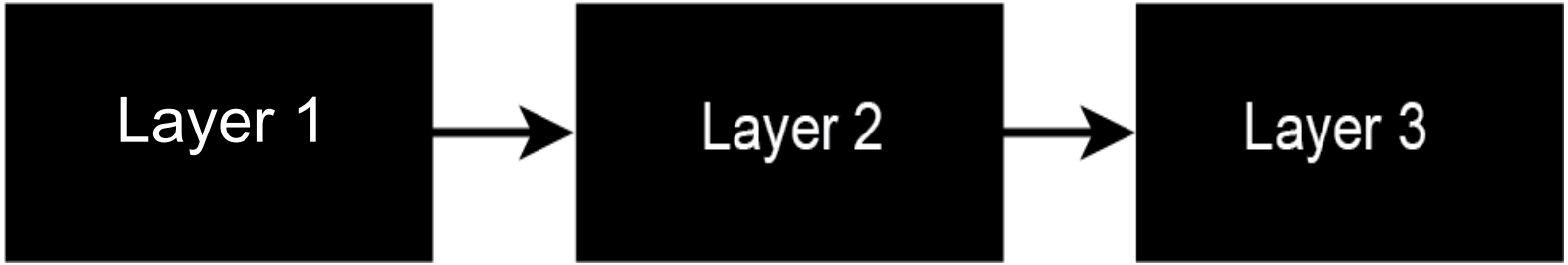
Master Lecture 4

ML Principles, Data Visualization

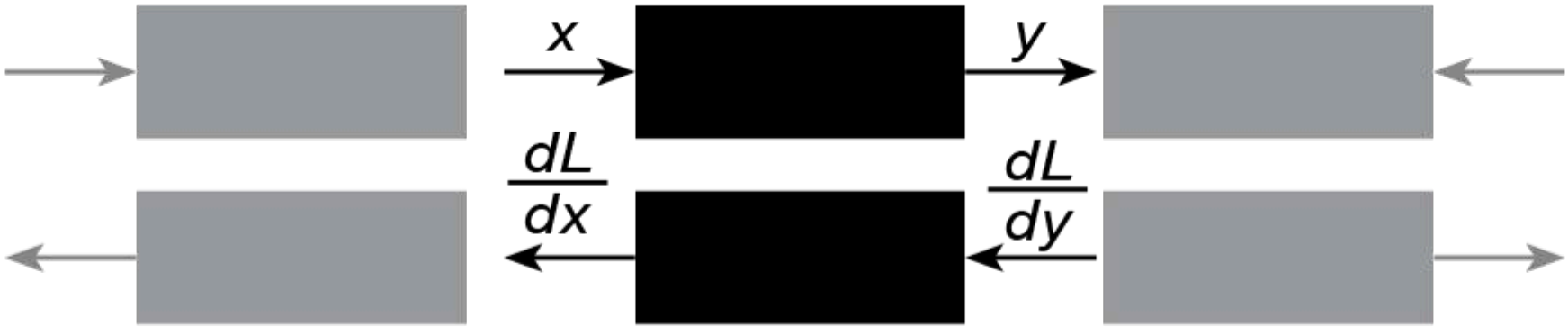
MLP and BP

Brief Review

Neural Network: Abstract View



Chain Rule



We know $\frac{\mathrm{d}y}{\mathrm{d}x}$ and $\frac{\mathrm{d}y}{\mathrm{d}W}$

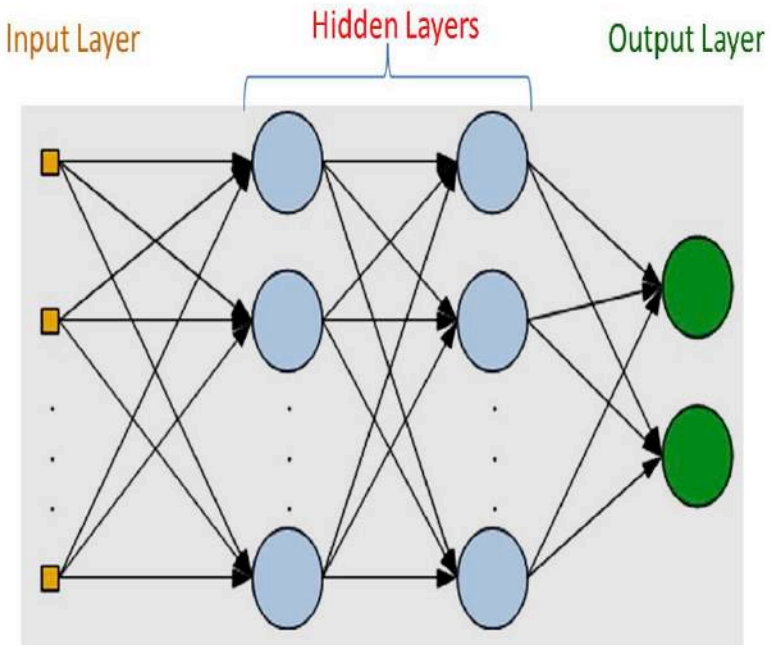
Back Propagation for MLP

Two Computational Blocks/ Steps

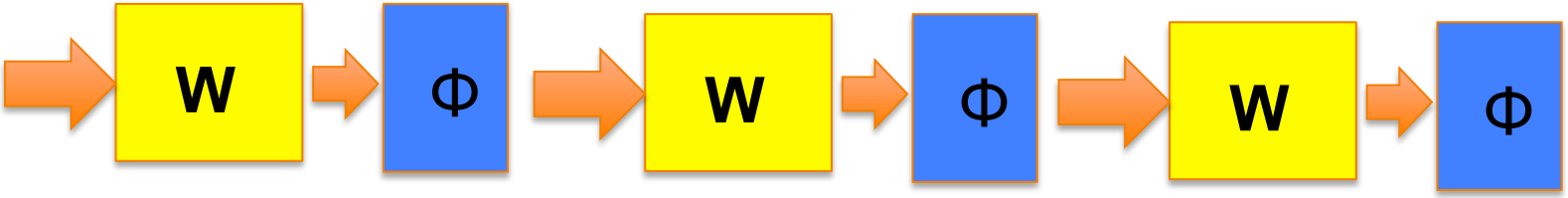
$$y = Wx$$

$$y = \phi(x) = \frac{1}{1 + e^{-x}}$$

In either case we can compute $\frac{dy}{dx}$ easily.



A simpler view point



Loss

Blocks with Learnable parameters Matrix Multiplication

Nonlinear functions (often non learnable)

Backpropagation

$$\frac{\mathrm{d}L}{\mathrm{d}x} = \frac{\mathrm{d}L}{\mathrm{d}y} \cdot \frac{\mathrm{d}y}{\mathrm{d}x} \quad (1) \qquad \frac{\mathrm{d}L}{\mathrm{d}W} = \frac{\mathrm{d}L}{\mathrm{d}y} \cdot \frac{\mathrm{d}y}{\mathrm{d}W} \quad (2) \qquad W^{n+1} = W^{n} - \eta \frac{dL}{dW} \quad (3)$$

Backpropagation

Let there be N stages. For a computational block *l*,

1. Compute $\frac{\mathrm{d}L}{\mathrm{d}x}$ using equation **1**.
2. If the block as a learnable parameter W, then
   - Compute $\frac{\mathrm{d}L}{\mathrm{d}W}$ using equation **2**.
   - Update the parameters using equation **3**.
3. Set the $\frac{\mathrm{d}L}{\mathrm{d}x}$ of stage l as $\frac{\mathrm{d}L}{\mathrm{d}y}$ of stage l – 1, and repeat the steps 1-3, until we reach first block.

The Machine Learning Framework

$$y = f(x)$$

Output **prediction function** **feature representation**

Training: given a training set, estimate the prediction functior $f()$ by minimizing the prediction error

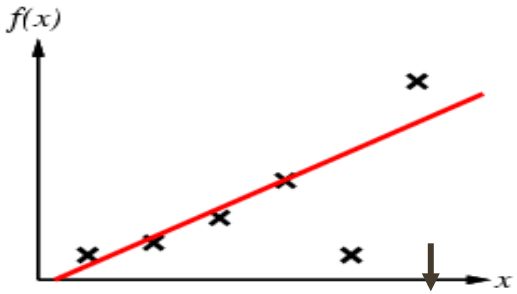
Testing: apply $f()$ to unknown test sample x and predicted value(output) is y

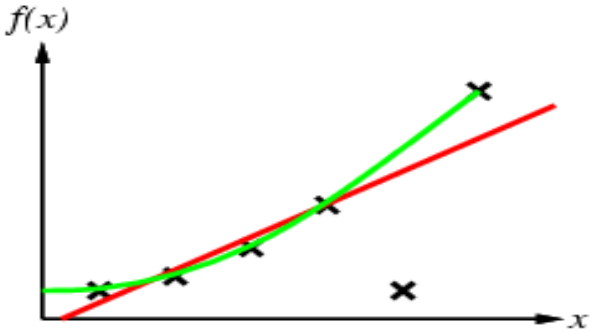
Principles of ML

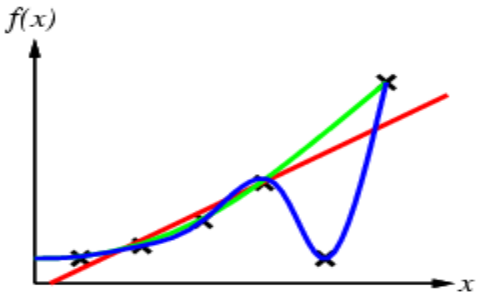
What happens when we “train”?

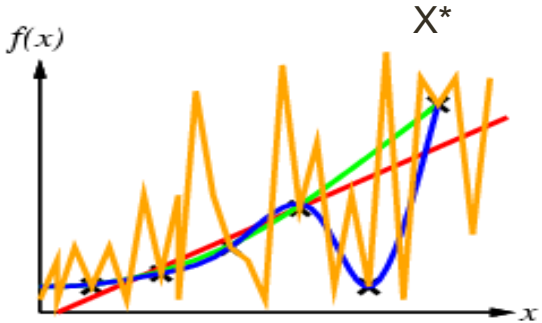
- “The problem of training is to find the best W (parameters, coefficients) that minimize the loss/error computed on the training samples”

What is the best model?





