Machine Learning Basics

- Classification & Text Categorization

- Features
- Overfitting and Regularization

- Perceptron Classifier

- Supervised Learning V.S. Unsupervised Learning
- Generative Learning V.S. Discriminative Learning

- Baselines

Text Categorization Examples

- Blogs
  - Recommendation
  - Spam filtering
  - Sentiment analysis for marketing
- Newspaper Articles
  - Topic based categorization
- Emails
  - Organizing
  - Spam filtering
  - Advertising on Gmail
- General Writing
  - Authorship detection
  - Genre detection

Text Classification – who is lying?

- I have been best friends with Jessica for about seven years now. She has always been there to help me out. She was even in the delivery room with me when I had my daughter. She was also one of the Bridesmaids in my wedding. She lives six hours away, but if we need each other we'll make the drive without even thinking.

- I have been friends with Pam for almost four years now. She's the sweetest person I know. Whenever we need help she's always there to lend a hand. She always has a kind word to say and has a warm heart. She is my inspiration.

Examples taken from Rada Mihalcea and Carlo Strapparava, The Lie Detector: Explorations in the Automatic Recognition of Deceptive Language, ACL 2009

➔ How would you make feature vectors?

Classification

- y : random variable for prediction (output)
- x : random variable for observation (input)
- Training Data = Collection of (x, y) pairs
- Machine Learning = Given the training data, learn a mapping function $f(x) = y$ that can map input variables to output variables

- Binary classification
- Multiclass classification

Classification

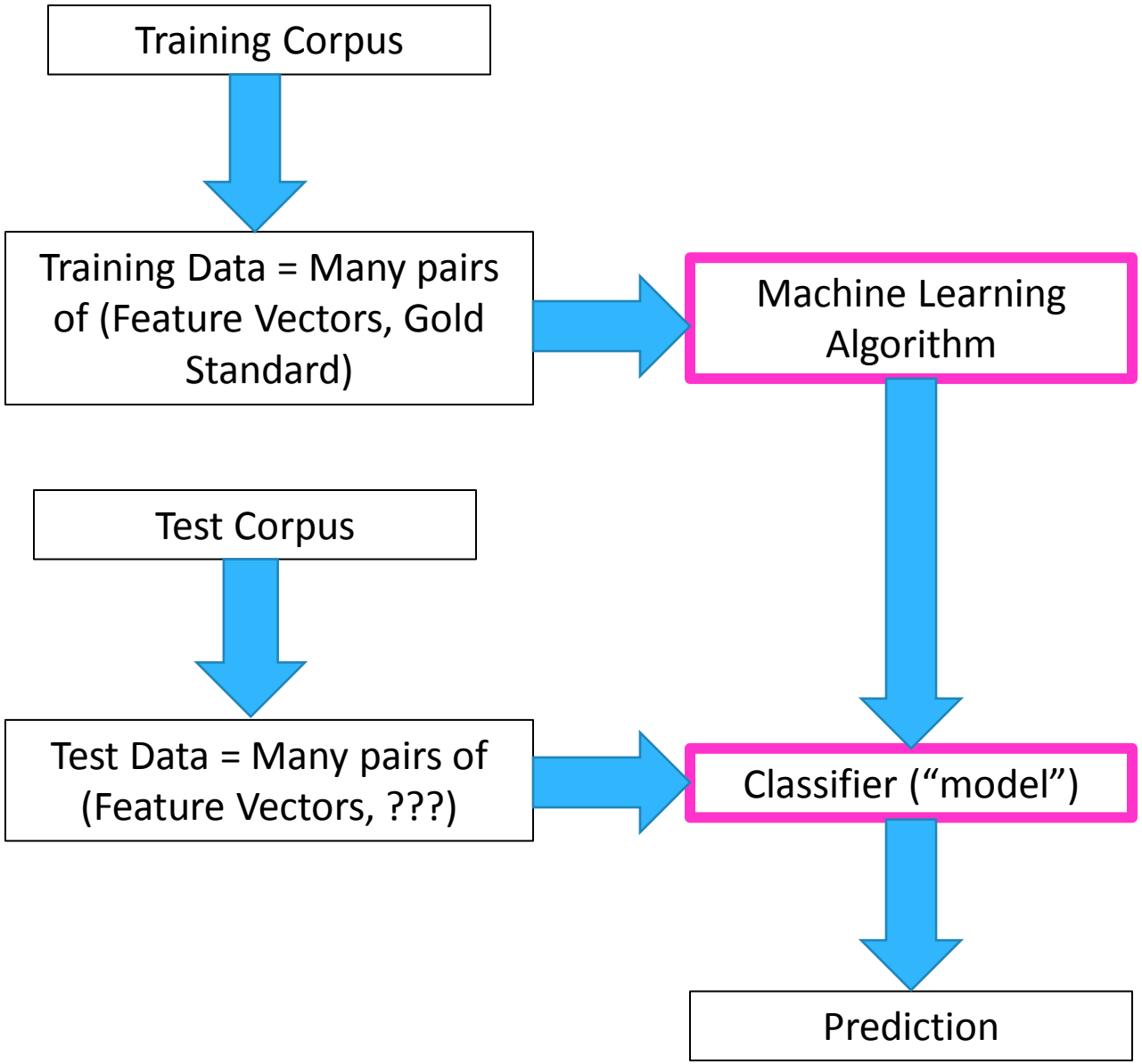
- Input variable x is defined (represented) as a feature vector $x = (f_1, f_2, f_3, \ldots)$
- Feature vector is typically defined by human, based on domain knowledge and intuitions.
- Machine Learning algorithms automatically learn the importance (weight) of each feature. That is, machine learning algorithms learn the weight vector $w = (w_1, w_2, w_3, \ldots)$

