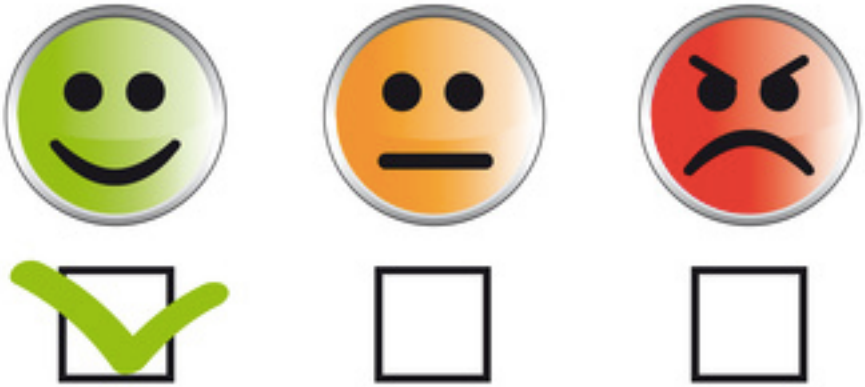
ML4NLP
Introduction to Classification



Dan Goldwasser

Purdue University

dgoldwas@purdue.edu

Statistical Language Modeling

- **Intuition**: by looking at large quantities of text we can find statistical regularities
  - *Distinguish between correct and incorrect sentences*
- Language models define a probability distribution over strings (e.g., sentences) in a language.
- ***We can use language model to score and rank sentences***

"I don't know {whether,weather} to laugh or cry"

P("I don't.. weather to laugh..") >< P("I don't.. whether to laugh..")

Language Modeling with N-grams

A language model over a vocabulary V assigns probabilities to strings drawn from V*.

Recall the chain rule:

$$P(w_1 ... w_i) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)...P(w_i|w_1 ... w_{i-1})$$

An n-gram language model assumes each word depends only on the last n-1 words:

$$P_{ngram}(w_1 ... w_i) := P(w_1)P(w_2|w_1)...P(\underbrace{w_i}_{nth\ word} | \underbrace{w_{i-n-1} ... w_{i-1}}_{prev.\ n-1\ words})$$

Unigram model	$P(w_1)P(w_2)...P(w_i)$
Bigram model	$P(w_1)P(w_2\|w_1)...P(w_i\|w_{i-1})$
Trigram model	$P(w_1)P(w_2\|w_1)...P(w_i\|w_{i-2}\ w_{i-1})$

Evaluating Language Models

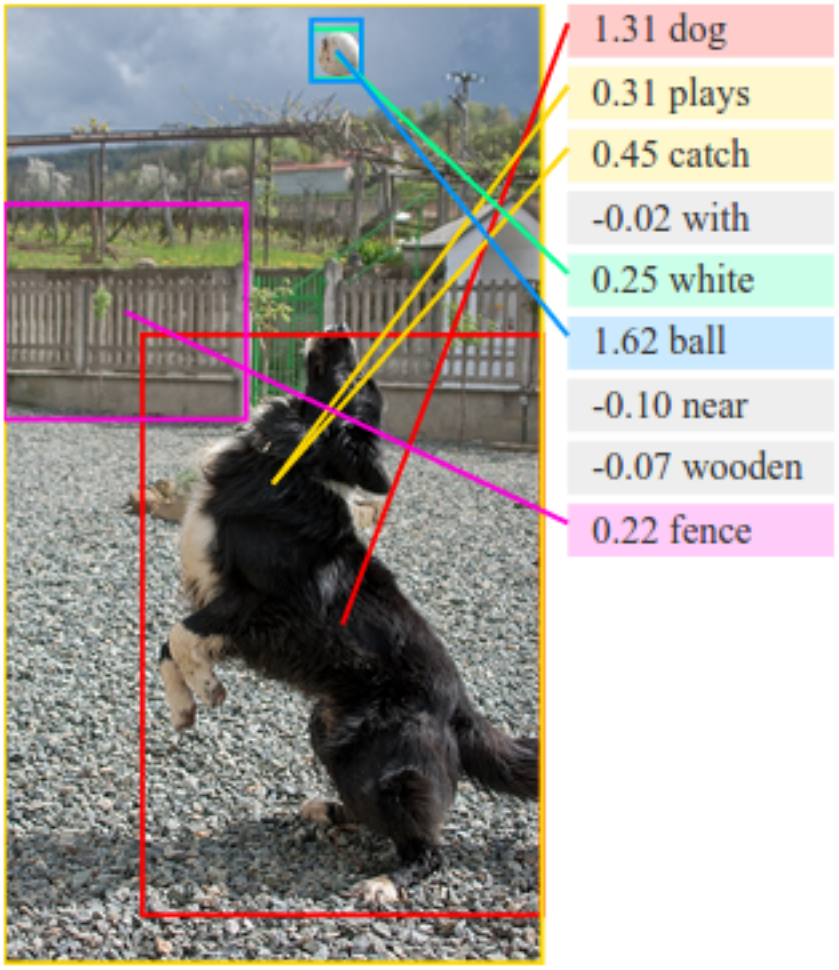
- Assuming that we have a language model, how can we tell if it's good?
- **Option 1**: ***try to generate Shakespeare..***
  - This is know as *Qualitative evaluation*
- ***Option 2: Quantitative evaluation***
  - ***Option 2.1:*** *See how well you do on Spelling correction*
    - *This is known as* ***Extrinsic*** *Evaluation*
  - ***Option 2.2:*** *Find an independent measure for LM quality*
    - *This is known as* ***Intrinsic*** *Evaluation*