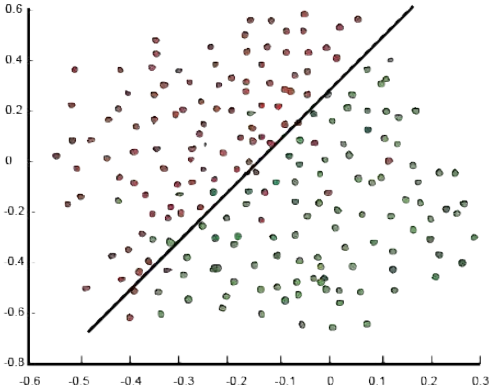


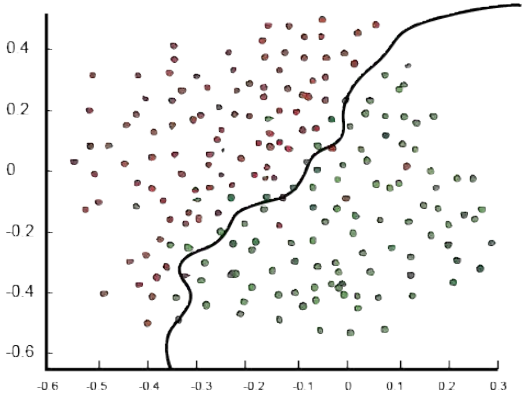


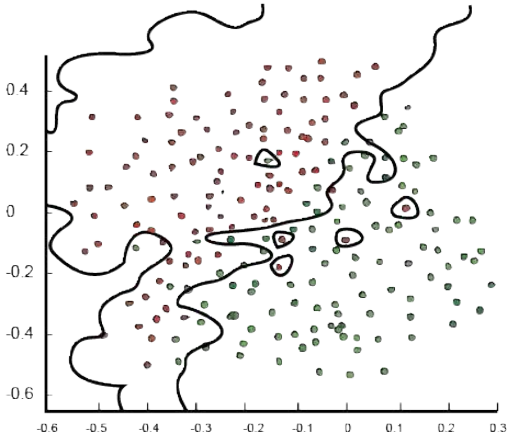
Given six "training" (x_i, y_i) pairs, find the y corresponding to the new "test" x*

Which curve is the best?

What is the best classifier?







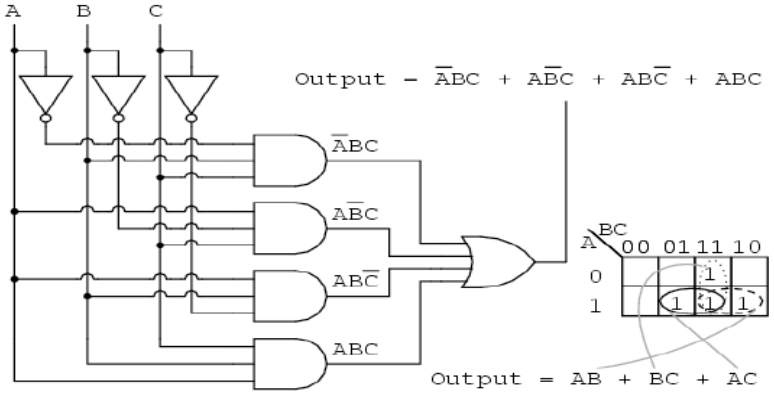
What happens when we simply learn?

- We often minimize an error over the training examples.
- The discrepancy between the actual and predicted. Let x_i is the input, z_i is the true output and y_i is the predicted output. f() is the model, say a Neural Network.
  - Eg.

$$\sum_i (z_i - y_i)^2 = \sum_i (z_i - f(x_i))^2$$

- If we want only to minimize this, we can even get zero error. The perfect error.
- But we do not want that.**Then what do we want?**

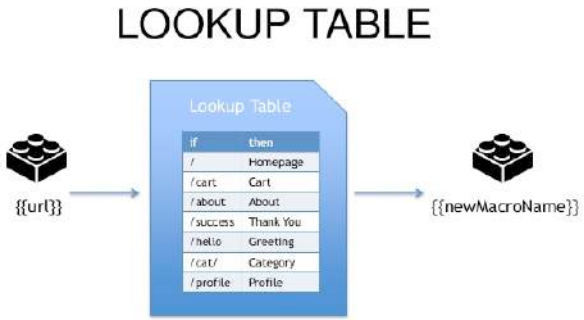
We know about f() that fits samples perfectly

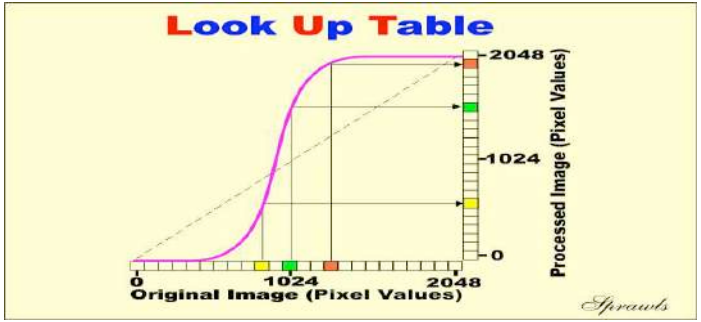


$$f(a,b,c) = \underset{111}{ABC} + \underset{110}{AB\overline{C}} + \underset{011}{\overline{A}BC} + \underset{001}{\overline{A}\overline{B}C}$$

C \ AB	00	01	11	10
0	0	2	1 6	4
1	1 1	1 3	1 7	5

Is K-Map Learning?





Is LUT Learning?

Learning

Learning is concerned with accurate prediction of future data, *not* accurate prediction of training or available data.

Memorization and generalization

- Memorization:
  - We are concerned with performance on our limited data.
- Generalization
  - Refers to the capability of applying learned knowledge to previously unseen data
  - Without generalization there is no learning, just memorizing!

Questions?

We only have “training data”. Then how do we minimize error on unseen/future data?

Occam's Razor

Select the *simplest* hypothesis (solution) that *suits/fits* the data.

Eg.Minimize Sum of "fit error" and "degree of the polynomial"

Model Complexity: Examples

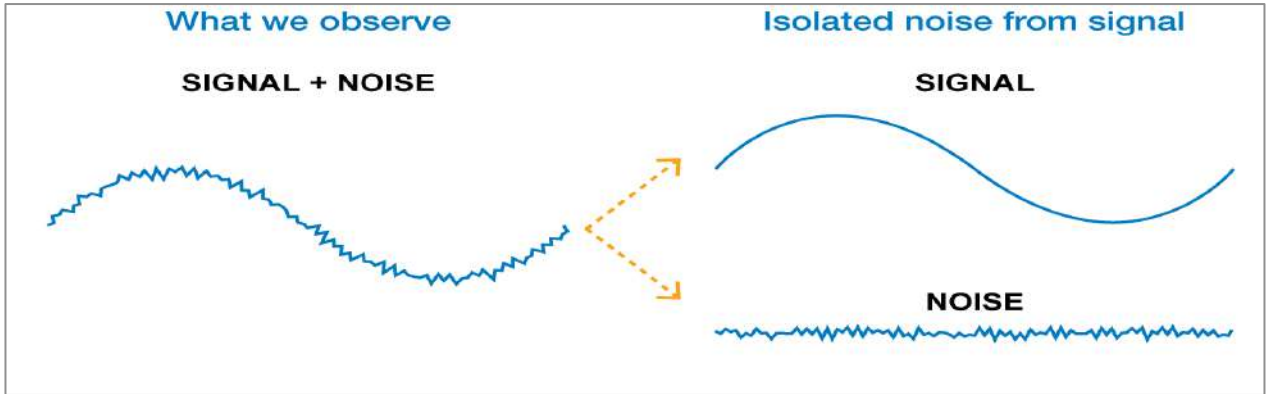
- This is the complexity of our solution
- Neural Networks: (i) Number of Neurons (ii) Number of Weights (iii) Number of Learnable parameters
- Decision Tree: (i) Depth of the tree.

What's in that we learn from

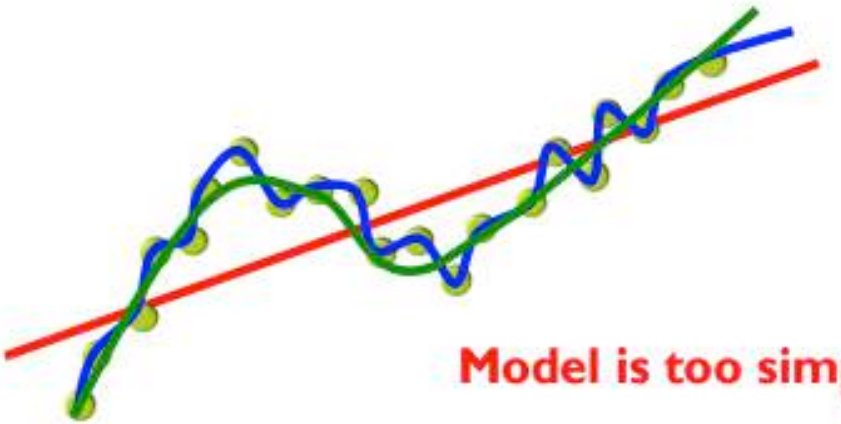
The data usually has two parts:

- Signal: information that is relevant to pattern detection or prediction
- Noise: information that is irrelevant **to the future**