Features

- This is the place where you will use your intuitions
- Features should describe the input in a way machine learning algorithms can learn generalized patterns from them.
- You can throw in anything you think might be useful.
- Example of features – words, n-grams (used as features), syntax oriented features (part-of-speech tags, semantic roles, parse tree based features), electronic dictionary based features (WordNet)

Features

- Even an output from another classifier (for the same task) can be used as features as well!
- There is no well-established best set of features you must use for each problem – you need to explore.
- “feature engineering” – you often need to repeat the cycle of [encoding basic features, running the machine learning algorithm, analyzing the errors, improving features, running the machine learning again], and so forth
- “feature selection” – a statistical method to select a small set of better features.


Training Corpus
Training Data = Many pairs of (Feature Vectors, Gold Standard)
Machine Learning Algorithm
Test Corpus
Test Data = Many pairs of (Feature Vectors, ???)
Classifier (“model”)
Prediction

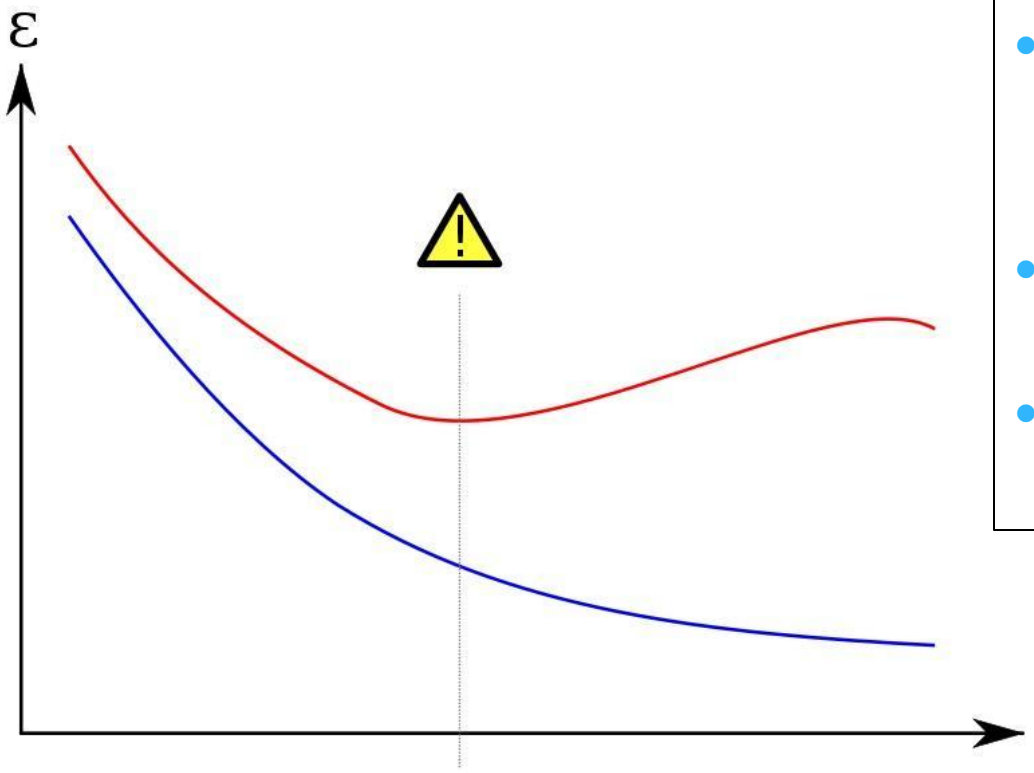
Overfitting and Regularization

- Suppose you need to design a classifier for a credit card company – you need to classify whether each applicant is likely to be a good customer or not, and you are given the training data.
- Features – ages, jobs, the number of credit cards, region of the country etc...
- How about “social security number”?