When are LM applicable?

Finding regularity in language is surprisingly useful!
Easy example: weather/whether

But also-
Translation (can you produce "legal" French from source English?)

Caption Generation (combine output of visual sensors into a grammatical sentence)



Deep Visual-Semantic Alignments for Generating Image Descriptions

Classification

- *A fundamental machine learning tool*
  - Widely applicable in NLP
- **Supervised learning**: Learner is given a collection of labeled documents
  - Emails: Spam/not spam; Reviews: Pos/Neg
- Build a **function** mapping documents to labels
  - Key property: ***Generalization***
    - function should work well on new data

Sentiment Analysis

Dude, I just watched this horror flick! Selling points: nightmares scenes, torture scenes, terrible monsters that was so bad a##!

Don't buy the popcorn it was terrible, the monsters selling it must have wanted to torture me, it was so bad it gave me nightmares!

What should your learning algorithm look at?

Deceptive Reviews

Which of these two hotel reviews is deceptive opinion spam?

My husband and I stayed at the James Chicago Hotel for our anniversary. This place is fantastic! We knew as soon as we arrived we made the right choice! The rooms are BEAUTIFUL and the staff very attentive and wonderful!! The area of the hotel is great, since I love to shop I couldn't ask for more!! We will definatly be back to Chicago and we will for sure be back to the James Chicago.

I have stayed at many hotels traveling for both business and pleasure and I can honestly stay that The James is tops. The service at the hotel is first class. The rooms are modern and very comfortable. The location is perfect within walking distance to all of the great sights and restaurants. Highly recommend to both business travellers and couples.

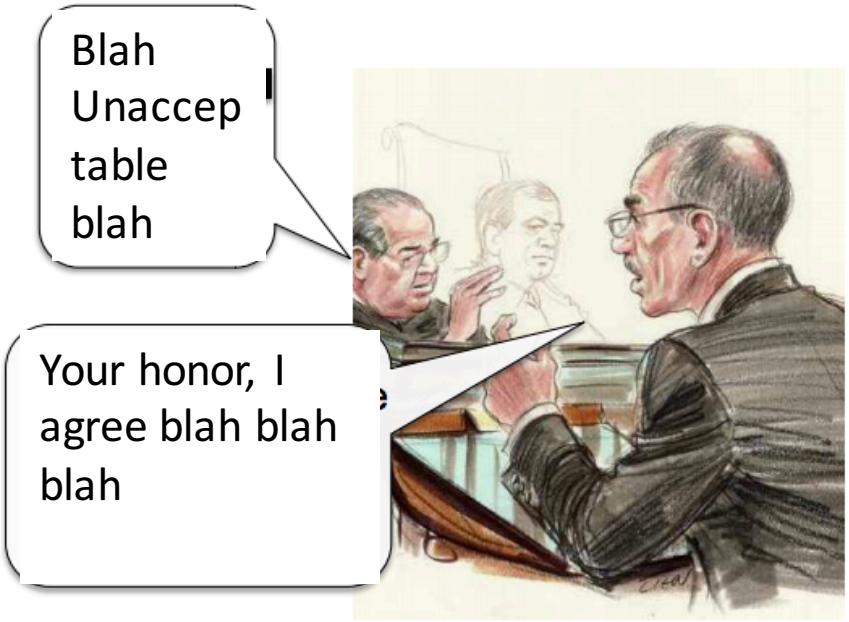
What should your learning algorithm look at?

Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Ott etal. ACL 2011

Power Relations

Can subtle, *domain-independent* linguistic cues reveal (situational) power?

Who's in charge?

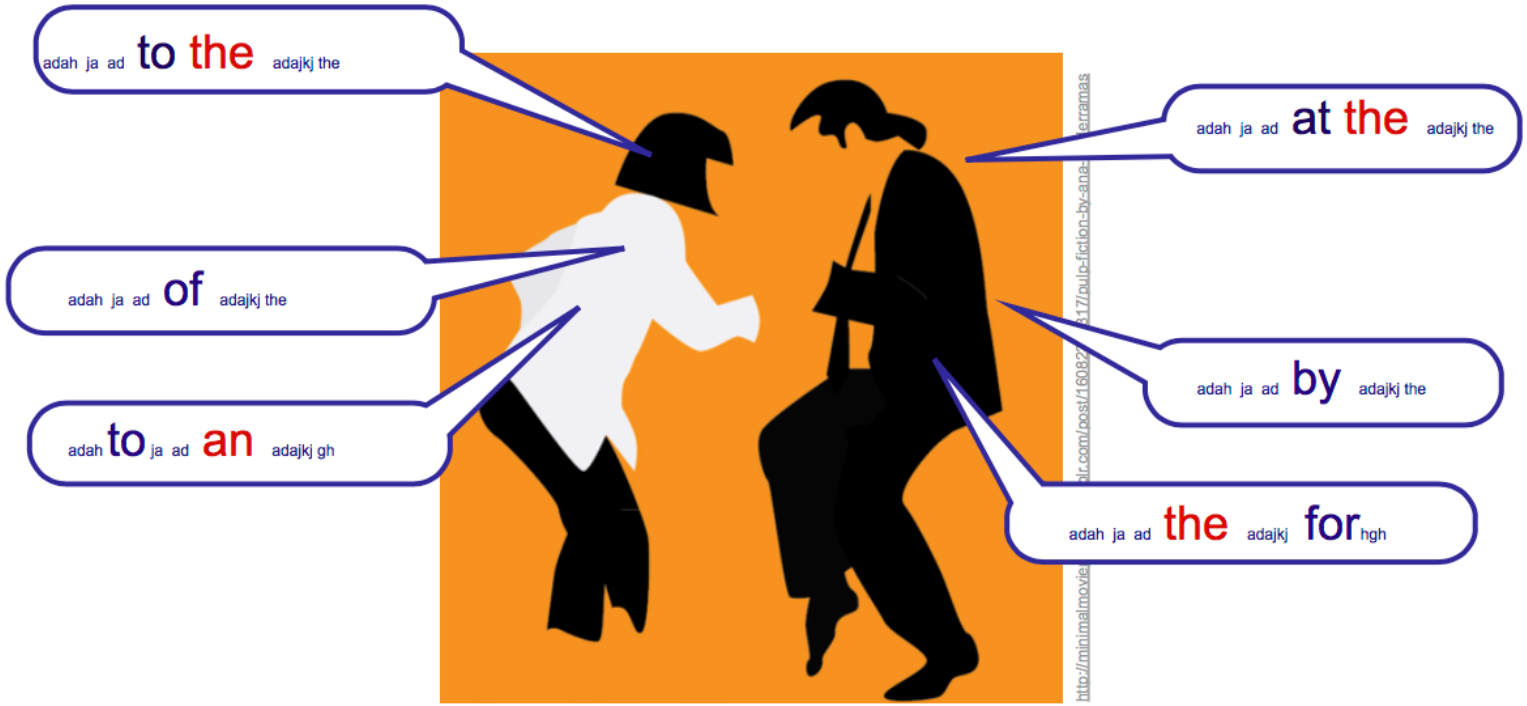


What should your learning algorithm look at?