Finding which is which is the challenge



Model the Signal; Not Noise

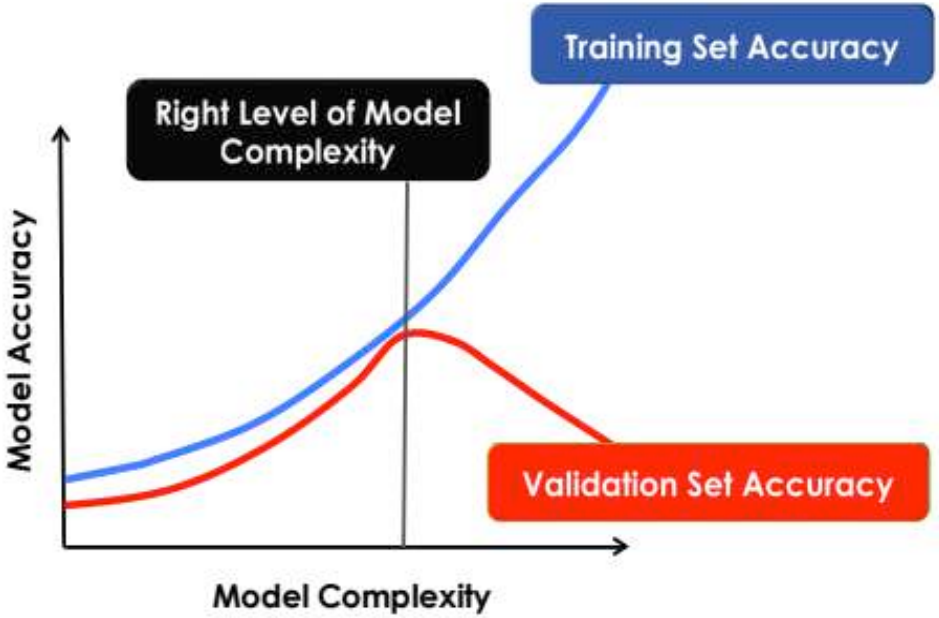


Model is too simple UNDER LEARN

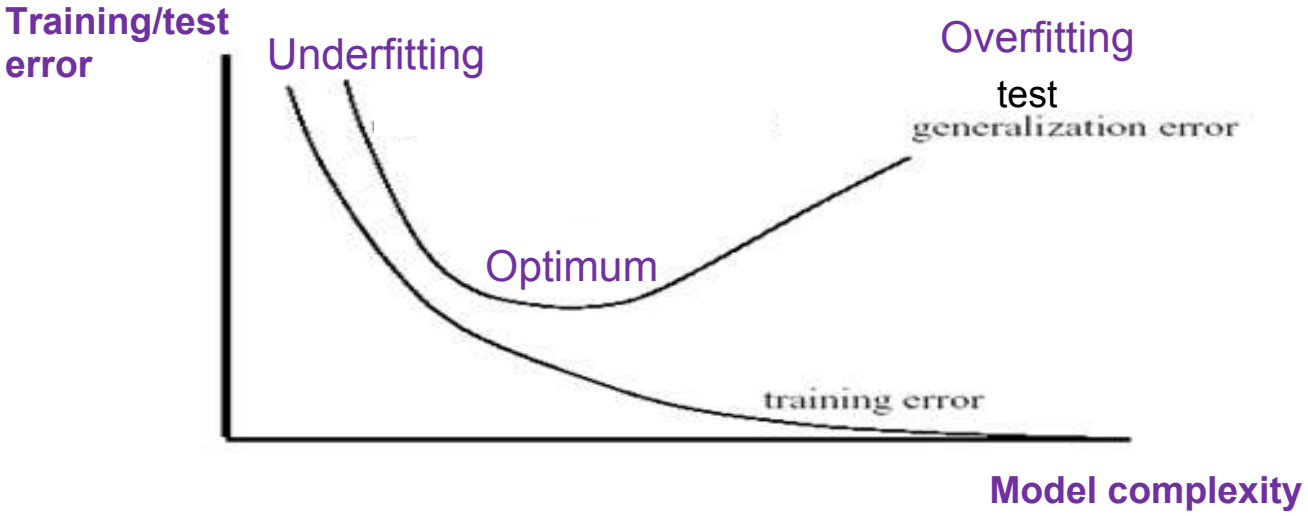
Model is too complex MEMORIZE

Model is just right GENERALIZE

Generalize; Don't Memorize!



We can plot learning curves to spot overfitting



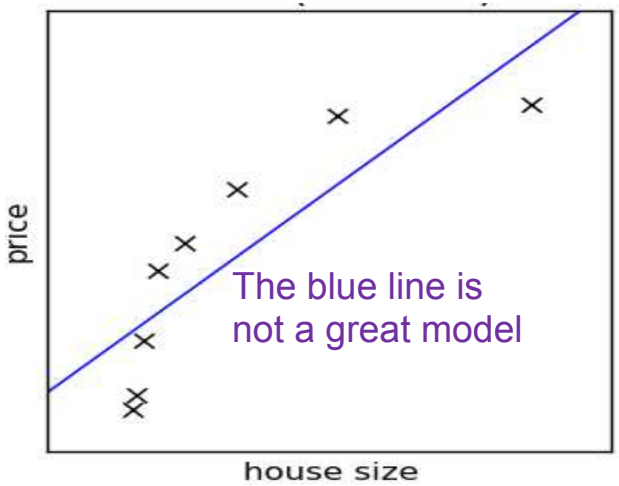
Questions?

Overfitting

- “Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study. In reality, the data often studied has some degree of error or random noise within it. Thus attempting to make the model conform too closely to slightly inaccurate data can infect the model with substantial errors and reduce its predictive power.”

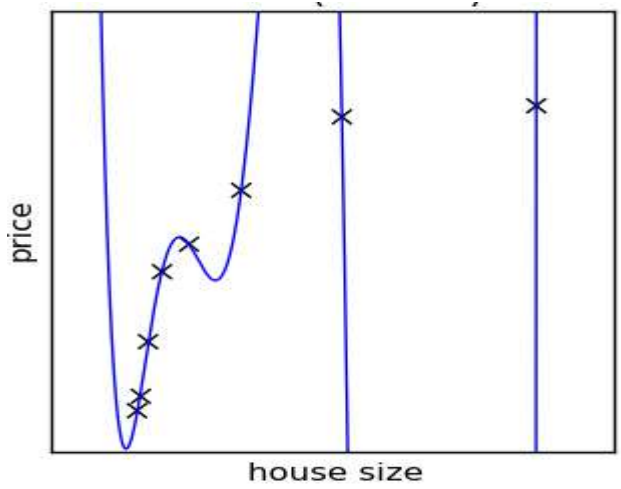
Simple models cause underfitting

- In underfitting, the training error and test error are high
- What does too simple mean?
  - Too few features
  - Use of features is not ‘complex’
- Examples
  - Can you predict well if an email is spam using only the word ‘free’?



Overfitting is when the model learns the noise and signal

- If the model overfits
  - It cannot generalize well to new data
  - It memorizes the training data
  - It has a low training error,
    - and high test error



The model has no error, but it does not seem to represent the data well

Overfitting is caused by:

- Too much model complexity
  - Typically too many features
  - Too many parameters to learn

- Too little data or not enough data diversity

Revisit our favorite question ☺️: “What is the “best” parameter for my training data? How do I find?”

Questions?

Avoiding Overfitting

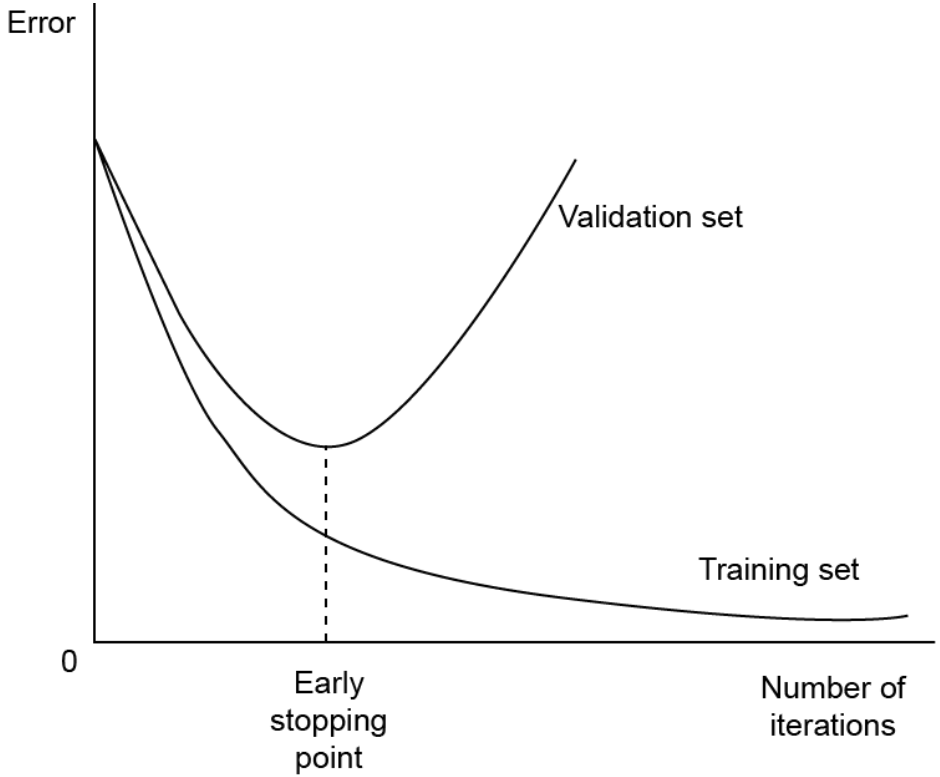
Addressing Overfitting

- Use More Data → Can lean more parameters
- Reduce the number of features or dimensions
  - Select which features to keep
  - Reduce dimension
  - Select which model to use

How to prevent/reduce overfitting?

- Train with more data
- Reduce/Remove features
- **Early Stopping**
- **Regularization**
- Ensembling (later)

Early Stopping



Regularization

Regularization

Regularization is a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting.

What to Minimize? (regularization)

- We were minimizing:

$$\varepsilon = \sum_i (z_i - y_i)^2 = \sum_i (z_i - f(x_i))^2$$

- We can minimize
  - Fit error + some measure of complexity of the model

ε + Sum of squares of Weights

ε + Number of Non-zero Weights

Summary

- Find model that is simple and fitting the data.
- Problem of overfitting.
  - How to detect? What to observe during learning? How to avoid.
- Regularize the solution by adding extra term that also minimize the “complexity”.
- Right way to design solution with
  - Train, Val and Test splits.

How to do experiments?

Practice of ML

1. Do not test your model on data it is been trained with!

the basic rule!!

Complexity Vs. Goodness of Fit

- More complex models can fit the data better.
    - but can overfit.
    - If we have more data, let us more complex models
- **Model selection**: enumerate possible model/hypothesis classes of increasing complexity, stop when "error levels" saturates or increase. (Often not attempted)
- **Regularization**: explicitly define a metric of complexity and penalize it in addition to loss

Why do we evaluate a model's performance?

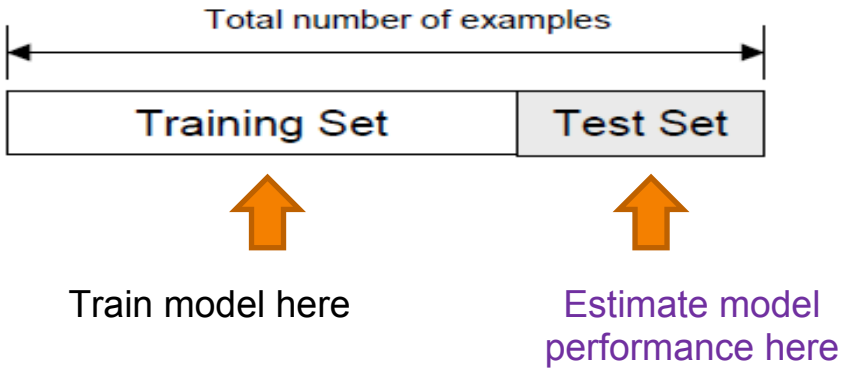
1. To know how well it works
   - Reporting results
   - Making scientific/business decision
2. To compare models
   - For choosing which models are best
   - For tuning a given model's parameters

How to measure performance?

