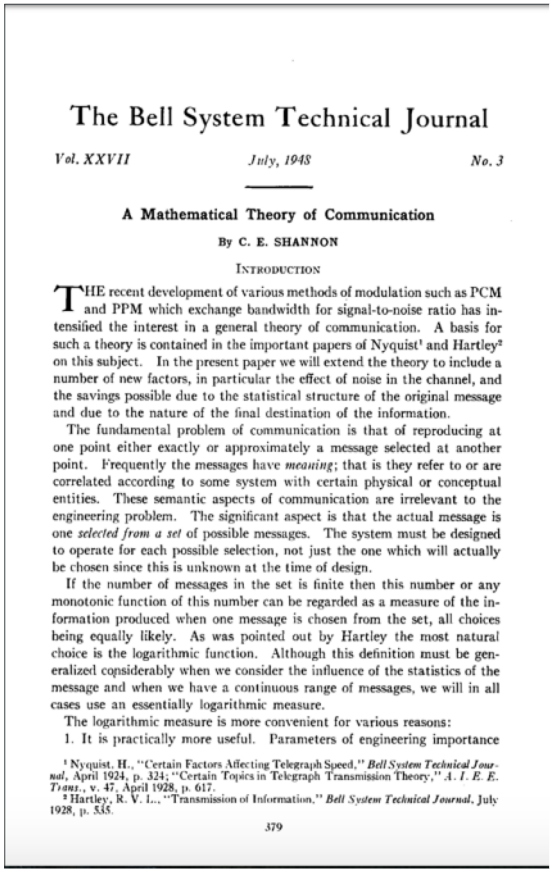
Pure

Source: http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

Information Theory



Img from wikipedia.org



The Bell System Technical Journal

Vol. XXVII *July, 1948* *No. 3*

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance

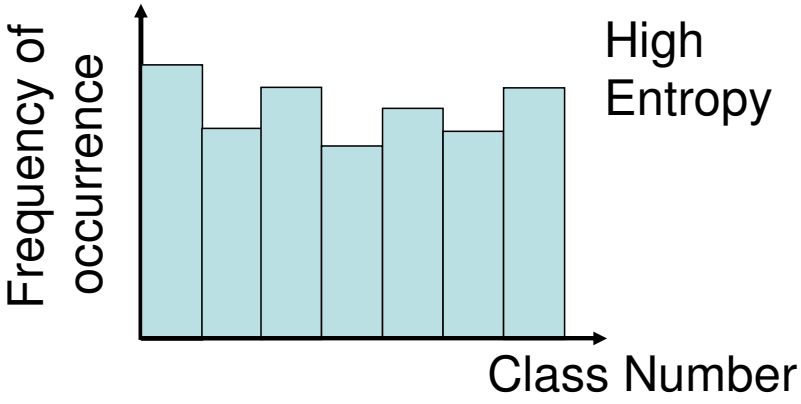
[1] Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A. I. E. E. Trans.*, v. 47, April 1928, p. 617.

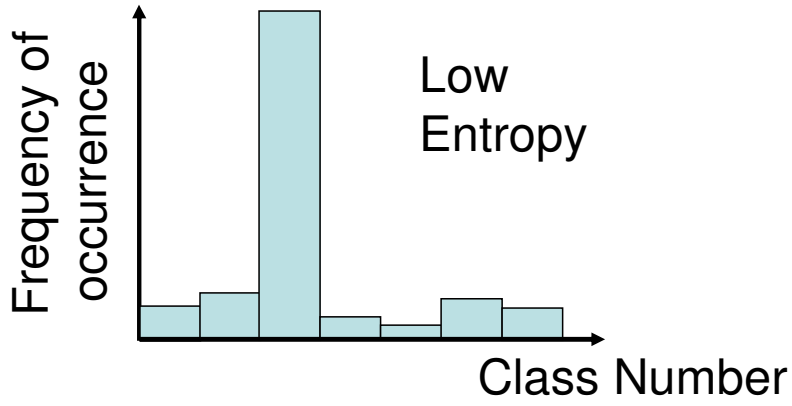
[2] Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

379

http://ieeexplore.ieee.org/document/6773024/?arnumber=6773024

Entropy as "impurity" measure





$$H = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Entropy quantifies uncertainty

Source: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf

Occam's Razor

- Also known as "Law of Parsimony"
- Simple solutions are preferred

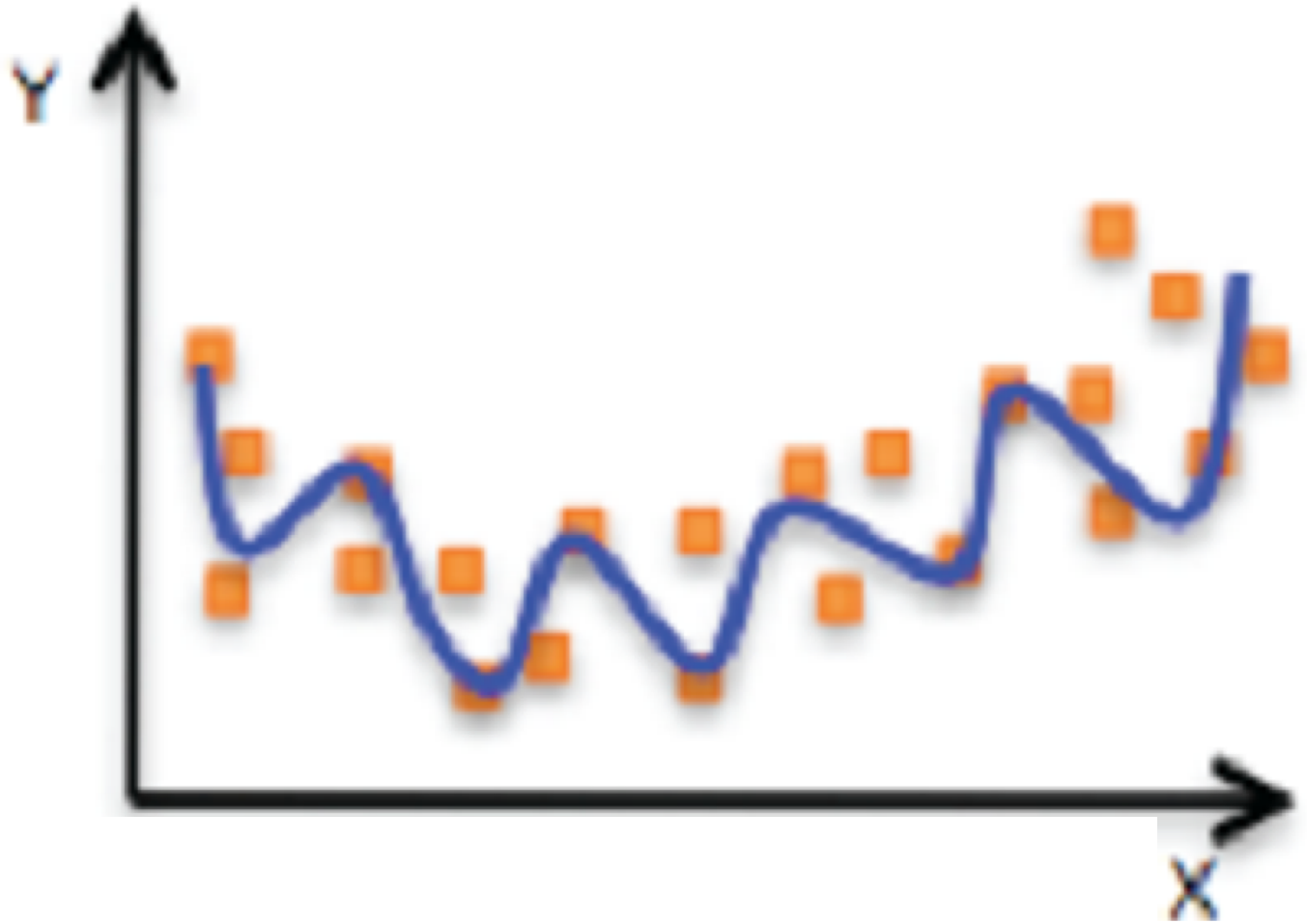
Img from Wikipedia

William of Ockham
(1287–1347)

Why are decision trees popular?

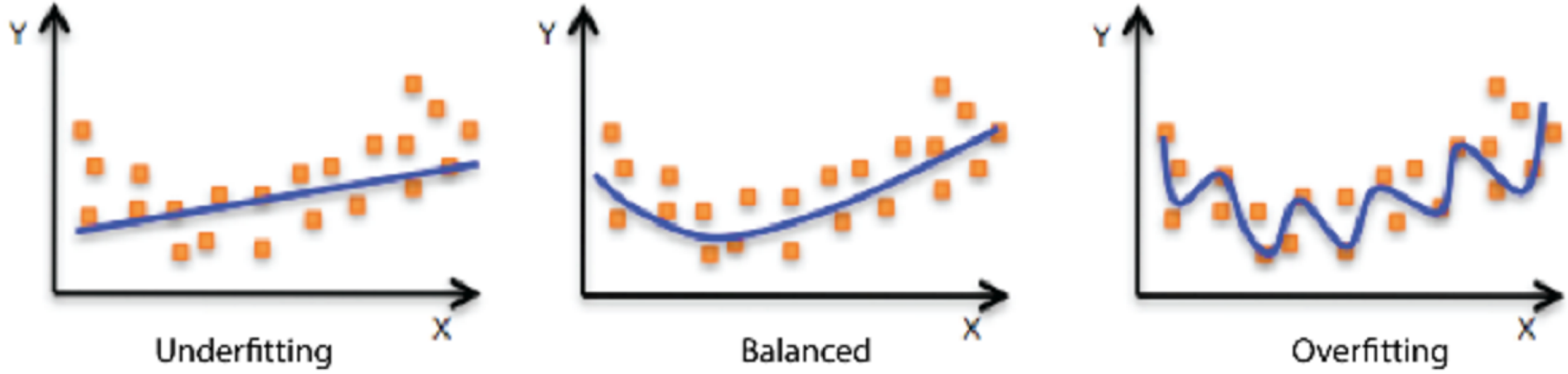
- Relatively faster learning speed (than other classification methods)
- Convertible to simple and easy to understand classification rules
- Comparable classification accuracy with other methods

What is wrong with this?