Overfitting and Regularization

- Overfitting: the phenomenon where a machine learning algorithm is fitting its learning model too specific to the training data, without being able to discover generalized concepts. – will not perform well on the previously unseen data
- Many of learning algorithms are iterative – overfitting can happen if you let them iterate for too long
- Overfitting can also happen if you define features that encourage learning models to memorize the training data, rather than generalize. (previous slide)

Overfitting and Regularization



- Y axis – performance of the trained model
- X axis – number of training cycles
- Blue – prediction errors in the training data
- Red – prediction errors in the test data

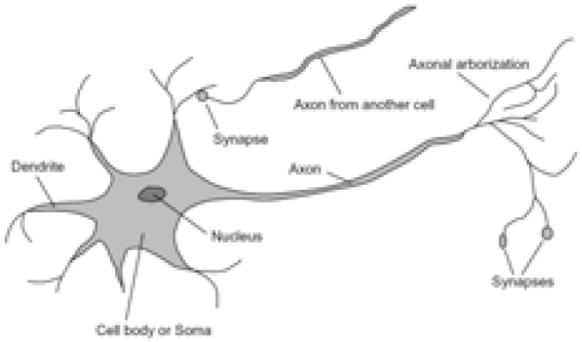
Overfitting and Regularization

- Regularization: typically enforces none of the features can become too powerful (that is, make sure the distribution of weights is not too spiky)
- Most of machine learning packages have parameters for regularization – Do play with them!

- Quiz: How should you pick the best value for the regularizing parameter?

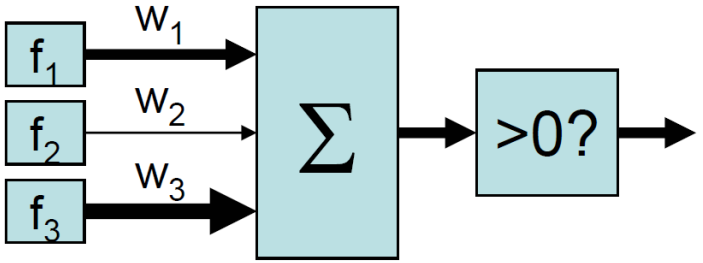
The Binary Perceptron

- Inputs are features
- Each feature has a weight
- Sum is the activation



$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x)$$

- If the activation is:
  - Positive, output 1
  - Negative, output 0



Example: Spam

- Imagine 4 features:
  - Free (number of occurrences of “free”)
  - Money (occurrences of “money”)
  - BIAS (always has value 1)