(Accuracy, mAP etc. are just measures)

Holdout test set: The naïve approach

- Randomly split the entire dataset into:
  - Training set: A dataset used for training the model
  - Test set (a.k.a validation set): Data only used for testing the model



The three-way split

- **Training set**

  A set of examples used for learning

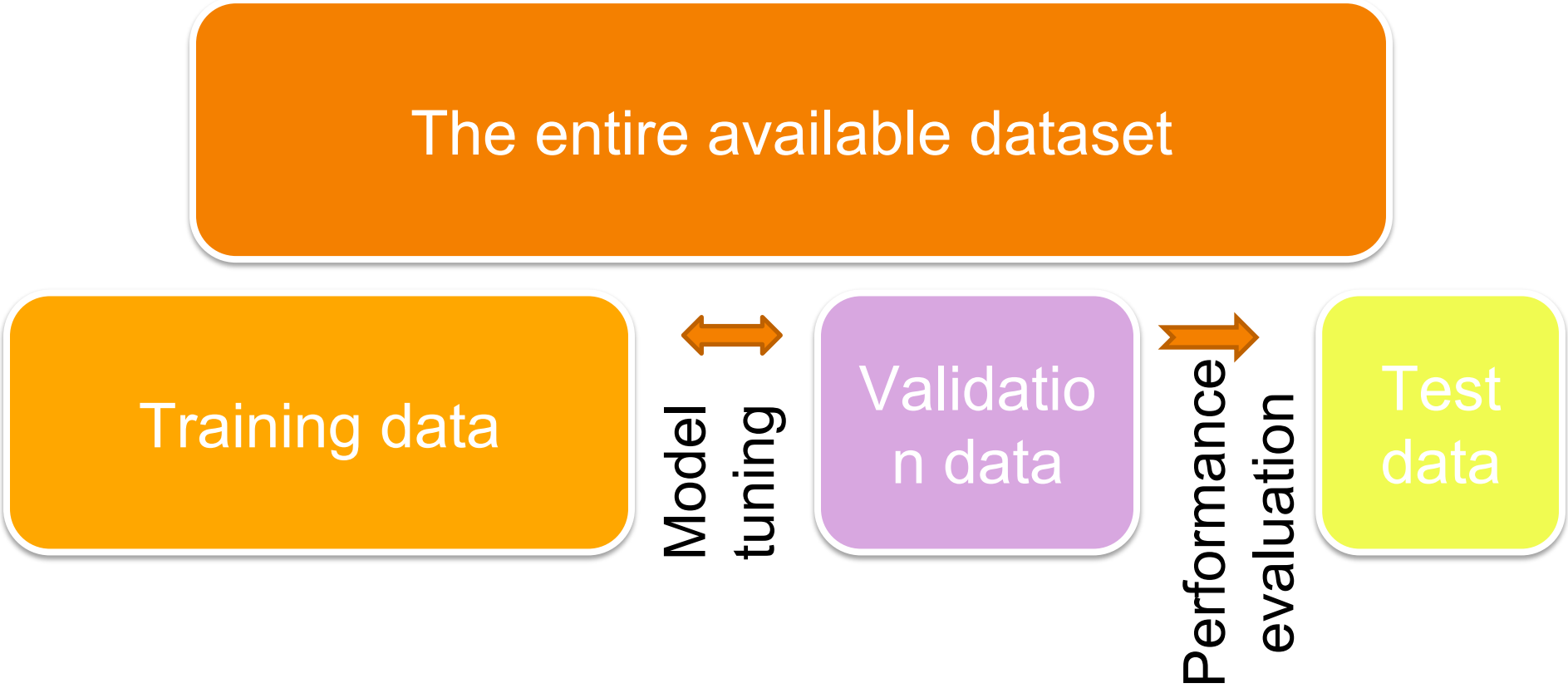
- **Validation set**

  A set of examples used to tune the parameters of a classifier

- **Test set**

  A set of examples used only to assess the performance of fully-trained classifier.

The three-way split



How to perform the split?

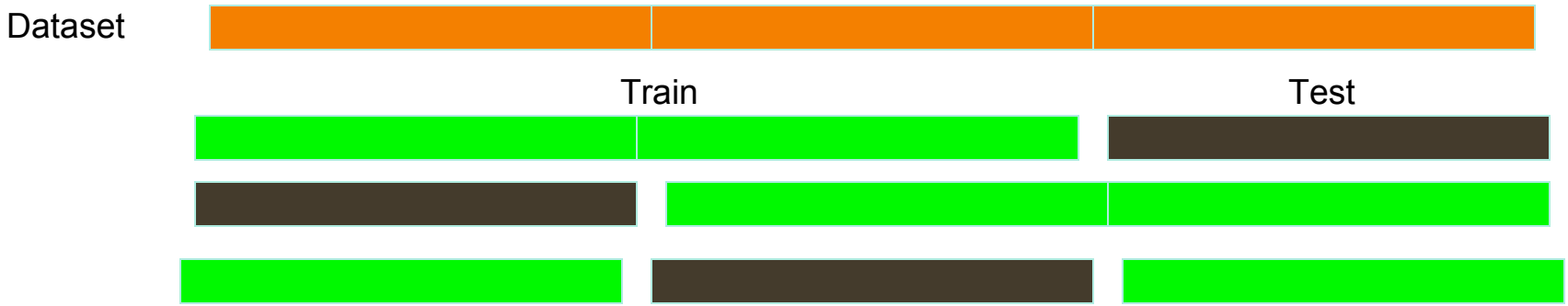
- How many examples in each data set?
  - Training: Typically 60-80% of data
  - Test set: Typically 20-30% of your data set
  - Validation set: Around 20% of data
- Examples
  - 3 way: Training: 60%, Val: 20%, Test: 20%
  - 2 ways: Training 70%, Test: 30%

Holdout summary

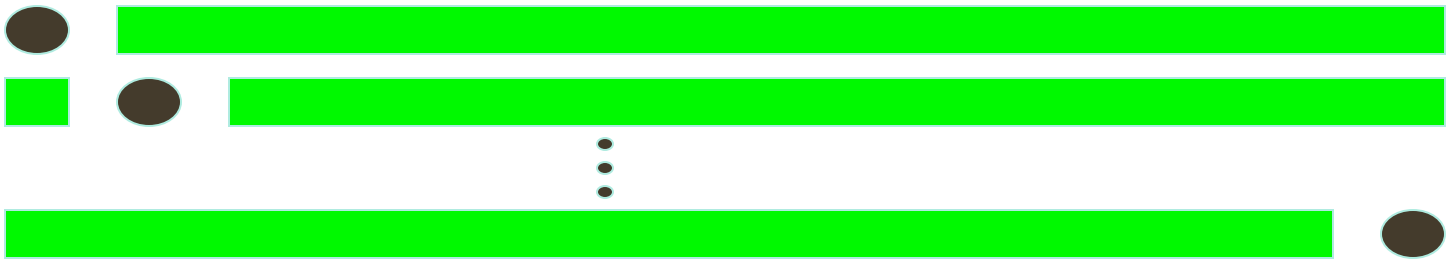
- Positive
  - Intuitive; Usually easy to perform; Considered the ideal method for evaluation
- Drawbacks:
  - In small datasets you do not have the luxury of setting aside a portion of your data
  - The performance will be misleading if we had unfortunate split

Common Splitting Strategies

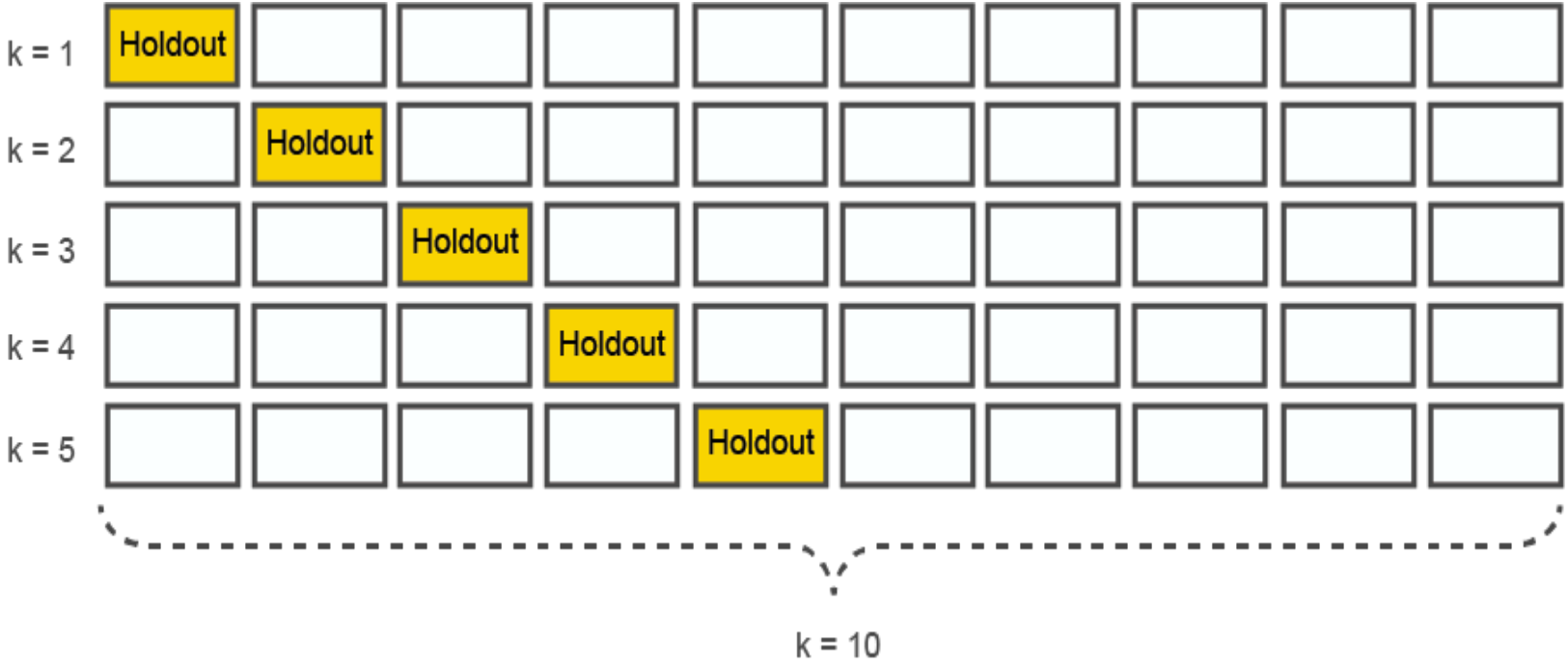
- k-fold cross-validation



- Leave-one-out (n-fold cross validation)