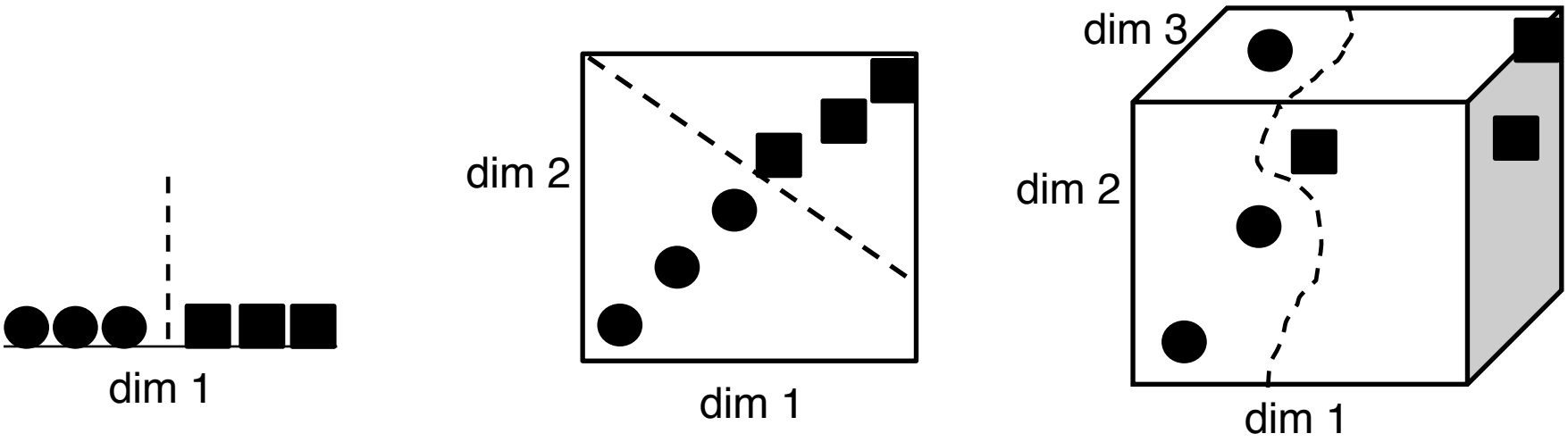
Overfitting

- Happens when the model fits almost perfectly the training data
- May not be able to generalize to new **unseen** examples



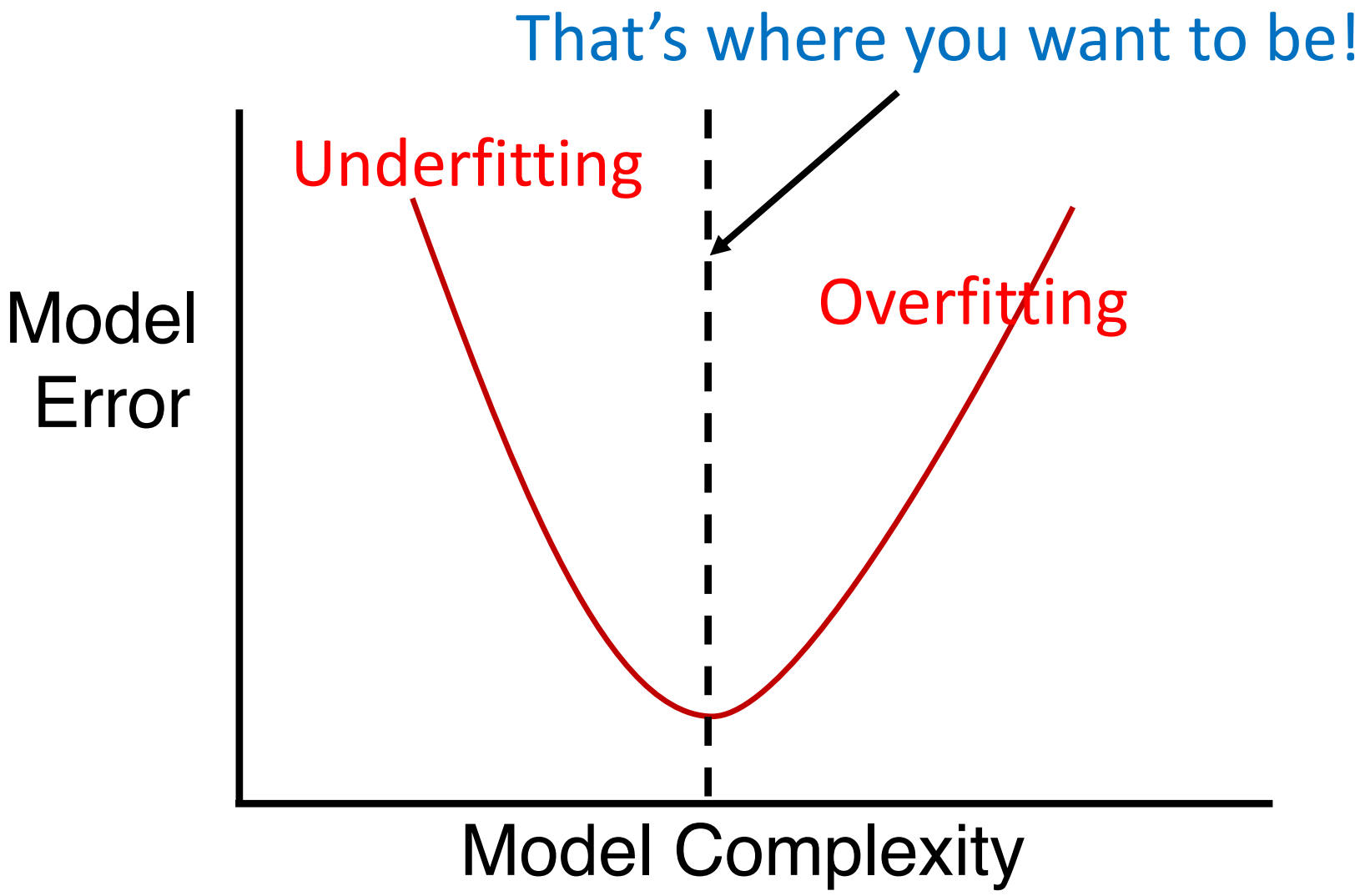
Source: http://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Overfitting & Curse of Dimensionality



Need exponential number of data so that we don't overfit as dims grow !!!

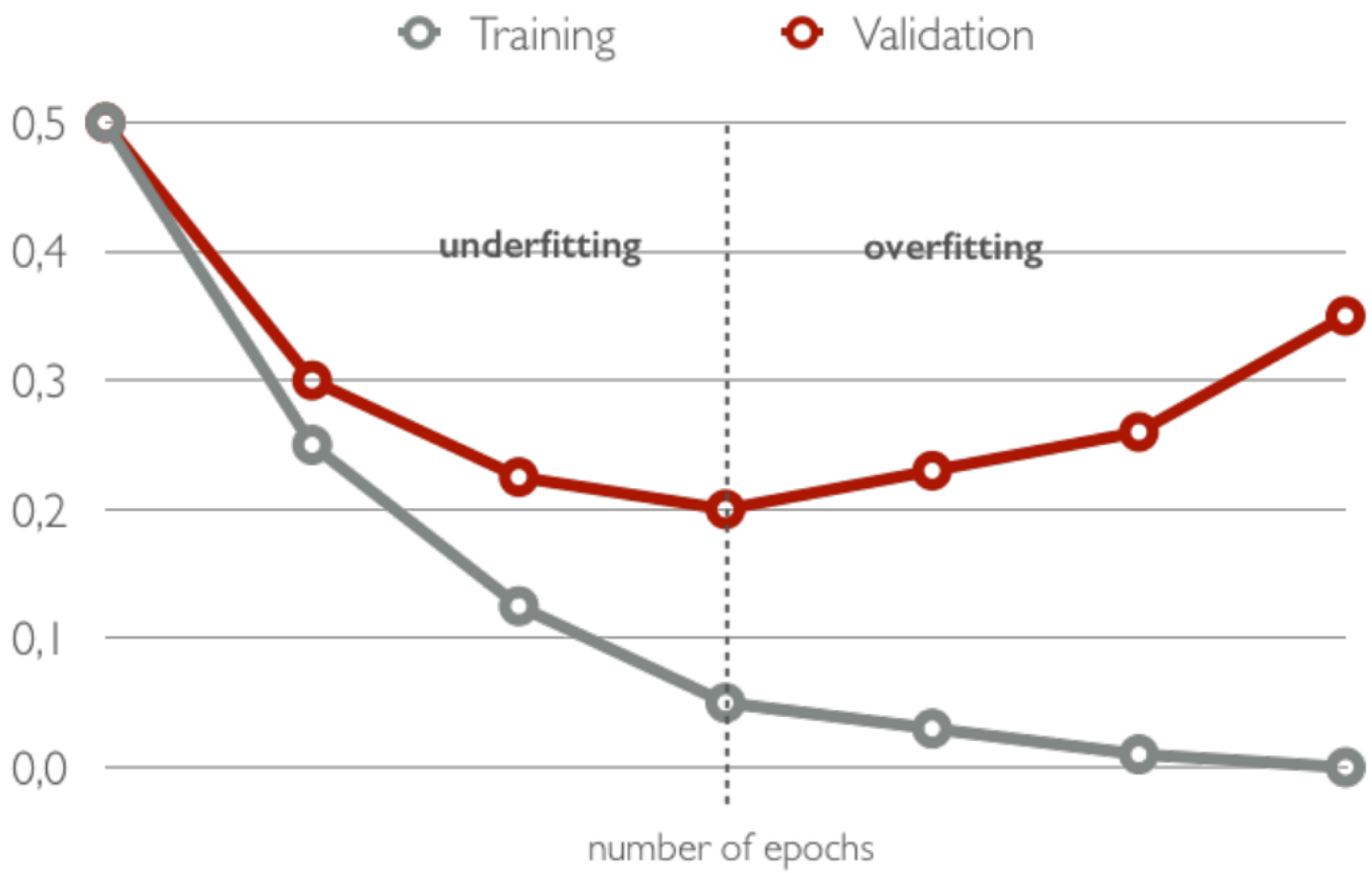
Overfitting Curve



Hold-out (Validation) Set

- Break the training set into two sets :
  1. The new training set (70-80% of the original training set)
  2. A "hold-out" set of data (the remaining)
- During training, in addition to measuring the objective function against the training data, **also measure against the hold-out set**
- Stop training when accuracy/performance starts deteriorating for **hold-out** (even if it may keep improving for the normal training data)

When to stop training?