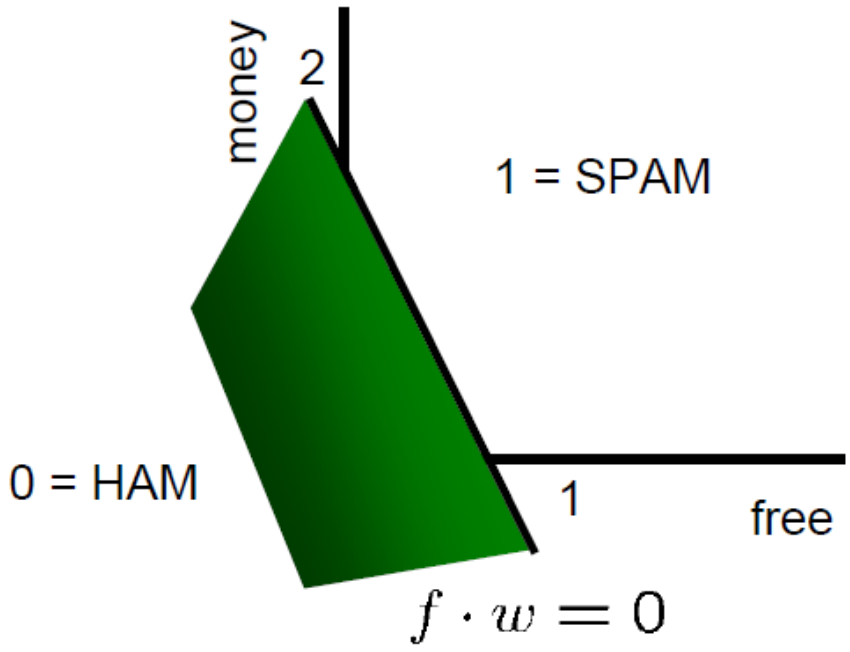
x	$f(x)$	w	$\sum_i w_i \cdot f_i(x)$
“free money”	BIAS : 1 free : 1 money : 1 the : 0 ...	BIAS : -3 free : 4 money : 2 the : 0 ...	(1)(−3) + (1)(4) + (1)(2) + (0)(0) + ... = 3

Binary Decision Rule

- In the space of feature vectors
  - Any weight vector is a hyperplane
  - One side will be class 1
  - Other will be class 0

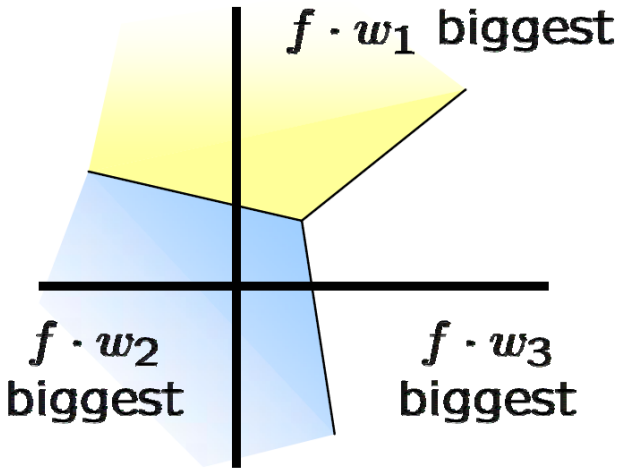
w

```
BIAS   : -3
free   :  4
money  :  2
the    :  0
...
```



The Multiclass Perceptron

- If we have more than two classes:
  - Have a weight vector for each class
  - Calculate an activation for each class



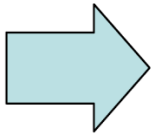
$$\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)$$

  - Highest activation wins

$$c = \arg\max_c \left(\text{activation}_w(x, c)\right)$$

Example

"win the vote"



BIAS	:	1
win	:	1
game	:	0
vote	:	1
the	:	1
...		

w_{SPORTS}

BIAS	:	-2
win	:	4
game	:	4
vote	:	0
the	:	0
...		

$w_{POLITICS}$

BIAS	:	1
win	:	2
game	:	0
vote	:	4
the	:	0
...		

w_{TECH}

BIAS	:	2
win	:	0
game	:	2
vote	:	0
the	:	0
...		