Echoes of Power: Language Effects and Power Differences in Social Interaction. Danescu-Niculescu-Mizil et-al . WWW 2012.

Power Relations

Communicative behaviors are "patterned and coordinated, like a dance" [Niederhoffer and Pennebaker 2002]



Echoes of Power: Language Effects and Power Differences in Social Interaction. Danescu-Niculescu-Mizil et-al . WWW 2012.

Classification

- We assume we have a labeled dataset. **How can we build a classifier?**
- Decide on a **representation** and a **learning algorithm**
  - Essentially: **Function approximation**
    - **Representation:** What is the domain of the function
    - **Learning:** How to find a **good** approximation
  - We will look into several simple examples
    - Naïve Bayes, Perceptron
- Let's start with some definitions..

Basic Definitions

- Given: D a set of labeled examples {*<x,y>*}
- ***Goal**: Learn a function f(x) s.t. f(x) = y*
  - ***Note**: y can be binary, or categorical*
  - Typically *the input x is represented as a vector of **features***
- Break D into three parts:
  - **Training** set (used by the learning algorithm)
  - **Test** set (evaluate the learned model)
  - **Development** set (tuning the learning algorithm)
- **Evaluation**:
  - performance measure over the test set
  - **Accuracy**: proportion of correct predictions (test data)

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - ***Did we do a good job?***
- By simply saying "NO" to all examples we reduce the error by a factor of 10!
  - ***Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!***
- **Intuition**: we need a measure that captures the class we care about! (rare)

Precision and Recall

- The learner can make two kinds of mistakes:
  - False Positive
  - False Negative

	True Label: 1	True Label: 0
Predicted: 1	True Positive	False Positive
Predicted: 0	False Negative	True Negative

- **Precision:**
- *"when we predicted the rare class, how often are we right?"*

$$\frac{\text{True Pos}}{\text{Predicted Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos}}$$

- **Recall**
- "Out of all the instances of the rare class, how many did we catch?"

$$\frac{\text{True Pos}}{\text{Actual Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Neg}}$$

F-Score

- Precision and Recall give us two reference points to compare learning performance

	Precision	Recall
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1

- *Which algorithm is better?* *We need a single score*
- Option 1: **Average** $\frac{P+R}{2}$
- Option 2: **F-Score** $2\frac{PR}{P+R}$

Properties of f-score:

- Ranges between 0-1
- Prefers precision and recall with similar values

Simple Example: **Naïve Bayes**

- **Naïve Bayes**: simple probabilistic classifier
  - Given a set of labeled data:
    - Documents **D**, each associated with a label **v**
    - Simple feature representation: BoW
  - **Learning**:
    - Construct a probability distribution P(v|d)
  - **Prediction**:
    - Assign the label with the highest probability
- Relies on **strong** simplifying assumptions

Simple Representation: BoW

- Basic idea: (sentiment analysis)
  - "I loved this movie, it's awesome! I couldn't stop laughing for two hours!"
  - Mapping input to label can be done by *representing the frequencies of individual words*
  - **Document = word counts**

- ***Simple, yet surprisingly powerful representation!***

Bayes Rule

- Naïve Bayes is a simple probabilistic classification method, based on Bayes rule.

$$\mathbf{P(v \mid d) = P(d \mid v)^{P(v)/P(d)}}$$

Basics of Naïve Bayes

- P(v) - the **prior probability** of a label v
  Reflects background knowledge; before data is observed. If no information - uniform distribution.