Src: Russ Salakhutdinov https://www.cs.cmu.edu/~rsalakhu/10707/Lectures/Lecture_NN_Part2.pdf

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Eager learning** (the methods we've seen so far): Given a set of training data, constructs a classification model before receiving new (e.g., test) data to classify
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test data instance

Lazy vs. Eager Learning

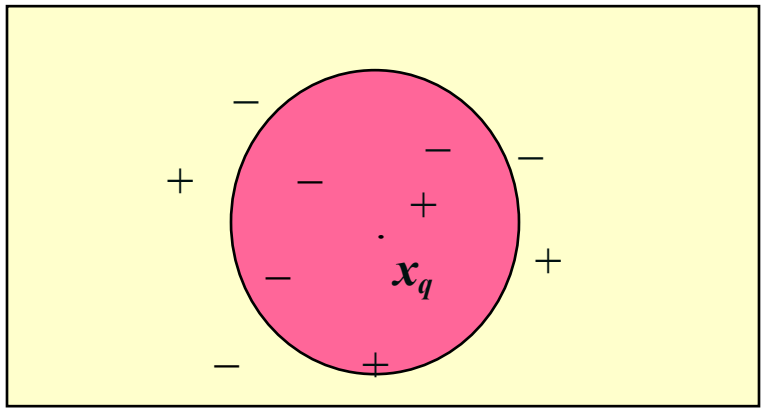
- Lazy:
  - less time in training
  - more time in predicting
- Accuracy comparison:
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Most typical approach:
  - *k*-nearest neighbors (k-NN) approach
    - Instances represented as points in a Euclidean space.

The *k*-NN Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor(s) are defined in terms of a distance measure
- Return the majority label of the k nearest neighbors



The vanilla *k*-NN Algorithm

- Given a data point x_q and #neighbors *k*
  1. Calculate $d_i = dist(x_q, x_i)$ for all x_i in training set
  2. Sort d_i
  3. Assign x_q label l_i as the majority label of the *k* nearest neighbors according to d_i

Confusion Matrix

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of instances in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Accuracy

- **Classifier Accuracy,** or recognition rate: percentage of test set instances that are correctly classified

  Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Class Imbalance

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Class Imbalance Problem**:
  - One class may be *rare*,
    - e.g. fraud
  - "98% accuracy" might be useless in this case!!!
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Precision and Recall

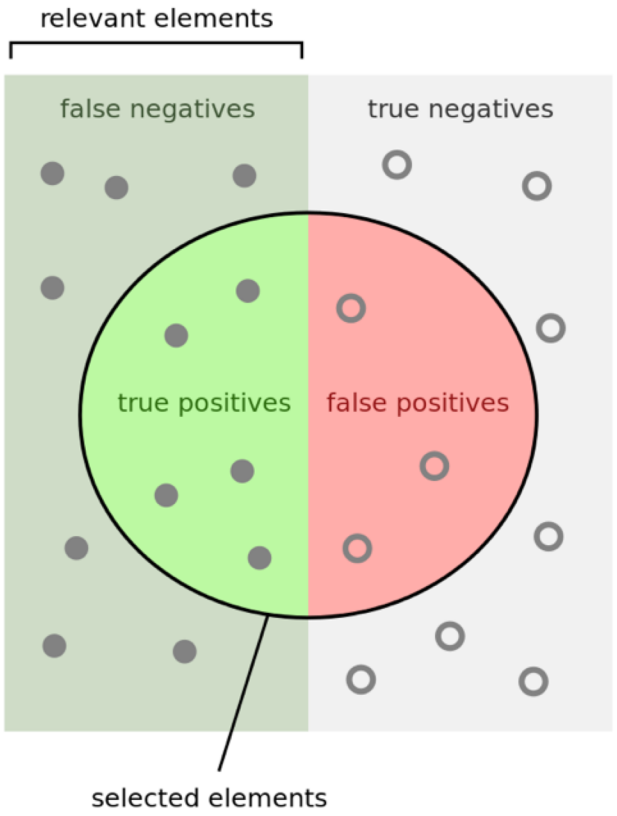
- **Precision**: exactness – what % of instances that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall (=sensitivity):** completeness – what % of positive instances did the classifier label as positive?

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

- Perfect score is 1.0
- Inverse relationship between precision & recall



How many selected items are relevant?

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

How many relevant items are selected?

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

Src: https://en.wikipedia.org/wiki/Precision_and_recall

Combining Precision & Recall

- Simple average may give wrong picture:
  - P = 0.5, R = 0.4, avg = 0.45
  - P = 0.7, R = 0.1, avg = 0.4
  - P = 0.02, R = 1, avg = 0.51
- Is that right??

Slide adapted from Andrew Ng

F-score

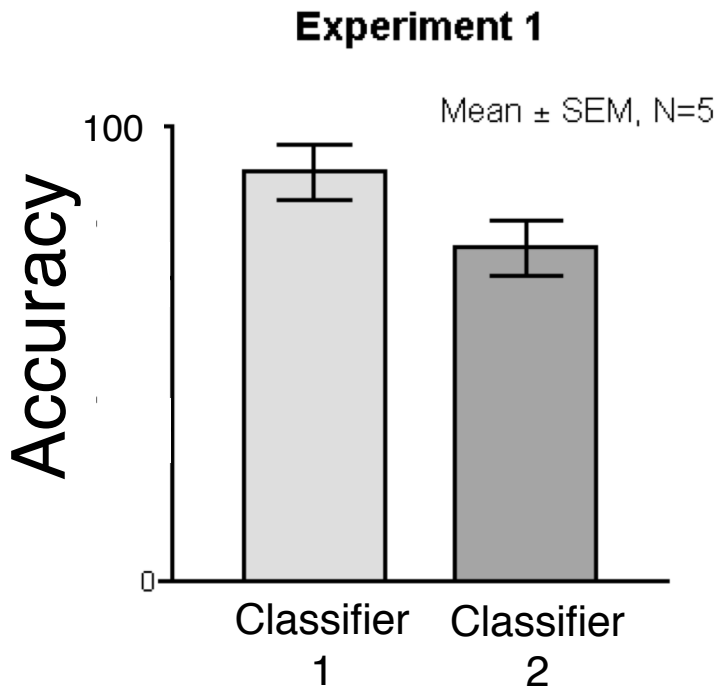
- ***F* measure (*F_1* or *F*-score)**:
  harmonic mean of precision and recall
  - In our example, F-scores are: **0.444**, 0.175, 0.039
  - Makes more sense in our example

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Cross-Validation Method

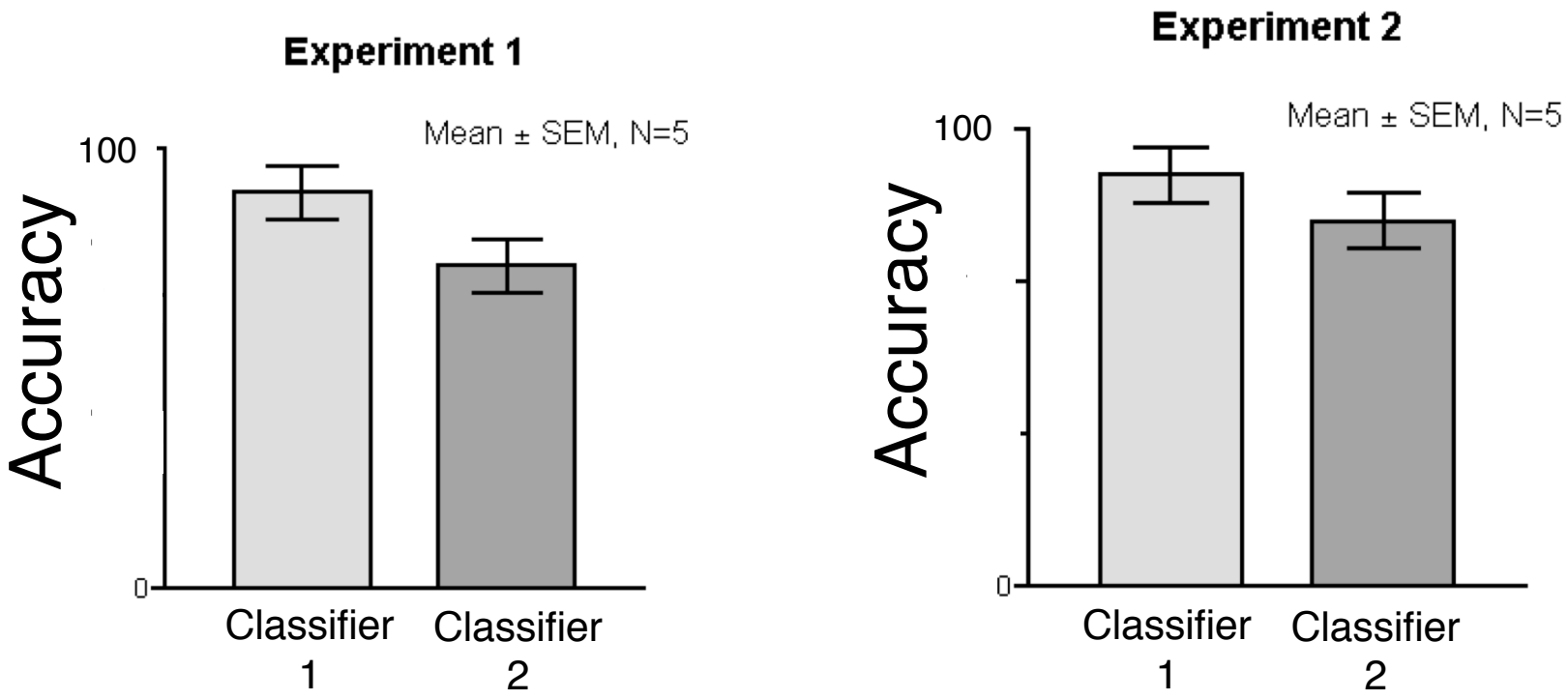
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Error-bars