The Perceptron Update Rule

- Start with zero weights
- Pick up training instances one by one
- Try to classify

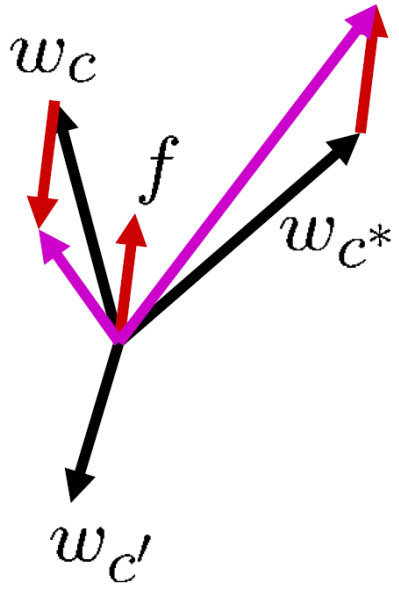
$$c = \arg\max_c \; w_c \cdot f(x)$$

$$= \arg\max_c \; \sum_i w_{c,i} \cdot f_i(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

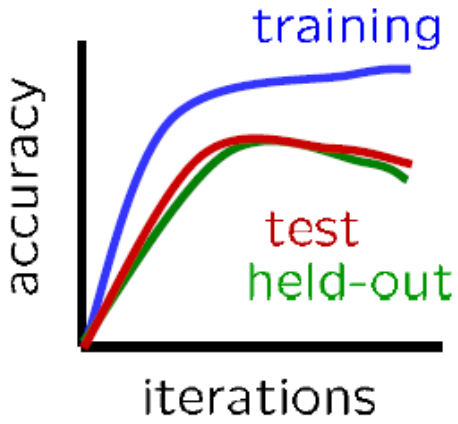
$$w_c = w_c - f(x)$$

$$w_{c^*} = w_{c^*} + f(x)$$

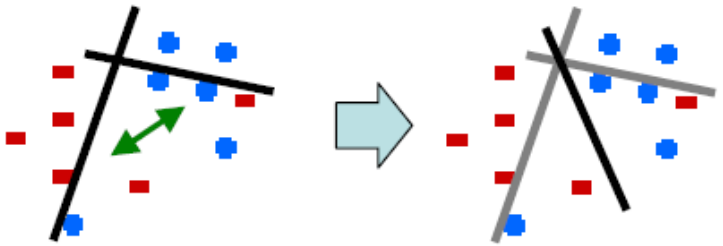


Issues with Perceptrons

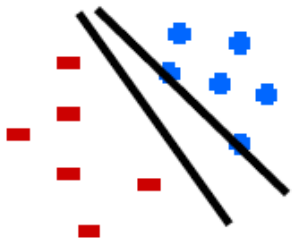
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn't quite as bad as overfitting, but is similar



- Regularization: if the data isn't separable, weights might thrash around
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution

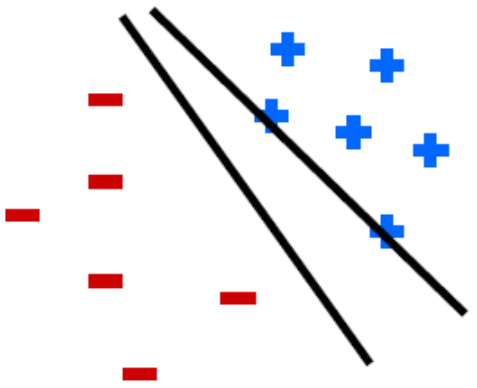


Properties of Perceptrons

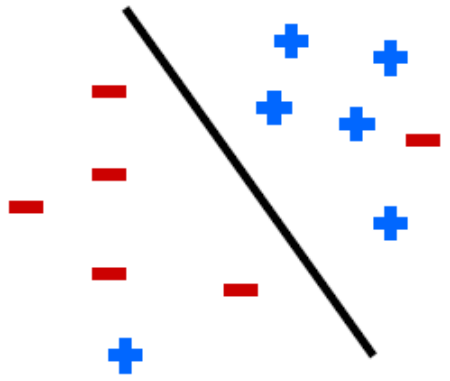
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{1}{\delta^2}$$