Eg. Cross validation (K=10)



Cross - Validation

- How do you summarize the performance? $E = \frac{1}{K}\sum_{i=1}^{K} E_i$
  - **Average**: Usually average of performance between experiments
- How many folds are needed?
  - Common choice: 5-fold or 10-fold cross validation (Some nice numbers)
  - Large datasets → even a 3-Fold cross validation will do
  - Smaller datasets → bigger K. Why?
  - Leave-One-Out approach (K=n). K is equal to the number of examples n. Used for very small datasets

More reliable Error Estimates

- Split the data into x and 100-x percentages for train and evaluation. Repeat the task many times and average the error.
- The same for K-fold

Imbalanced Class Sizes

- Sometimes, classes have very unequal frequency
  - Cancer diagnosis: 99% healthy, 1% disease
  - eCommerce: 90% don’t buy, 10% buy
  - Security: >99.99% of citizens are not terrorists
- Similar situation with multiple classes
- This creates problems for training and evaluating a model

Evaluating a model with imbalanced data

- Example: 99% of people do not have cancer
- If we simply create a ‘trivial classifier’ – predict that nobody has cancer, then 99% of our predictions are correct! Bad news! – we incorrectly predict nobody has cancer
- If we make only a 1% mistake on healthy patients, and accurately find all cancer patients
    - Then ~50% of people that we tell have cancer are actually healthy!

Learning in imbalance: balance the classes

- **Important: Estimate the final results using an imbalanced held-out (test) set**
- How to create a “balanced” set
  - Treat the majority class
    - Down-sample
  - Treat the small class
    - Up-Sample the small class; Assign larger weights to the minority class samples

Down-sample and Up-sample

- Down-sample:
  - Sample (randomly choose) a **random subset of points** in the majority class
- Up-sample:
  - Repeat/augment with minority points
  - Also see/reference: Synthetic Minority Oversampling Technique (SMOTE) (no detail now.)

Questions?

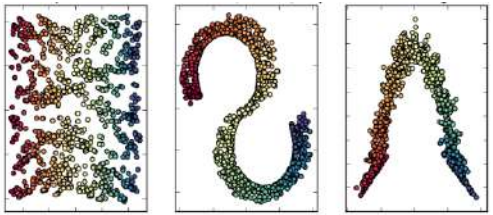
Data Visualization

When Data is High-Dimensional

Motivation

Two sides to data visualization:

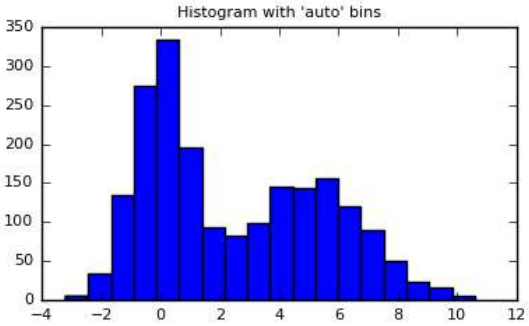
- **Data Exploration** – Making sure **you** understand your data



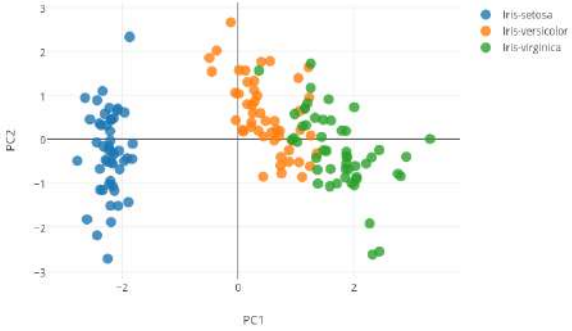
- **Data Communication** – Making sure **others** understand your insights and/or can use your data easily

Motivation

- With high-dimensional data, both stages are potentially difficult.
- Standard visualization methods can usually capture only one or two variables at a time



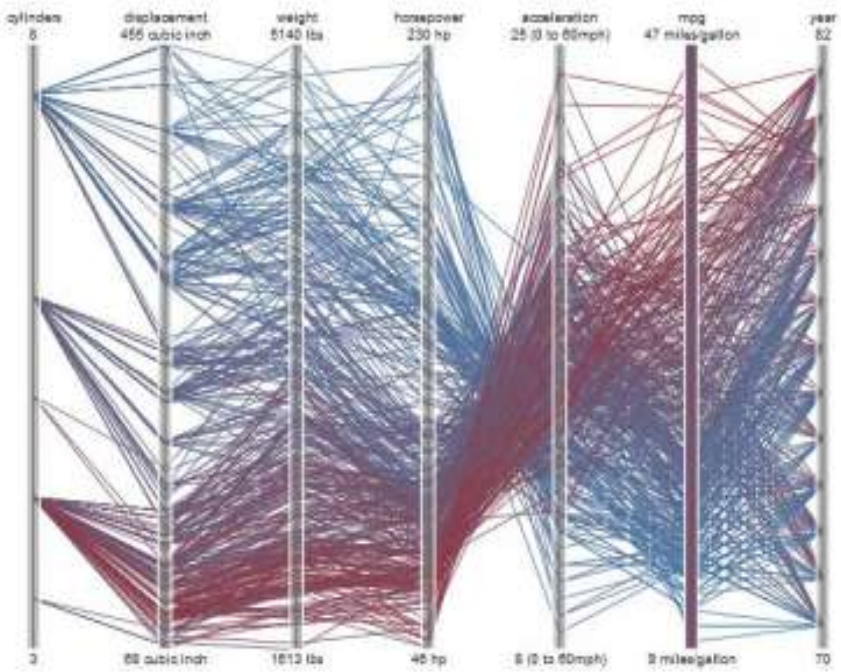
SIMPLE HISTOGRAMS



PLOT ON 2 PRINCIPLE COMPONENTS

Why Data Visualization

- How do we look at HD data?
  - Dimensionality Reduction
  - Other methods
    - Have limitations