We visualize mean +/- error:
SEM: Standard error of mean (σ/sqrt(n))
STD: Standard deviation
Confidence intervals: e.g., 95% interval

Error-bars



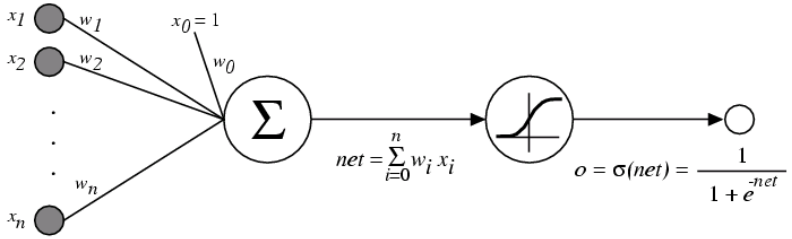
When there is overlap, this *may* point to problems in significance

Img src: https://egret.psychol.cam.ac.uk/statistics/local_copies_of_sources_Cardinal_and_Aitken_ANOVA/errorbars.htm

Outline

- Data Visualization & Manipulation
- Basic Supervised Concepts
- Advanced Supervised Concepts
- Parting Thoughts

The Perceptron



Frank Rosenblatt

$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Training data are in form (**x**,y_j) (value, label)

1. Initialize weights **w** (zero or small value)
2. For each training data point **x**
   1. Compute $y_{pred} = sigmoid(\sum_{i=1}^{n} w_i x_i)$
   2. Compute error = ($y_j - y_{pred}$)
   3. "Correct" the weights: $w_i \leftarrow w_i$ + error* x_i

Gradient Descent Update Rule

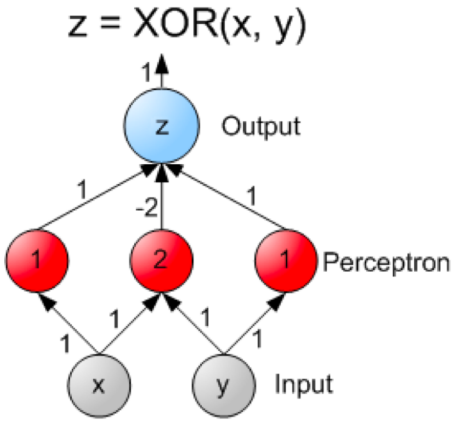
$$\theta_j := \theta_j + \alpha \underbrace{\left(y^{(i)} - h_\theta(x^{(i)})\right)}_{\text{error}} x_j^{(i)}.$$

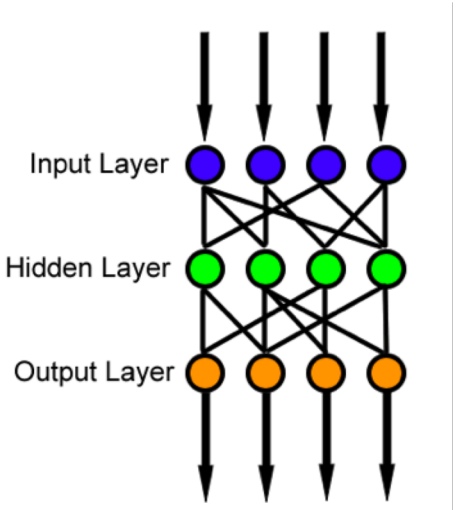
- LMS or Widrow-Hoff learning rule
- Quite intuitive:
  - Magnitude of the update is proportional to the error term
  - If we have small error, there is need for very little correction and vice versa

Adapted from Andrew Ng's Notes

How can we express general non-linear functions?

- Perceptron still defines a linear decision surface
  - Can only separate data that are linearly separable
- Solution:
  - Add more Perceptron units
    - Feed-Forward Neural Network
  - Add more layers
    - Multi-layer Feed-Forward Neural Network or Multilayer Perceptron (MLP)





Img source: https://en.wikipedia.org/wiki/Feedforward_neural_network

Multilayer Perceptron (MLP)

- The **inputs** to the network correspond to the attributes measured for each training vector
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- Weighted outputs are the **output layer**, which emits the prediction

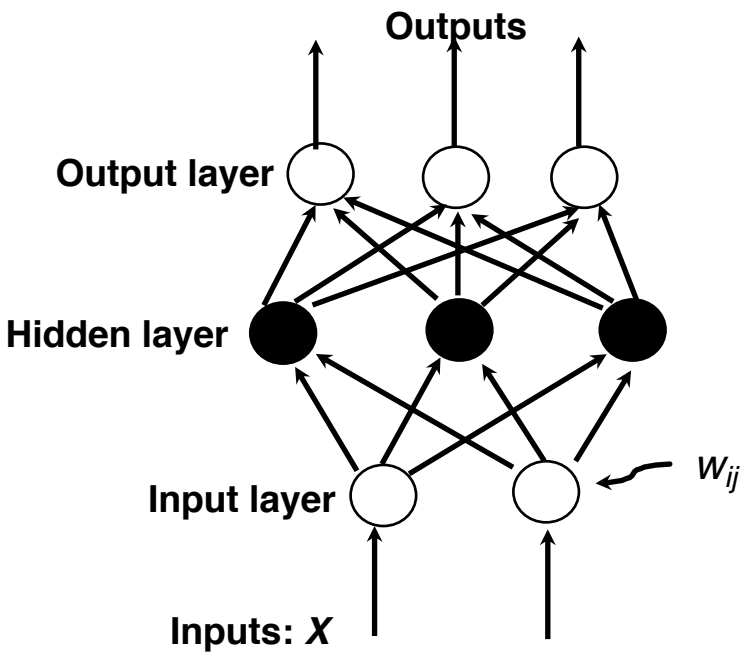
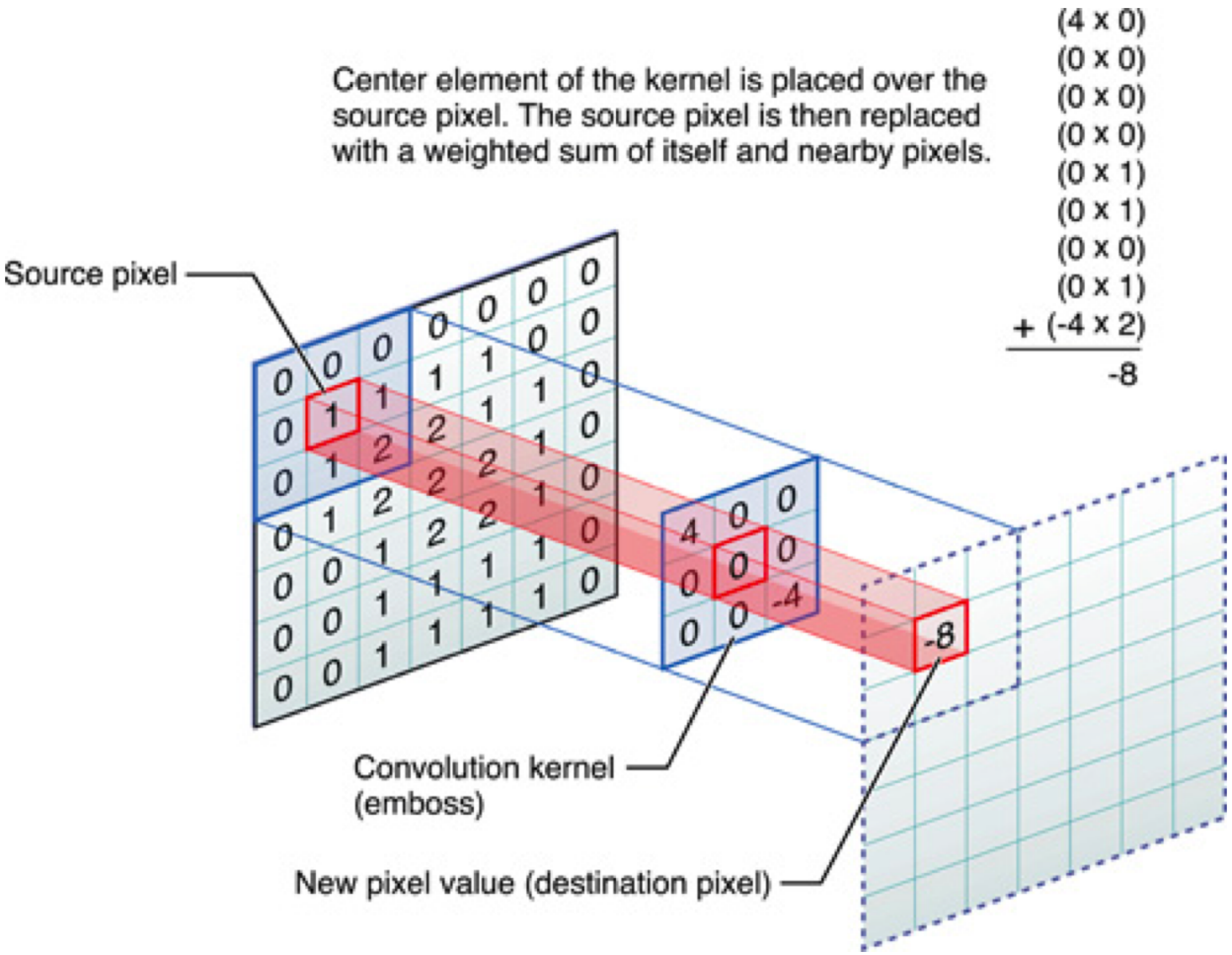