Separable



Non-Separable



Supervised V.S. Unsupervised Learning

- Supervised Learning
  - Training data includes "**Gold standard**" or "true prediction" which is typically from human "**annotation**"
  - For text categorization, the correct category of each document is given in the training data
  - Human annotation is typically VERY VERY expensive, which limits the size of training corpus. – the more data, the better your model will perform.
  - Sometimes it's possible to obtain gold standard automatically. Eg. Movie review data or Amazon product review data.
  - Annotation typically has some noise. Especially for NLP tasks that are hard to judge even for human. Examples?

Supervised V.S. Unsupervised Learning

- Unsupervised Learning
  - Training data does not have gold standard. Machine learning algorithms need to learn from the data based on statistical patterns alone.
  - E.g. “Clustering” or “K-nearest neighbors (KNN)”
  - Suitable when obtaining annotation is too expensive, or one has a cool idea about how to devise a statistical method that can learn directly from the data.
  - Supervised Learning generally performs better than unsupervised alternatives, especially if the size of training corpus is identical. Typically a bigger training corpus can be utilized for unsupervised learning.
- Semi-supervised Learning
  - Only a small portion of your training data comes with gold standard.

Generative V.S. Discriminative Learning

- Generative Learning
  - Tries to “generate” the output variables (often tries to generate input variables as well)
  - Typically involves “probability”
  - For instance, Language Models can be used to generate sequence of words that resemble natural language. (by drawing words proportionate to the n-gram probabilities)
  - Generative learning tends to waste the effort in preserving a valid probability distribution (that sums up to 1) which might not be always necessary in the end.

Generative V.S. Discriminative Learning

- Discriminative Learning
  - Perceptron!
  - Only care about making a correct prediction for the output variables. That is, “discrimination” between the correct prediction and incorrect ones. But doesn’t care about which is more correct than the other by how much.
  - Often does not involve probability
  - For tasks that do not require probabilistic outputs, discriminative methods tend to perform better. (because learning is focused on making correct predictions, rather than preserving a valid prob. distribution.)

“No Free Lunch”

“No Free Lunch”

- No Free Lunch Theorem by Wolpert and Macready, 1997
- Interpretation for Machine Learning: There is no single classifier that works best on all problems.
- Metaphor
  - Restaurant – classifier
  - Menu – a set of problems (dishes)
  - Price – the performance of each classifier for each problem
  - Suppose all restaurants serve identical menu, except the prices differ such that the average price of the menu is identical across different restaurants. ➔ If you are an omnivore, you cannot pick one single restaurant that is the most cost-efficient.

Practical Issues

- Feature Vectors are typically sparse
  - Remember Zipf's Law?

  ➔ Use sparse encoding (e.g. linked list rather than array)
- Different machine learning packages accept different types of features
  - Categorical features – some machine learning packages require for you to change "string" features into "integer" features. (assign unique id for each different string feature)
  - Binary features – binarized version of categorical features. Some machine learning packages will accept categorical features, but convert them into binary features internally.
  - Numeric features – need to normalize!!! Why?
  - You might need to convert numerical features into categorical features

Practical Issues

- Popular choices
  - Boosting
    - BoosTexter
  - Decision Trees
    - Weka
  - Support Vector Machines (SVMs)
    - SVMLight, libsvm
  - Conditional Random Fields (CRFs)
    - Mallet
- Weka and Mallet contain other algorithms as well.
- Definitely play with parameters for regularization!

Baseline

- Your evaluation must compare your proposed approaches against reasonable baselines.
- Baseline shows a lower bound of performance.
- Baseline can be either simple heuristics (hand-written rules) or based on simple machine learning techniques.
- Sometimes a very simple baseline might turn out to be quite difficult to beat
- Examples? Learn from research papers.

Recommended Reading to learn more about machine learning

- Part V Learning
  - Ch-18 Learning from Examples
  - Ch-20 Learning Probabilistic Models