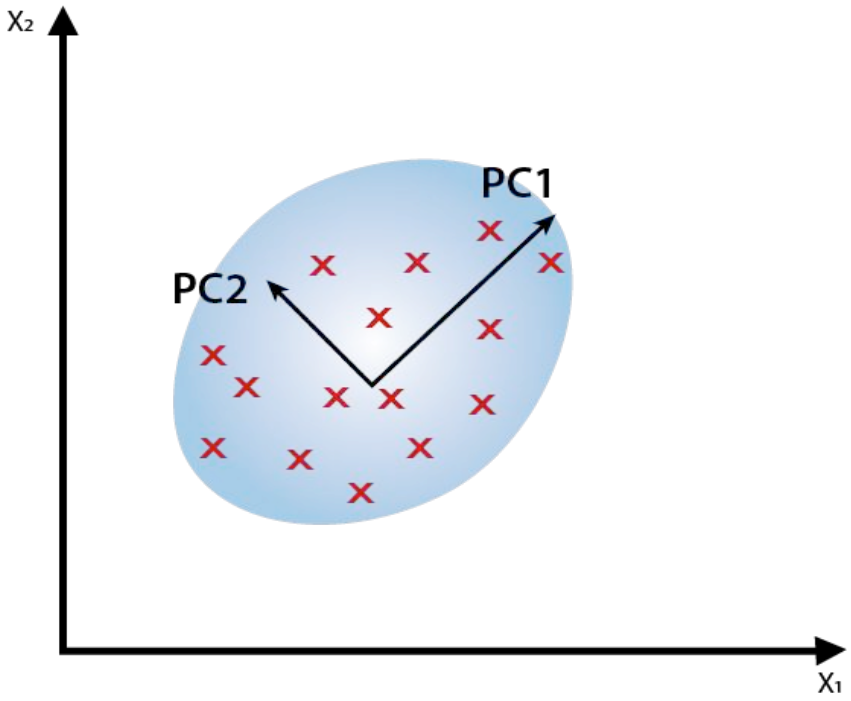
PLOT INDEPENDENT DIMENSIONS

Principal Component Analysis

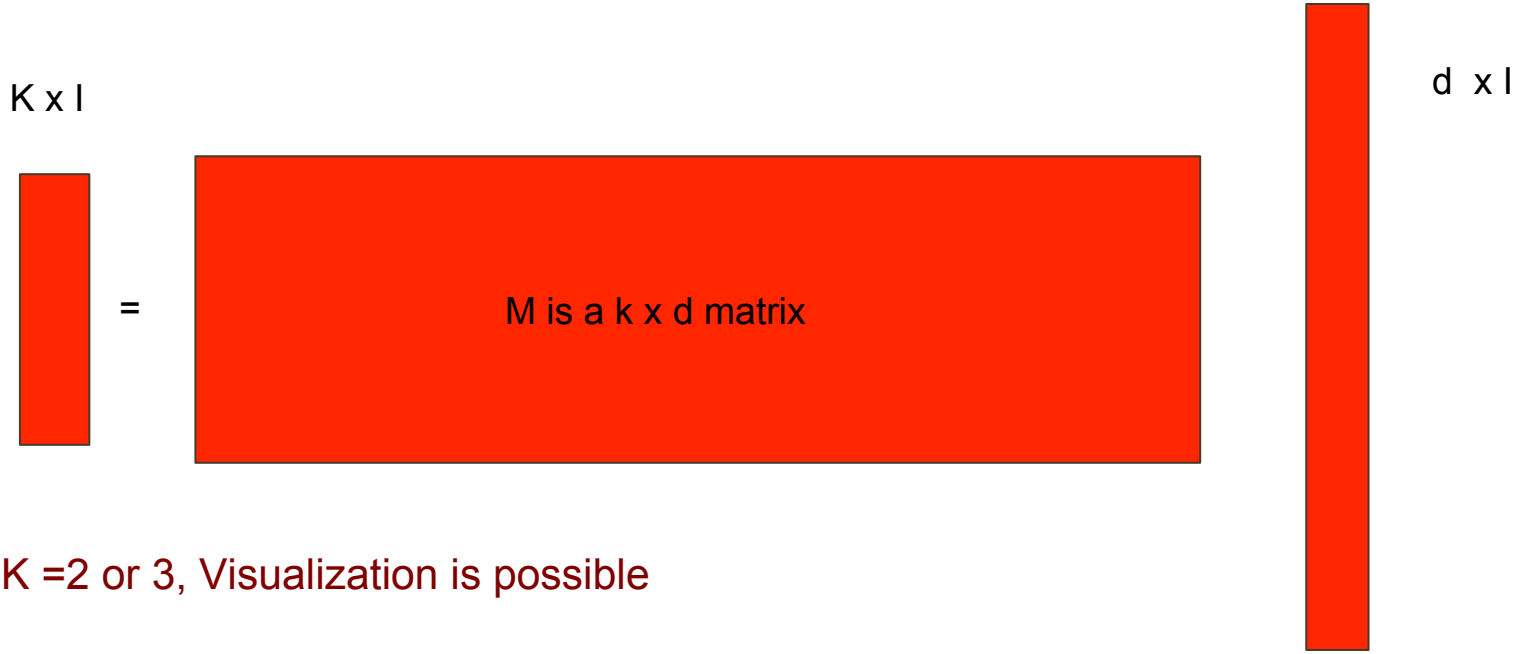
Simplifying Representations

Principal Component Analysis

- PCA is used to reduce the dimensionality of data



PCA based Feature Extraction



When K =2 or 3, Visualization is possible

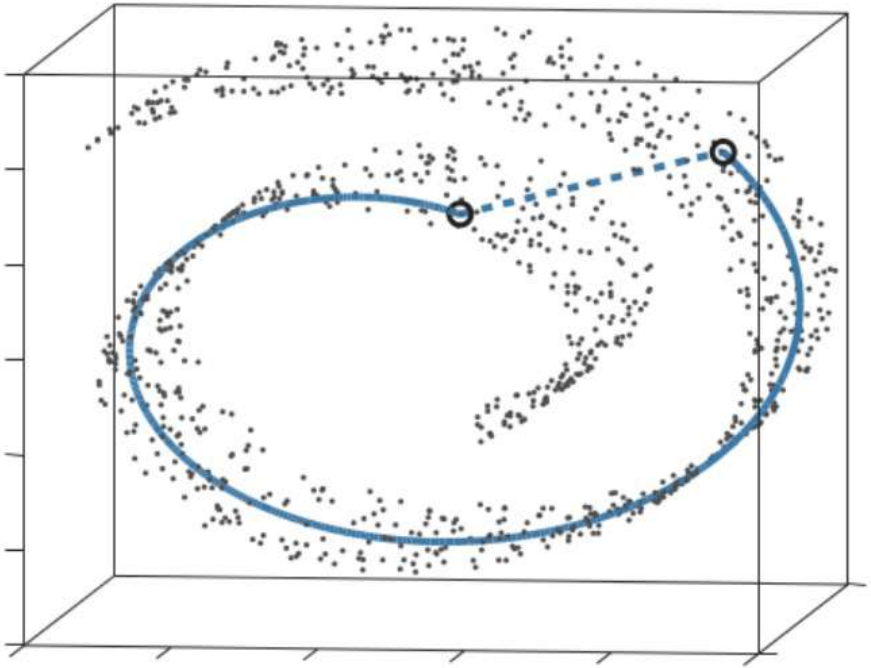
ISOMAP and LLE

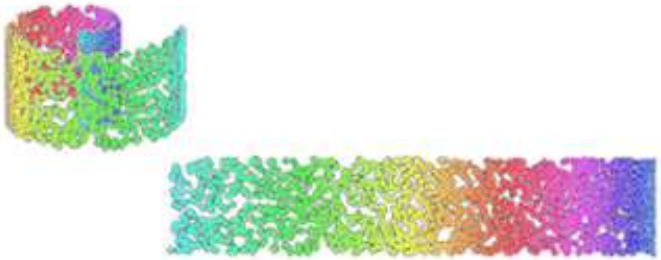
Nonlinear Dimensionality Reduction/Visualization

Dimensionality Reduction Approaches:

- **Global Approach:** All distances in HD are equally important and should be captured by LD representation
- **Local Approach:** Only smaller distances in HD are meaningful/reliable/interesting to us
  - Could also weigh smaller and larger distances differently

In complex datasets, large distances are usually *less* indicative





Learning Problem

high-dimensional data set

$$\mathcal{X} = \{x_1, x_2, \ldots, x_n\} \longrightarrow$$

two or three-dimensional data

$$\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$$

Formal Framework

minimize an objective function that measures the discrepancy between similarities in the data and similarities in the map

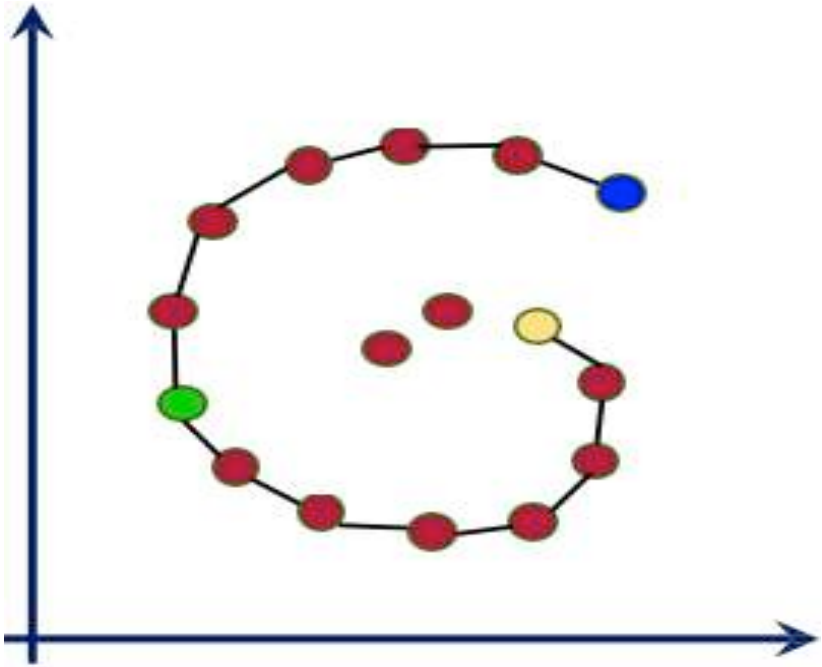
MDS (Multidimensional scaling)

minimize an objective function that measures the discrepancy between similarities in the data and similarities in the map

Distance between samples in “high” dimension and “low’ dimension is same (or D-d) is minimized.

ISOMAP

- d(●,●) > d(●,●)
- Is Euclidean metric the right distance metric
- How to robustly measure distances along the manifold (MD)?

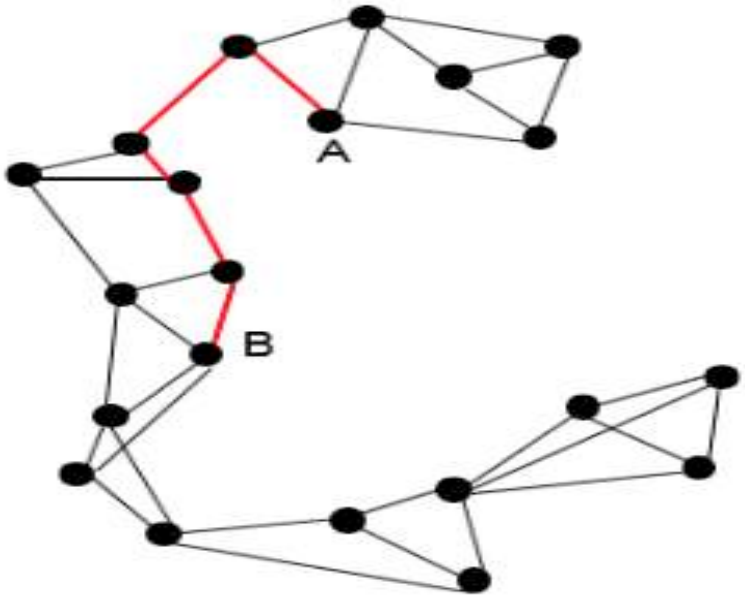


ISOMAP

- Distance on the manifold in HD is preserved in the LD.
- (or the discrepancy/loss is minized.)

ISOMAP

- How does ISOMAP measure the MD?
- Connect each data point to its k nearest neighbors in the high-dimensional space.
- MD(A,B) = ShortestPath (A,B) in this *neighborhood graph*.
- Compute the low-dimensional embedding as in Metric MDS.

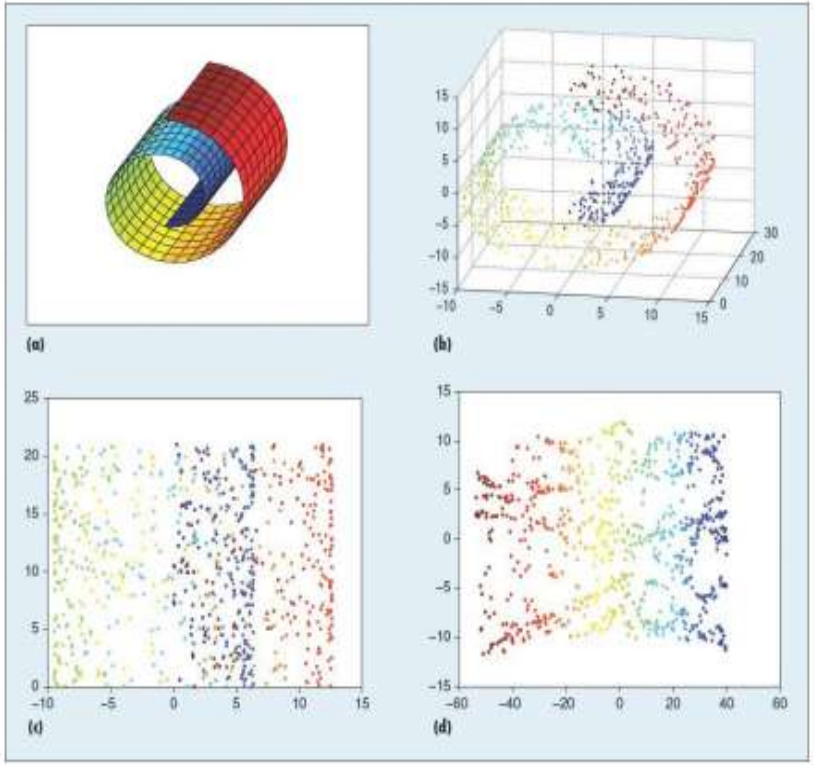


LLE: Locally Linear Embedding

- Idea: Preserve the structure of local neighborhood
- Approach:
  - Represent each point as a weighted combination of its Neighbors in HD.
  - Find a LD representation that minimize the representation error with the same combination.