- P(D) - The probability that this sample of the Data is observed. (*No knowledge of the label*)

- P(D|v): The probability of observing the sample D, given that the label v is the target (*Likelihood*)

- P(v|D): The **posterior probability** of v. The probability that v is the target, given that D has been observed.

Bayes Rule

- Naïve Bayes is a simple classification method, based on Bayes rule.

$$P(v \mid d) = P(d \mid v)^{P(v)/P(d)}$$

Check your intuition:

P(v|d) increases with P(v) and with P(d|v)

P(v|d) decreases with P(d)

Naïve Bayes

$$P(v|D) = P(D|v)\ P(v)/P(D)$$

- The learner considers a set of candidate labels, and attempts to find the most probable one $v \in V$, given the observed data.
- Such maximally probable assignment is called maximum a posteriori assignment (MAP); Bayes theorem is used to compute it:

$$v_{MAP} = \text{argmax}_{v \in V} P(v|D) = \text{argmax}_{v \in V} P(D|v)\ P(v)/P(D)$$

$$= \text{argmax}_{v \in V} P(D|v)\ P(v)$$

Since P(D) is the same for all $v \in V$

Naïve Bayes

- How can we compute P(v |D)?
  - Basic idea: represent document as a set of features, such as BoW features

$$v_{MAP} = \text{argmax}_{v_j \in V} P(v_j \mid x) = \text{argmax}_{v_j \in V} P(v_j \mid x_1, x_2, \ldots, x_n)$$

$$v_{MAP} = \text{argmax}_{v_j \in V} \frac{P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \text{argmax}_{v_j \in V} P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)$$

NB: Parameter Estimation

$$V_{MAP} = \text{argmax}_v \, P(x_1, x_2, \ldots, x_n \mid v)P(v)$$

- Given training data we can estimate the two terms
- Estimating $P(v)$ is **easy**. For each value v count how many times it appears in the training data.

***Question**: Assume binary x_i's. How many parameters does the model require?*

- However, it is not feasible to estimate $P(x_1,\ldots, x_n \mid v)$
  - In this case we have to estimate, for each target value, the probability of each instance (most of which will not occur)
- In order to use a Bayesian classifiers in practice, we *need to make assumptions* that will allow us to estimate these quantities.

NB: **Independence Assumption**

- **Bag of words representation**:
  - Word position can be ignored
- **Conditional Independence**: Assume feature probabilities are independent given the label
  - $P(x_i|v_j) = P(x_i|x_{i-1}; v_j)$
- Both assumptions are **not true**
  - Help simplify the model
  - *Simple models work well*

how many features are needed now?

Naive Bayes

$$V_{MAP} = \text{argmax}_v \, P(x_1, x_2, \ldots, x_n \mid v)P(v)$$

$$\begin{aligned}
&P(x_1, x_2, \ldots, x_n \mid v_j) = \\
&= P(x_1 \mid x_2, \ldots, x_n, v_j)P(x_2, \ldots, x_n \mid v_j) \\
&= P(x_1 \mid x_2, \ldots, x_n, v_j)P(x_2 \mid x_3, \ldots, x_n, v_j)P(x_3, \ldots, x_n \mid v_j) \\
&= \ldots\ldots\ldots \\
&= P(x_1 \mid x_2, \ldots, x_n, v_j)P(x_2 \mid x_3, \ldots, x_n, v_j)P(x_3 \mid x_4, \ldots, x_n, v_j)\ldots P(x_n \mid v_j)
\end{aligned}$$

Assumption: feature values are independent given the target value

$$= \prod_{i=1}^{n} P(x_i \mid v_j)$$

Estimating Probabilities (MLE)

Assume a document classification problem, using word features

$$v_{NB} = \text{argmax}_{v \in \{like, dislike\}} P(v) \prod_i P(x_i = word_i \mid v)$$

How do we estimate $P(word_k \mid v)$*?*

$$P(word_k \mid v) = \frac{\#(word_k \text{ appears in training in v documents})}{\#(v \text{ documents})} = \frac{n_k}{n}$$