Image Convolution



Source: https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Examples of Convolution

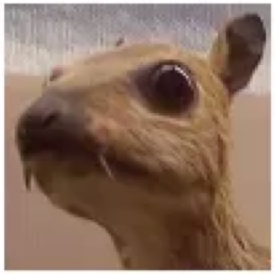
Averaging (Blurring)

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Src: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Edge Detection

Input image



Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

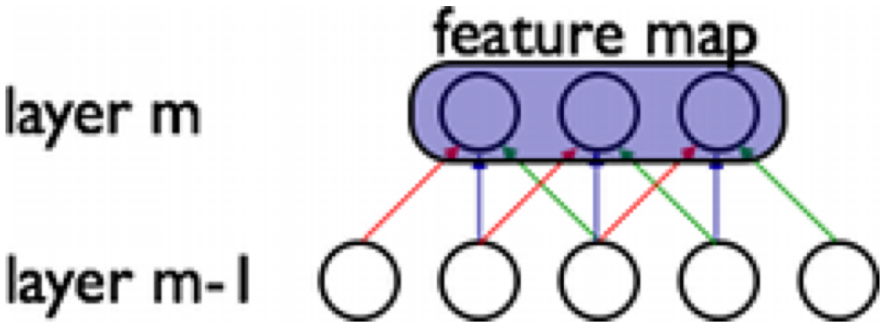
Feature map

Src: http://timdettmers.com/2015/03/26/convolution-deep-learning/

Why conv filters?

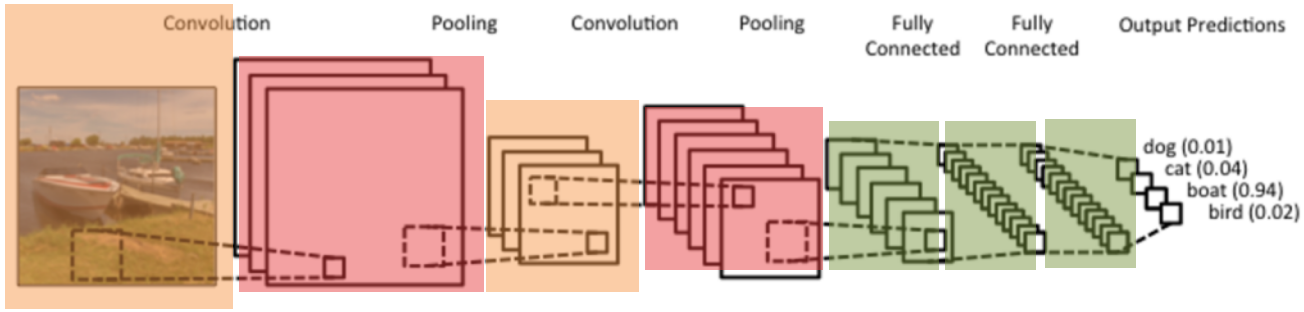
- This is a way of reducing #parameters:
  - Instead of having a single weight for every pixel, each pixel is written as a linear combination of its value and the conv filter weights
  - This leads to more efficient training than having a gigantic fully-connected set of layers
- Convolution is also natural for image-related tasks

One-dimensional example



http://deeplearning.net/tutorial/lenet.html

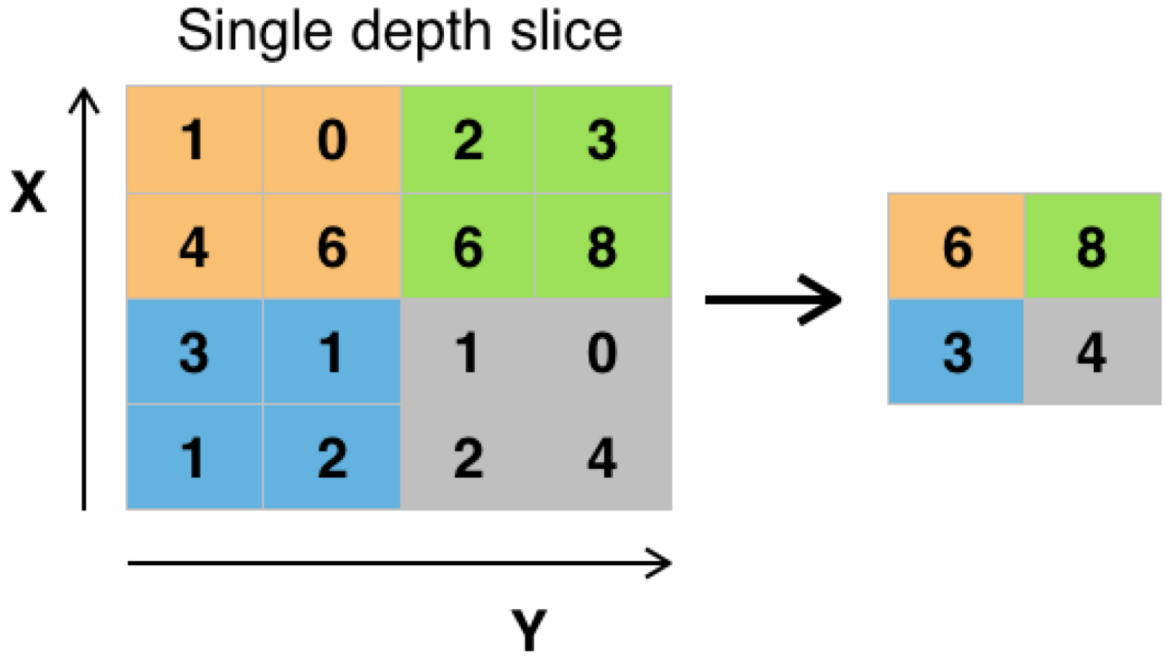
Deep CNN blocks



- **Convolution:** choose a conv. kernel with particular size, zero padding.
- **Pooling:** Non-linear down sampling (reduce the size of data between layers). Helps with overfitting.
- **Rectified Linear Units (ReLU):**
  - Applies f(x) = max(x,0) (can also use sigmoid or tanh)
  - Increases non-linearity of output function
- **Fully Connected Layer:** Feed-forward Muliti-Layer (as we saw before)

Img Source: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Max Pooling



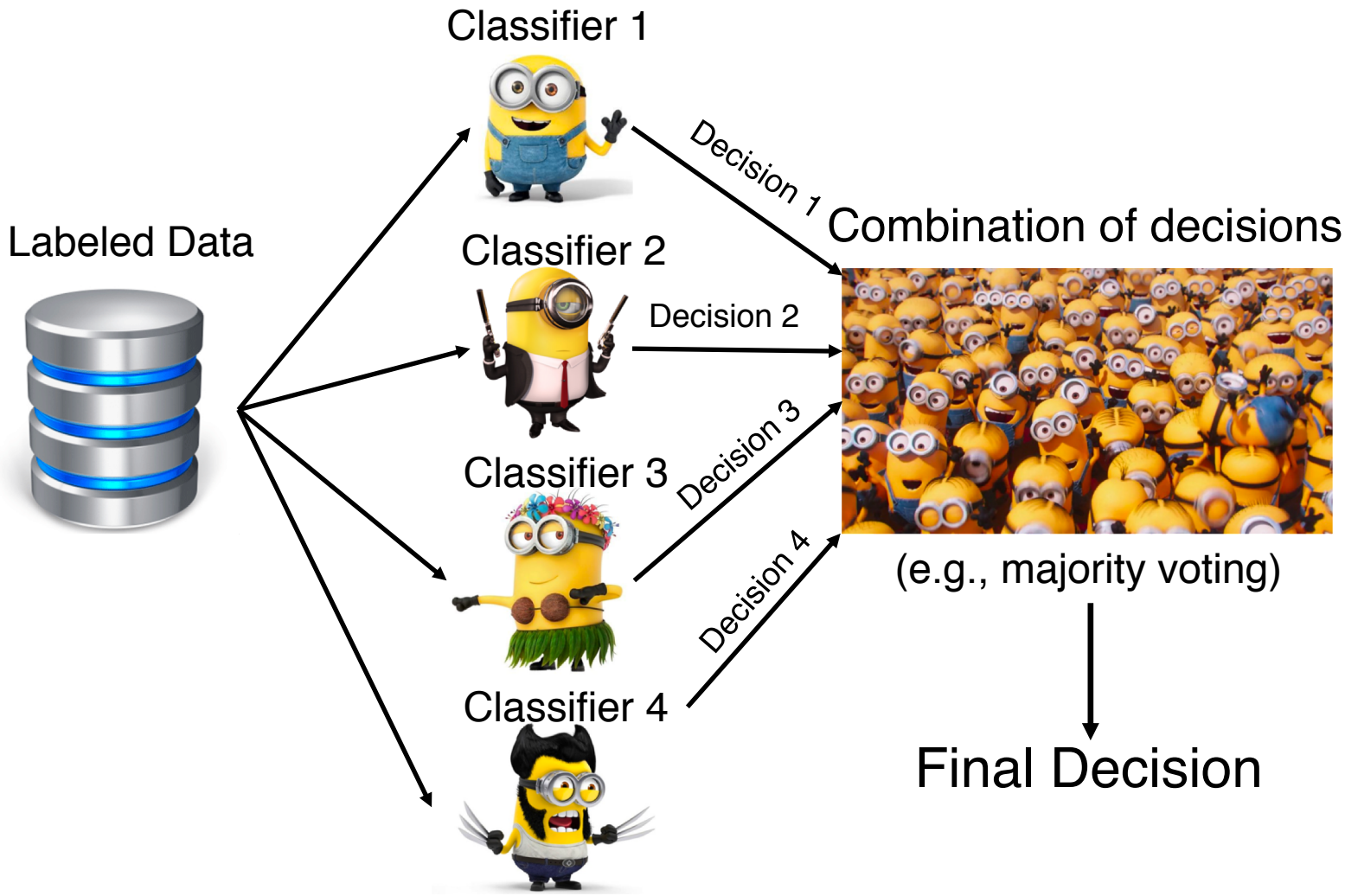
Src: https://en.wikipedia.org/wiki/Convolutional_neural_network