What about the Swiss Roll?

- The bottom two are ISOMAP and LLE



MDS, ISOMAP and LLE on MNIST

MDS

ISOMAP

LLE

Questions?

t-SNE

Improving Visualisation

SNE and t-SNE

- Idea is simple: Instead of distance think about probabilities. P_{ij} as the probability of j in the neighborhood of i.
- For each point, we have now a probability vector (of size N).
  - SNE uses Gaussian. T-SNE uses another t-distribution.
- We want it to be the same in low dimensional.
- Optimize using gradient descent.

Computing the LD Embedding

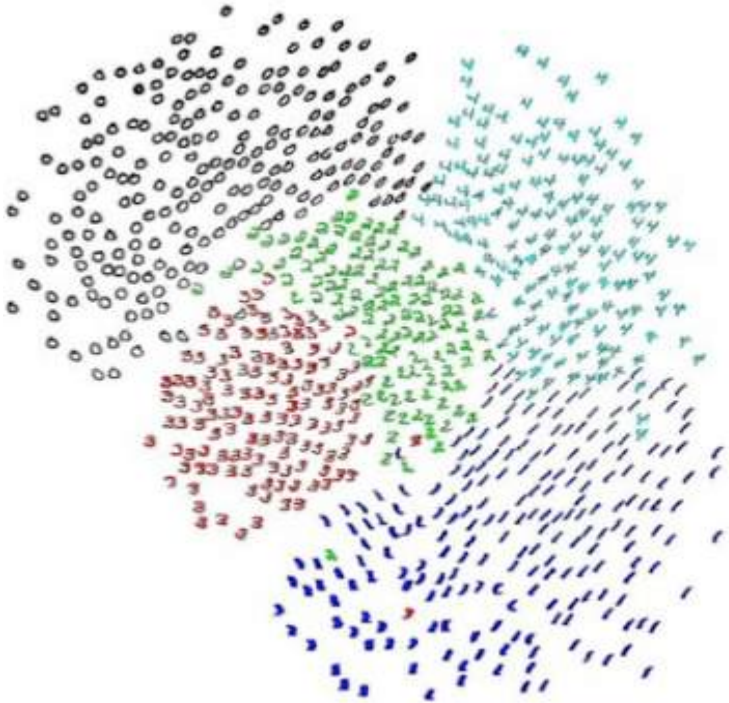
$$Cost = \sum_i KL\left(P_j \parallel Q_i\right) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- For points where p_{ij} is large and q_{ij} is small we lose a lot.
  - Nearby points in high-D really want to be nearby in low-D

SNE on MNIST

MNIST Handwritten digits dataset

- 28x28 binary images
- Large variations in writing



Details(SNE and t-SNE)

$$p_{ij} = \frac{\exp\left(-\delta_{ij}^2/\sigma\right)}{\sum_k \sum_{l \neq k} \exp\left(-\delta_{kl}^2/\sigma\right)}, \qquad \text{for } \forall i \forall j : i \neq j.$$

$$C(Y) = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

$$q_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{y}_k - \mathbf{y}_l\|^2)}, \qquad \text{for } \forall i \forall j : i \neq j,$$

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}, \qquad \text{for } \forall i \forall j : i \neq j,$$

MDS, ISOMAP and LLE on MNIST

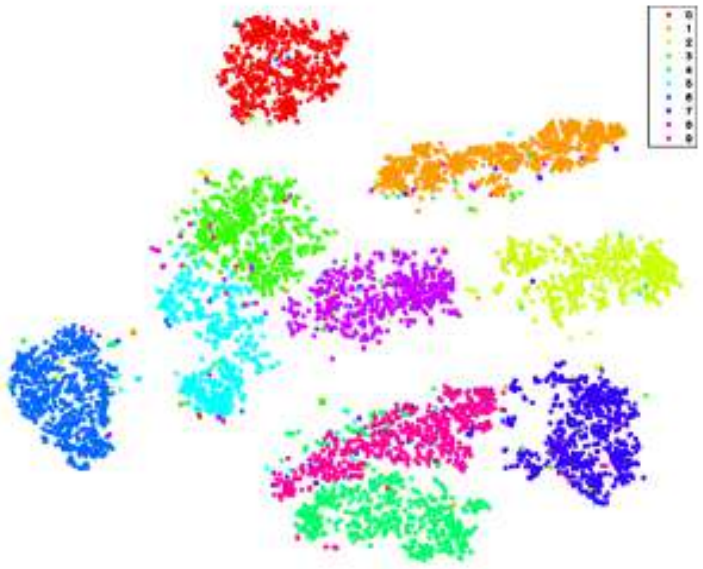
MDS

ISOMAP

LLE

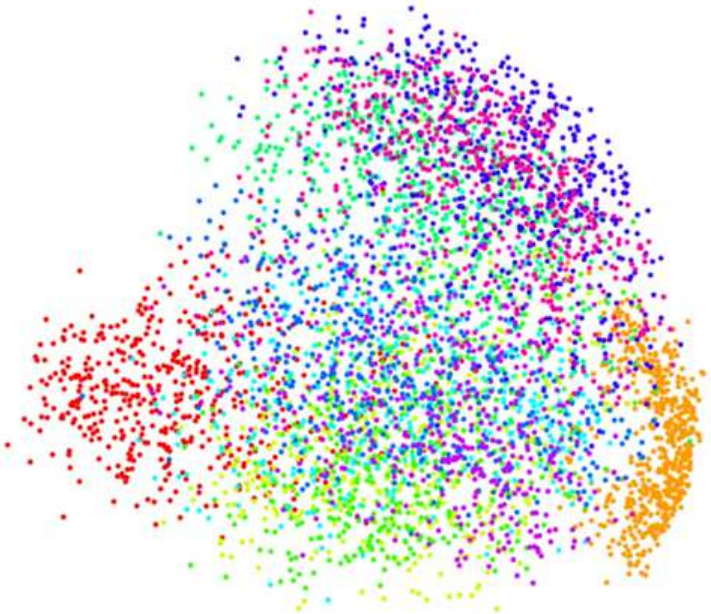
t-SNE on MNIST; Summary

- Classes are much better separated
- Note that the method is unsupervised!!!
- Efficient approximations exist
- Most popular LD visualization at the moment.

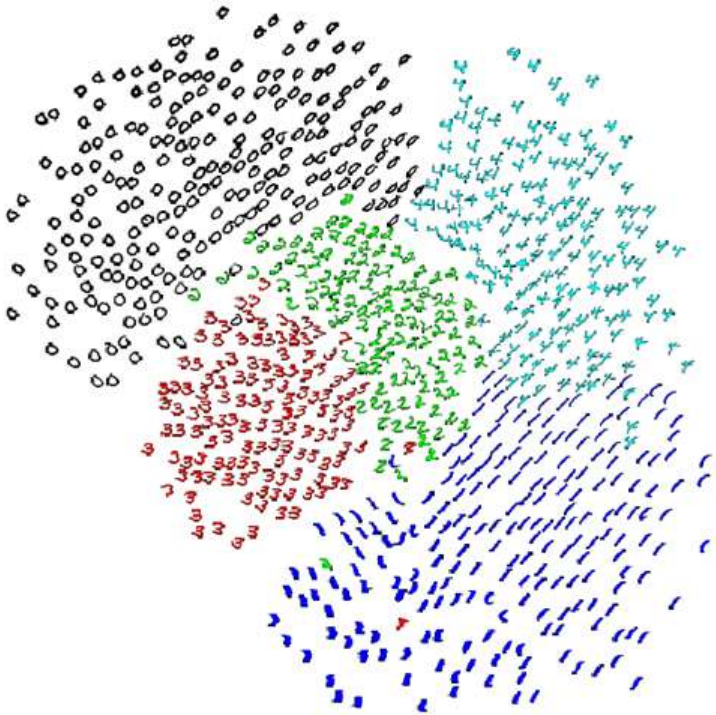


PCA

PCA on MNIST *(0-9)*



SNE on MNIST *(0-5)*



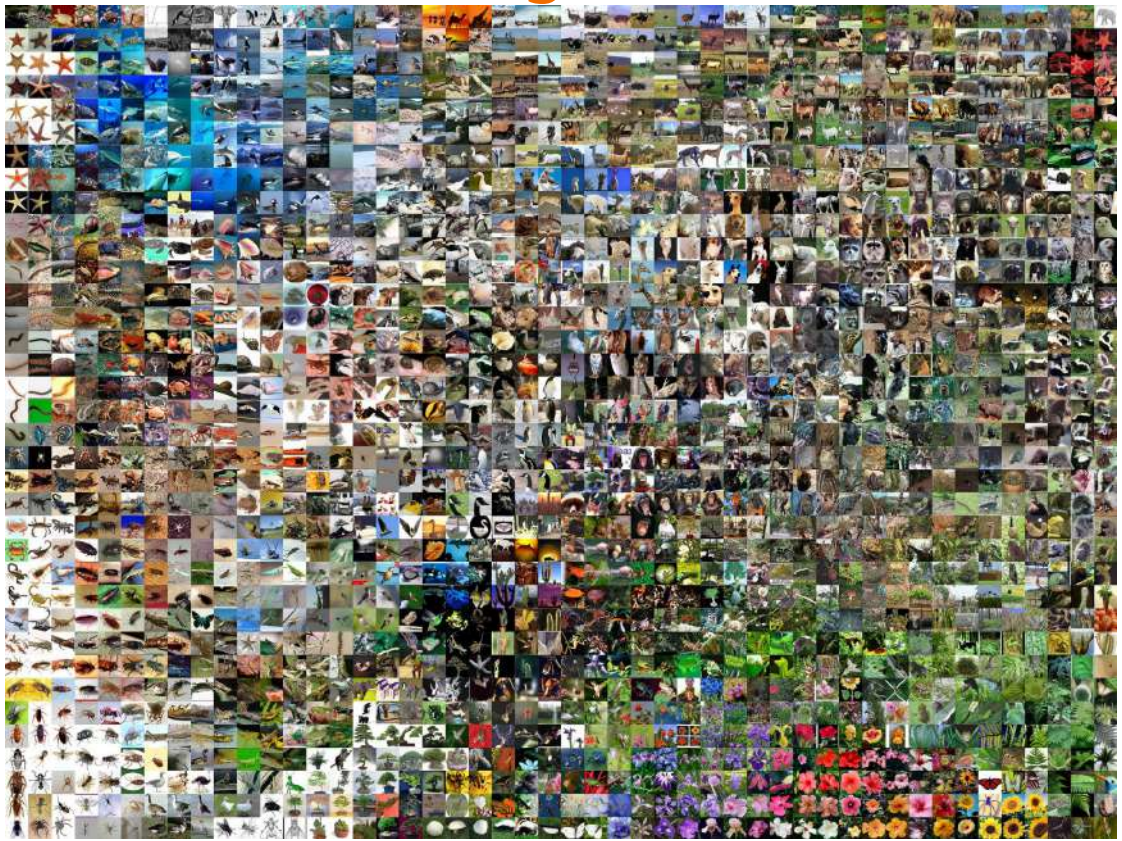
Are we overfitting?