Ensemble Methods

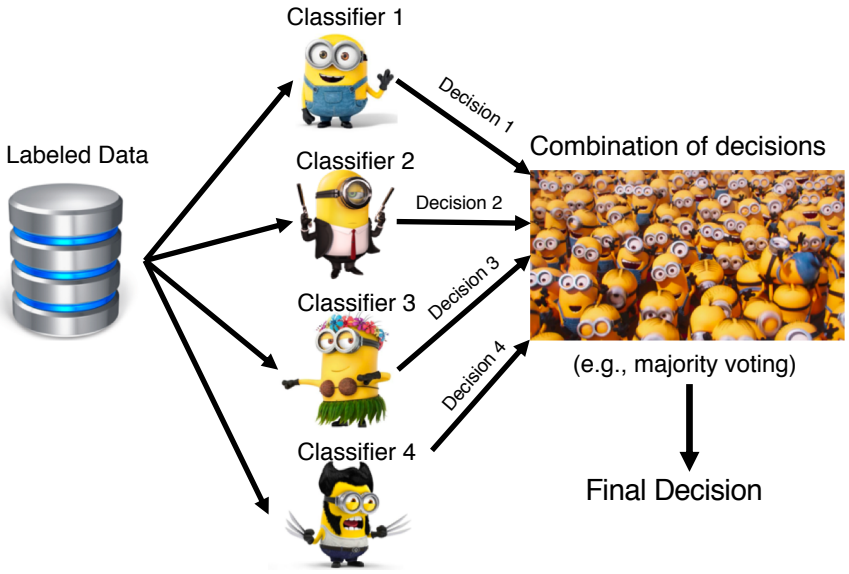
Can a collection of weak learners make a strong learner?

Weak learner: works better than random, but not great
Strong learner: correlates very well with the labels of a task

Ensemble Methods



Ensemble Methods: Increasing the Accuracy



- Popular ensemble methods
  - **Bagging**: averaging the prediction over a collection of classifiers
  - **Boosting**: weighted vote with a collection of classifiers
  - **Random Forest**

Bagging: Boostrap Aggregation

- Majority vote based
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample X
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to X

Boosting

- Use weighted majority vote of different classifiers, adapt the weights according to their performance
- How boosting works:
  - Assign a weight to each training point
  - Learn k classifiers iteratively
  - After learning classifier model M_i update the weights so that M_{i+1}, **pays more attention to missclassified points** by M_i
  - **Combines the votes** to **M*** of each individual classifier using weights that reflect their accuracy
- Popular example: **Adaboost** (Adaptive Boost)

Random Forest

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier that is generated with a randomness element:
    - On a random bootstrap sample of the training data
    - On a random selection of features
  - During classification, each tree votes and the most popular class is returned
- Comparable in accuracy to other state-of-the-art ensemble methods, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and generally faster than other methods

Out-of-bag error

- One way of measuring performance
- Mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample

Random Forest for Feature Selection

- Can use Random Forest to rank features according to **importance**.
- For the j-th feature create two versions of the training data:
  - Original version
  - Random permutation where values of the j-th feature are "scrambled" in a meaningless way
- Measure the difference in the error (aka out-of-bag error) between original and permuted data
  - This gives a notion of importance
- Described in original paper: Breiman "Random Forests", 2001 https://www.stat.berkeley.edu/users/breiman/randomforest2001.pdf

Outline

- Data Visualization & Manipulation
- Basic Supervised Concepts
- Advanced Supervised Concepts
- Parting Thoughts

Class Imbalance

- Class-imbalance problem:
  - Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- We saw how we measure classifier performance in this case, but there is another danger:
  - ***The classifier does not have enough examples to learn the minority class properly!***

Class Imbalance

- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Penalize:** penalize the algorithm more when it makes a mistake on the minority class
- Still difficult for class imbalance problem on multiclass tasks

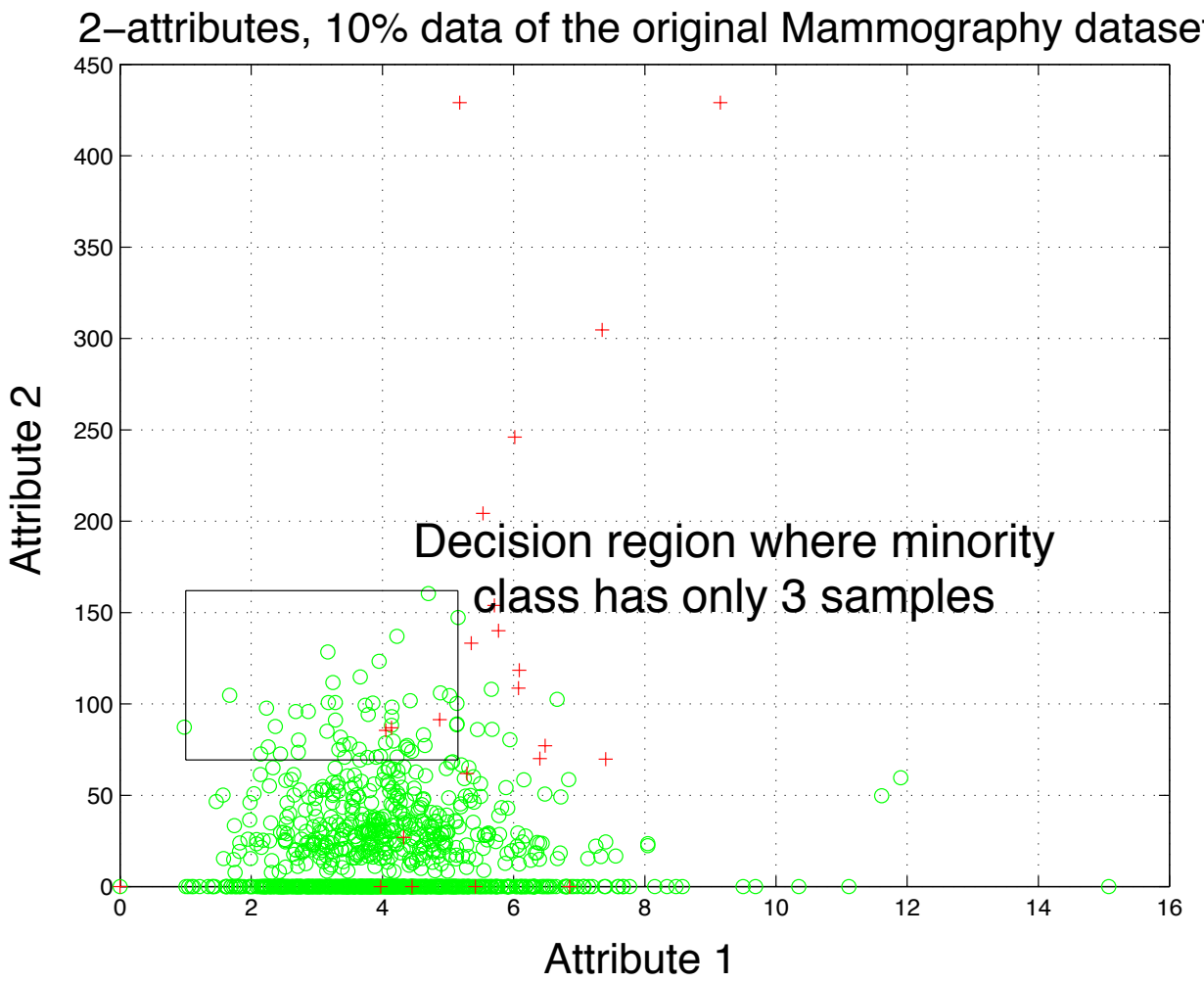
Examples of Oversampling

- SMOTE: Synthetic Minority Oversampling Technique
- MixUp
- Generative Adversarial Networks (GANs)