“classic” Machine Learning	VS	Visualization
Goal: Generalization		Goal: Visualization
Given a Training set, Do well on a Test set.		We just want to “do well” on our data (“training set”)
Overfitting is undesirable		**“Overfitting” is desirable**

9

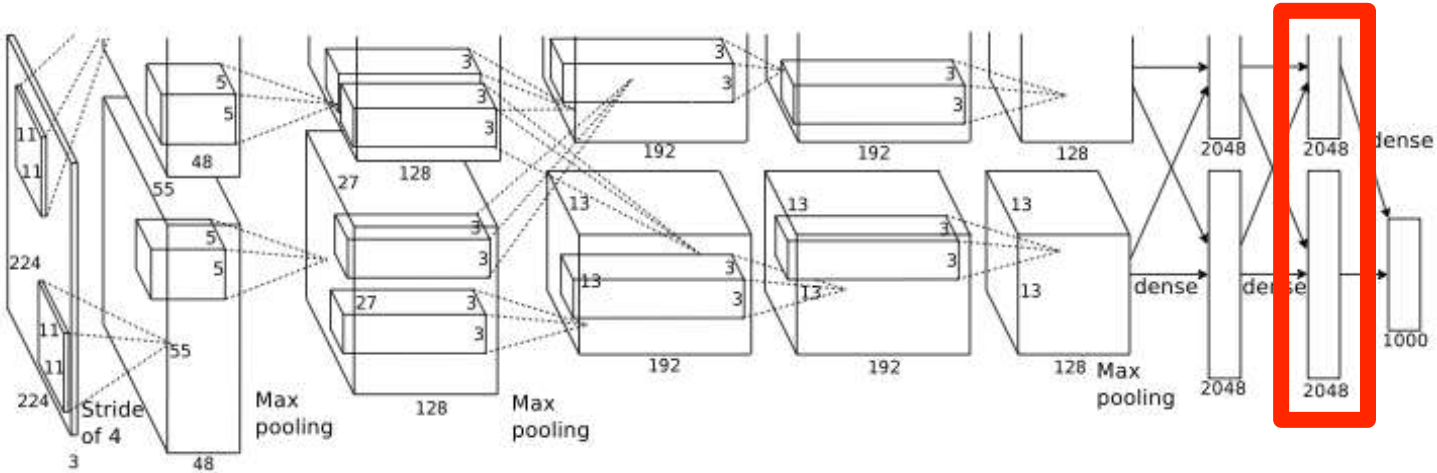
https://github.com/oreillymedia/t-SNE-tutorial

ImageNet



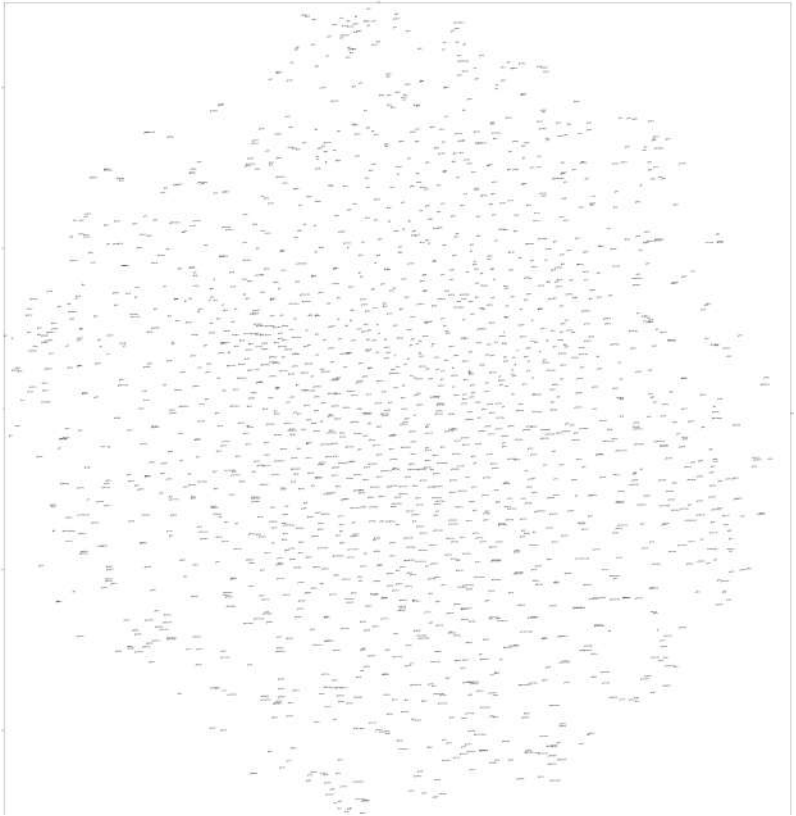
ImageNet

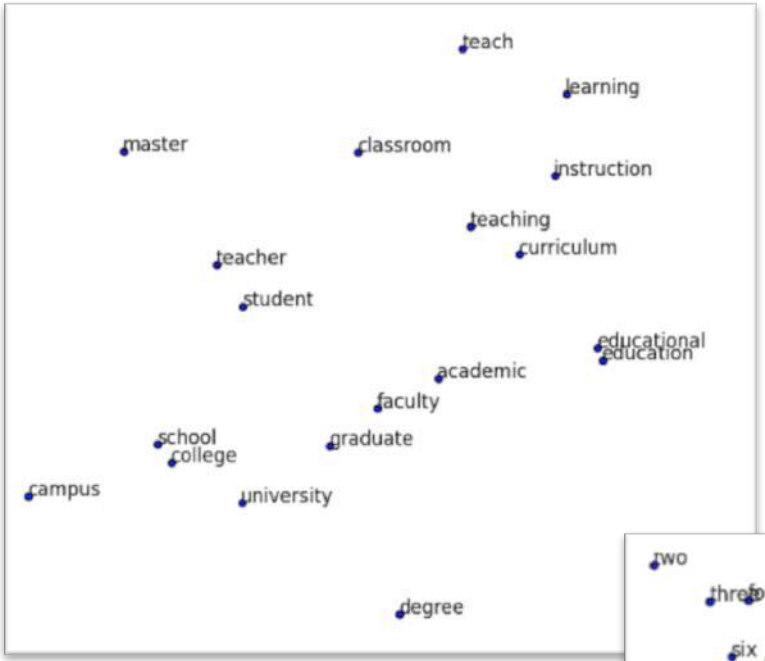
AlexNet

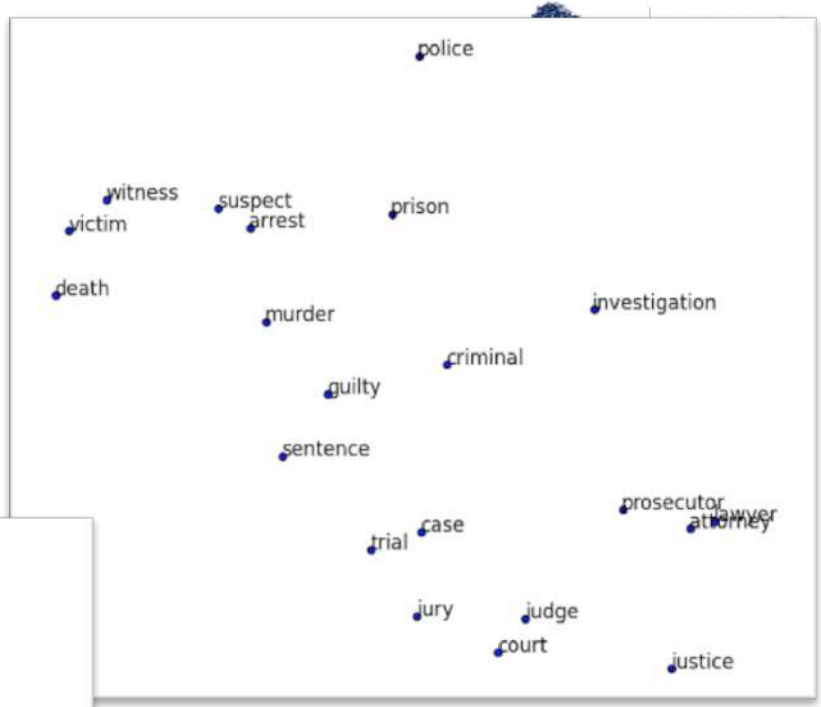


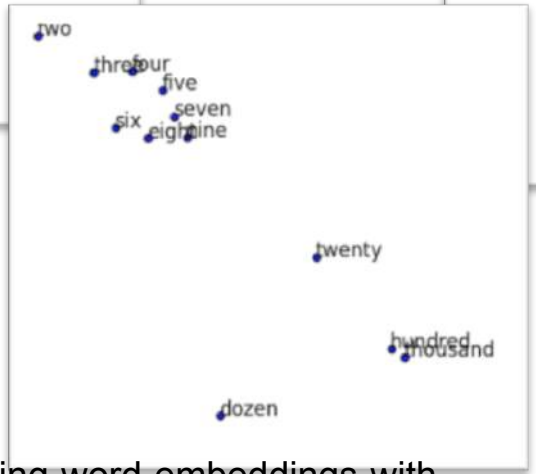
Word2vec

- Input: large corpus of text
- Embed words into a high-dim space
  - words with common contexts in the corpus are close in the space









http://nlp.yvespeirsman.be/blog/visualizing-word-embeddings-with-tsne/

Questions?