SMOTE: Synthetic Minority Oversampling Technique

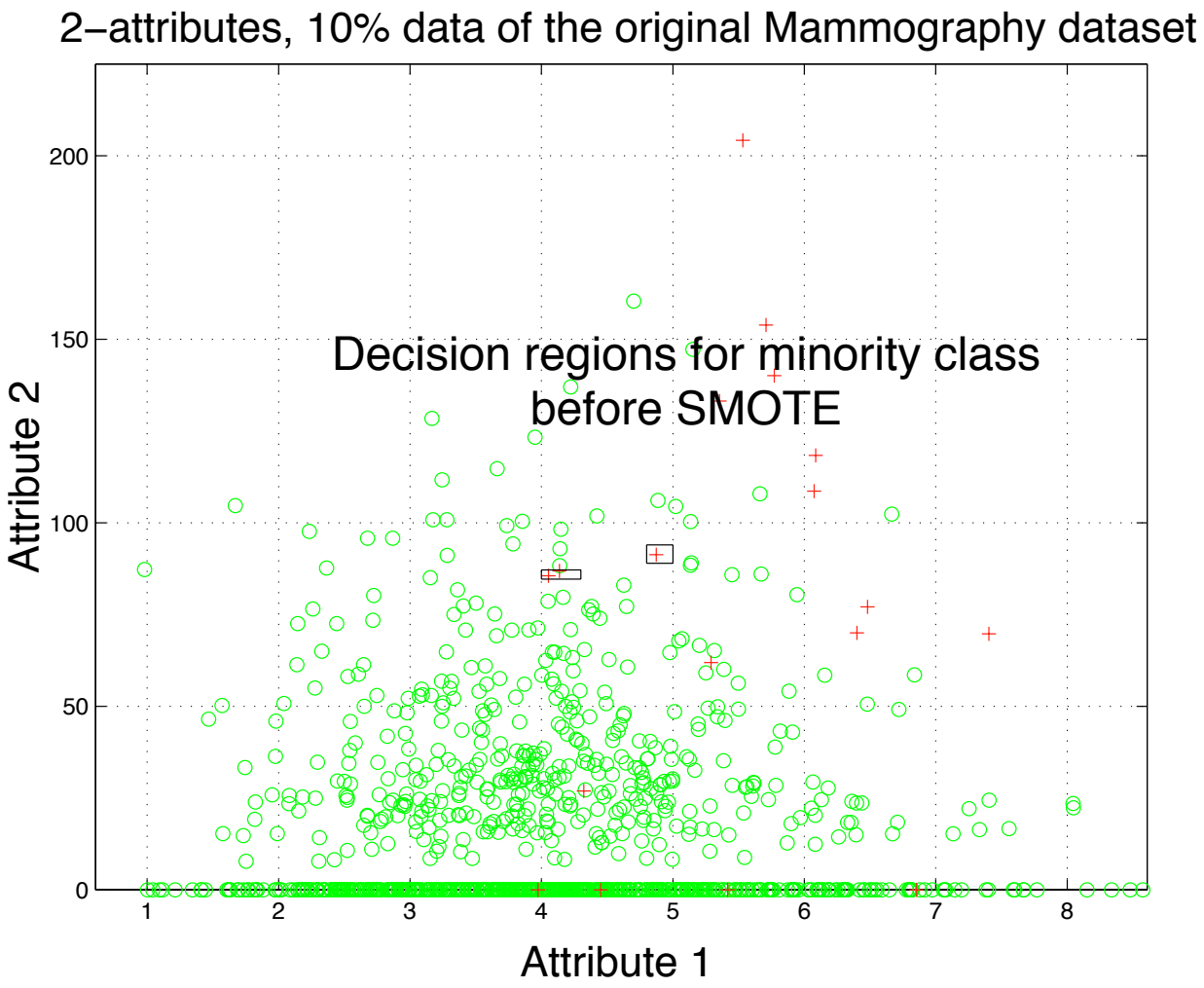
- Chawla et al. "SMOTE: Synthetic Minority Oversampling Technique", JAIRC 2002
- **Key idea**: Oversample minority class by generating synthetic data points
  - **Step 1**: Find *k* nearest neighbors of training examples of minority class
  - **Step 2**: Generate synthetic data points along the lines that connect those neighbors (with some random noise)

SMOTE Example



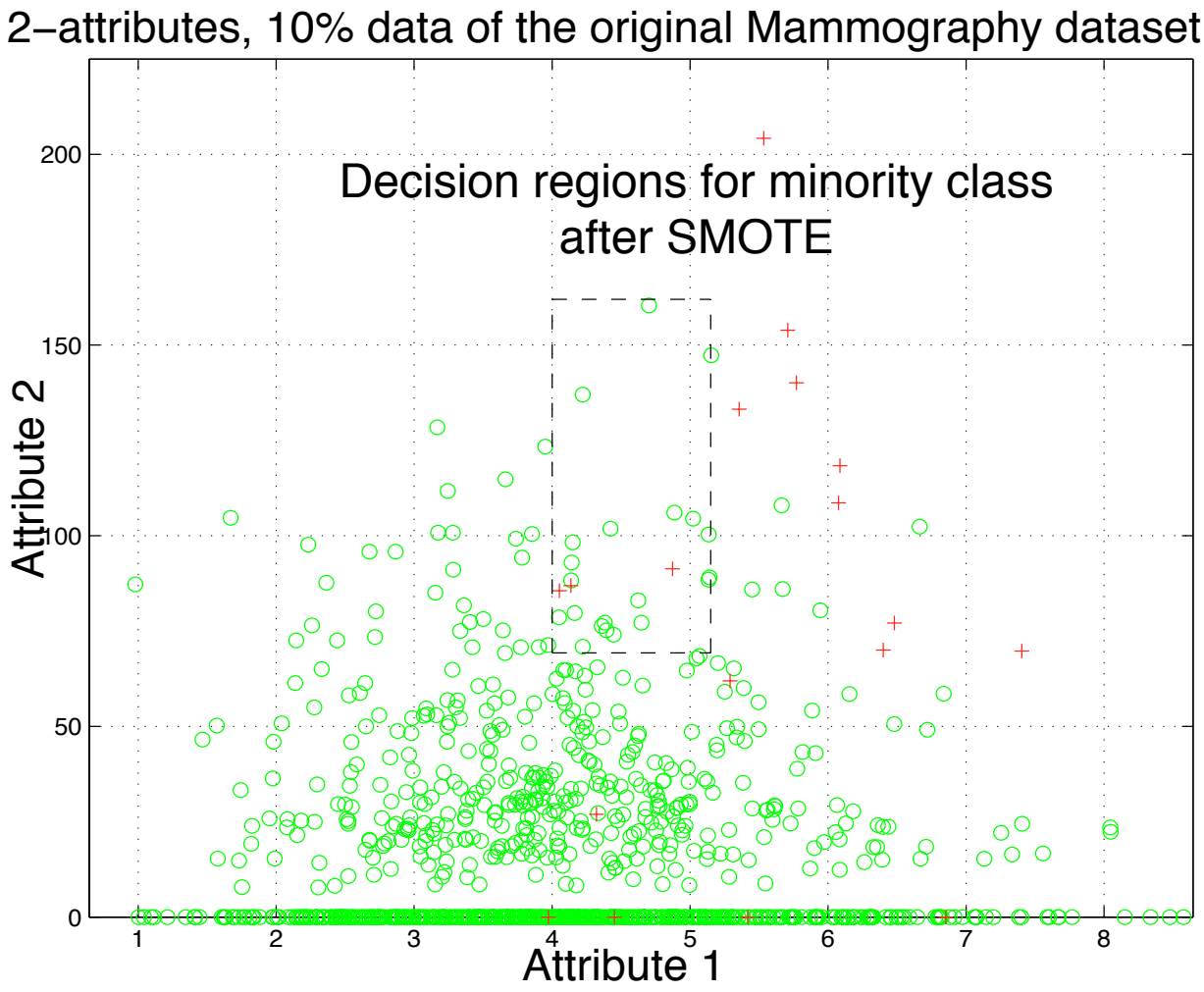
Img taken from SMOTE paper

SMOTE Example



Img taken from SMOTE paper

SMOTE Example



Img taken from SMOTE paper

MixUp

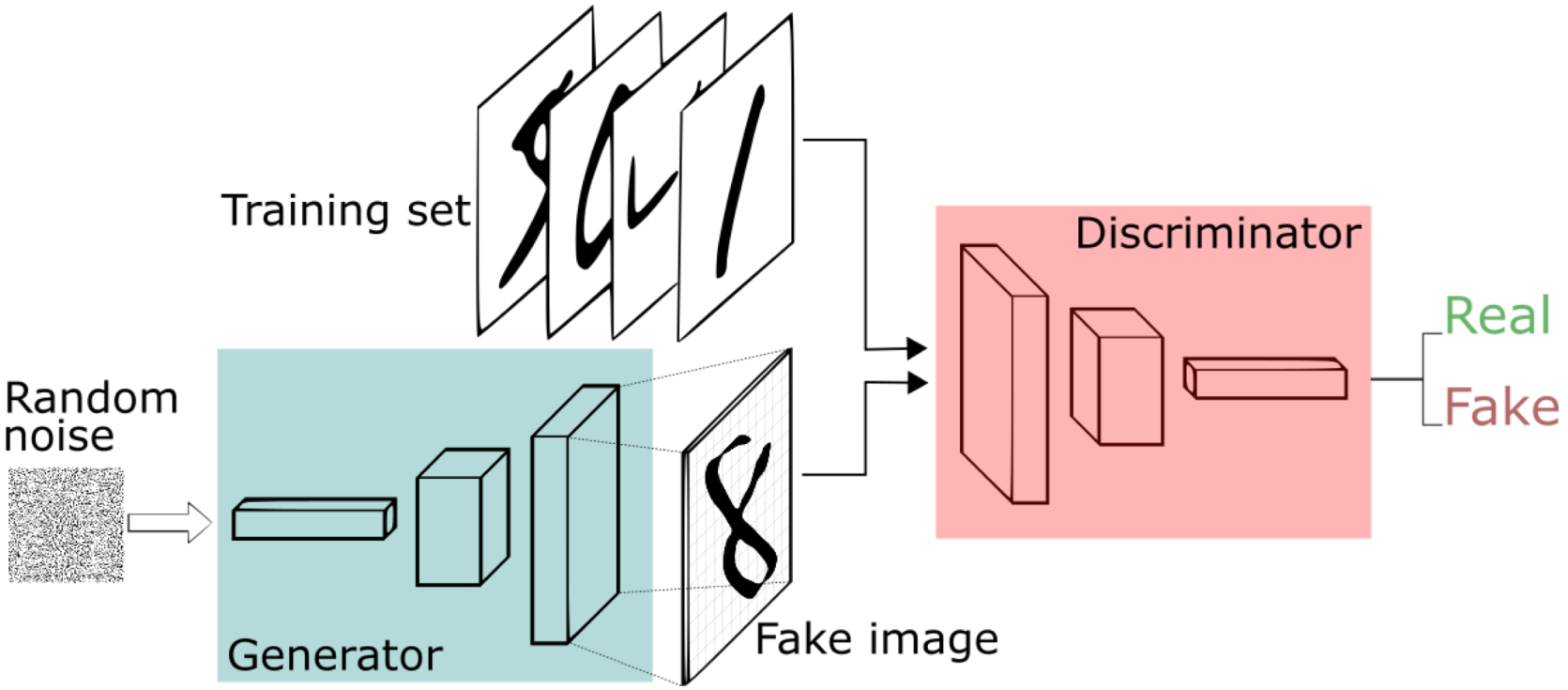
Contribution Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed *mixup* (Section 2). In a nutshell, *mixup* constructs virtual training examples

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$. Therefore, *mixup* extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. *mixup* can be implemented in a few lines of code, and introduces minimal computation overhead.

Zhang et al. mixup: BEYOND EMPIRICAL RISK MINIMIZATION, ICLR 2018
https://arxiv.org/pdf/1710.09412.pdf

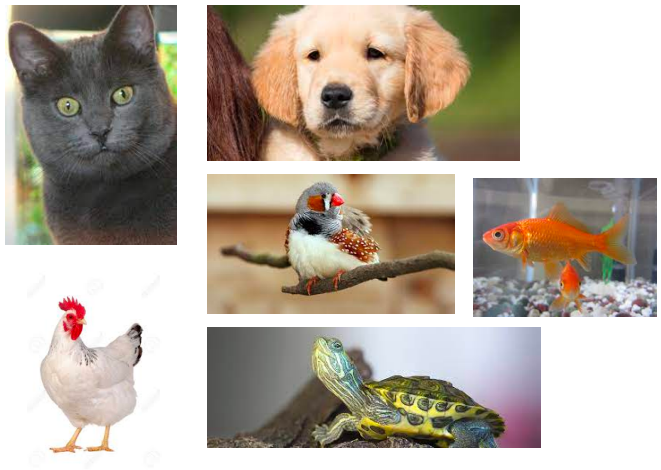
Generative Adversarial Networks (GANs)



Img credit: https://deeplearning4j.org/generative-adversarial-network

Transfer Learning Example

Task 1:
Domesticated/pet animals



We have lots of labeled data points
Our classifier can be trained to work very well in this task.

Task 2:
Wild animals

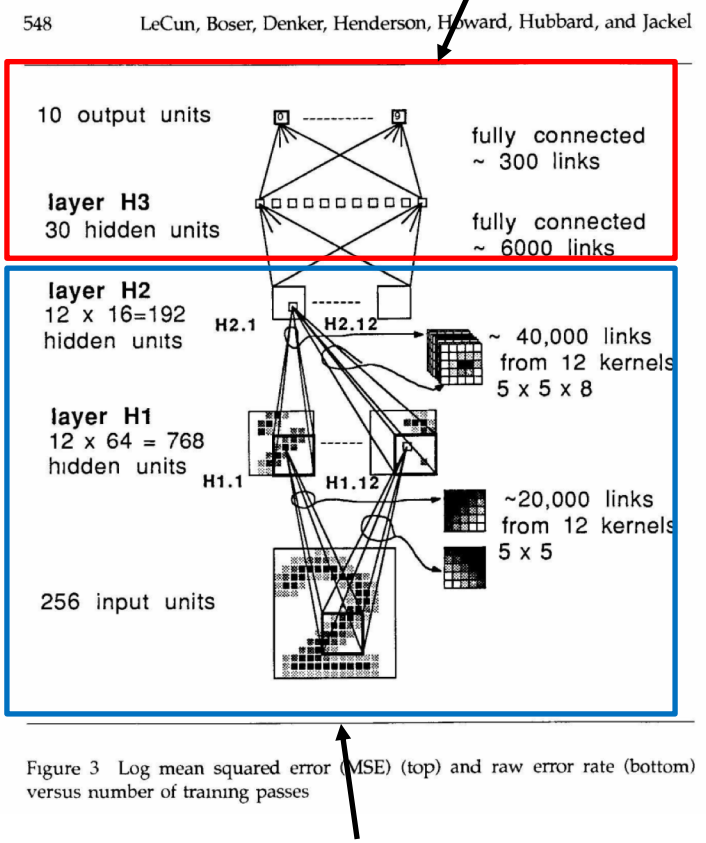
Labeled data points are hard to get
May not have enough to train a good classifier.

Can we borrow/repurpose some information from Task 1?

Deep Transfer Learning

- Use an existing deep network trained on e.g., ImageNet
- Substitute the "classifier" part of the network (the last fully connected set of layers)
- Add a new "classifier" FC layer
- Re-train
  - FC layer
  - OR the whole network
- using examples from the new problem

Replace classification layer



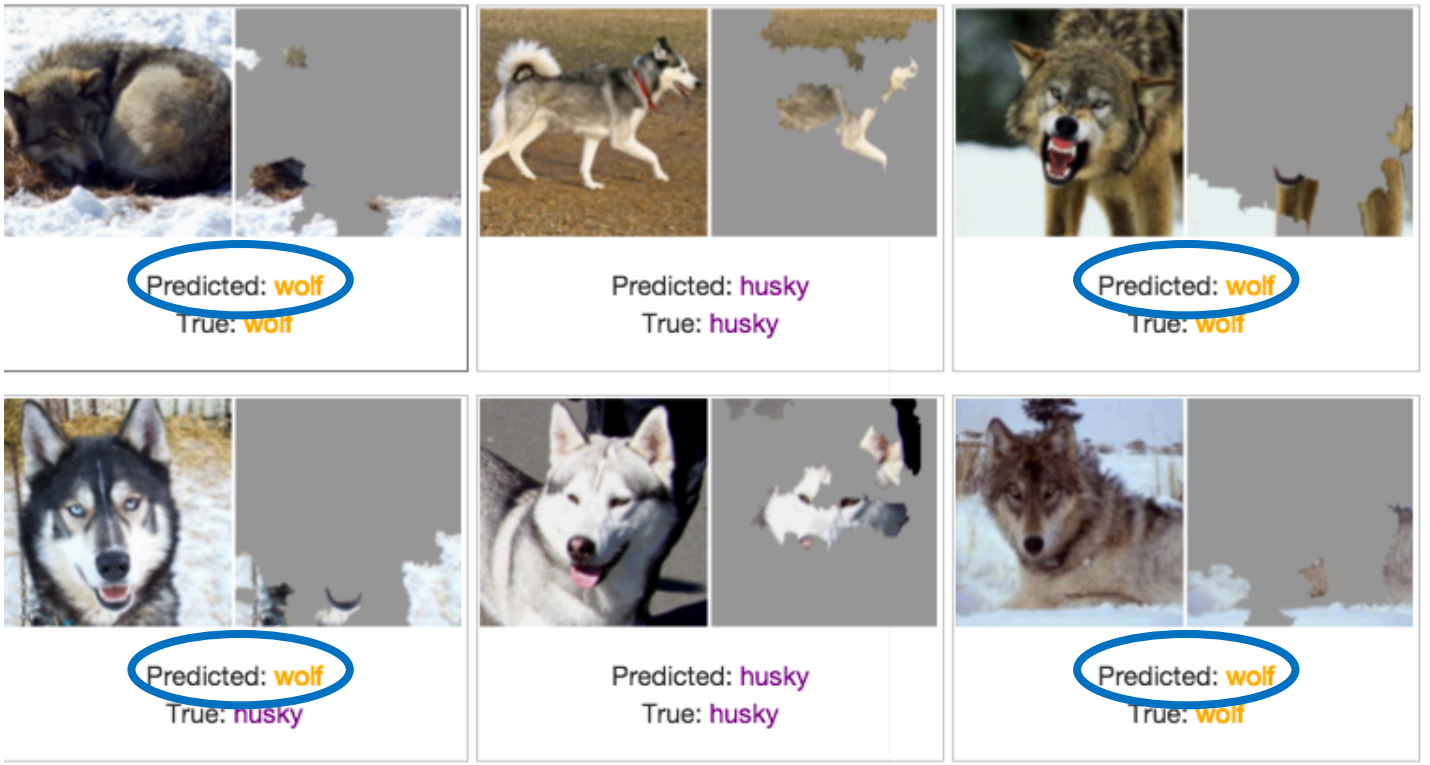
This has learned a nice set of features

Wolf or Husky?

Only 1 mistake!

Slide by Sameer Singh
https://filene.org/assets/images-layout/Panel_Singh.pdf

Wolf or Husky?



We've built a great snow detector…

Slide by Sameer Singh
https://filene.org/assets/images-layout/Panel_Singh.pdf