Sparsity of data is a problem

-- if n is small, the estimate is not accurate

-- if n_k is 0, it will dominate the estimate: we will never predict v if a word that never appeared in training (with v) appears in the test data

Robust Estimation of Probabilities

$$v_{NB} = \text{argmax}_{v \in \{like, dislike\}} P(v) \prod_i P(x_i = word_i \mid v)$$

- **This process is called smoothing.**
- There are many ways to do it, some better justified than others;
- An empirical issue.

$$P(x_k \mid v) = \frac{n_k + mp}{n + m}$$

- **Here:**
  - n_k is ***#(of occurrences of the word in the presence of v)***
  - n *is* ***#(of occurrences of the label v)***
  - p is a ***prior estimate of v*** *(e.g., uniform)*
  - m is ***equivalent sample size*** *(# of labels)*
- **Laplace Rule**: for the Boolean case, p=1/2 , m=2

$$P(x_k \mid v) = \frac{n_k + 1}{n + 2}$$

Naïve Bayes

- **Very easy to implement**
- **Converges very quickly**
  - Learning is just counting
- **Performs well in practice**
  - Applied to many document classification tasks
  - If data set is small, NB can perform better than sophisticated algorithms
- **Strong independence assumptions**
  - If assumptions hold: **NB is the optimal classifier**
  - Even if not, can perform well
- **Next**: *from NB to learning linear threshold functions*

Naïve Bayes: Two Classes

- *Notice that the naïve Bayes method gives a method for predicting rather than an explicit classifier*
- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 1)}{P(v_j = 0) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 0)} > 1$$

Naïve Bayes: Two Classes

- Notice that the naïve Bayes method gives a method for predicting rather than an explicit classifier.
- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 1)}{P(v_j = 0) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 0)} > 1$$

Denote: $p_i = P(x_i = 1 \mid v = 1), \quad q_i = P(x_i = 1 \mid v = 0)$

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1 - q_i)^{1-x_i}} > 1$$

Naïve Bayes: Two Classes

In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1-p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1-q_i)^{1-x_i}} = \frac{P(v_j = 1) \bullet \prod_{i=1}^{n} (1-p_i)\left(\frac{p_i}{1-p_i}\right)^{x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} (1-q_i)\left(\frac{q_i}{1-q_i}\right)^{x_i}} > 1$$

Naïve Bayes: Two Classes

In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1-p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1-q_i)^{1-x_i}} = \frac{P(v_j = 1) \bullet \prod_{i=1}^{n} (1-p_i)(\frac{p_i}{1-p_i})^{x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} (1-q_i)(\frac{q_i}{1-q_i})^{x_i}} > 1$$

Take logarithm; we predict $v = 1$ iff :

$$\log \frac{P(v_j = 1)}{P(v_j = 0)} + \sum_i \log \frac{1-p_i}{1-q_i} + \sum_i (\log \frac{p_i}{1-p_i} - \log \frac{q_i}{1-q_i}) x_i > 0$$

Naïve Bayes: Two Classes

In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j=1)\bullet\prod_{i=1}^{n} p_i^{x_i}(1-p_i)^{1-x_i}}{P(v_j=0)\bullet\prod_{i=1}^{n} q_i^{x_i}(1-q_i)^{1-x_i}} = \frac{P(v_j=1)\bullet\prod_{i=1}^{n}(1-p_i)\left(\frac{p_i}{1-p_i}\right)^{x_i}}{P(v_j=0)\bullet\prod_{i=1}^{n}(1-q_i)\left(\frac{q_i}{1-q_i}\right)^{x_i}} > 1$$

Take logarithm; we predict $v=1$ iff :

$$\log\frac{P(v_j=1)}{P(v_j=0)} + \sum_i \log\frac{1-p_i}{1-q_i} + \sum_i \left(\log\frac{p_i}{1-p_i} - \log\frac{q_i}{1-q_i}\right)x_i > 0$$

- **We get that naive Bayes is a linear separator with :**

$$w_i = \log\frac{p_i}{1-p_i} - \log\frac{q_i}{1-q_i} = \log\frac{p_i}{q_i}\frac{1-q_i}{1-p_i}$$

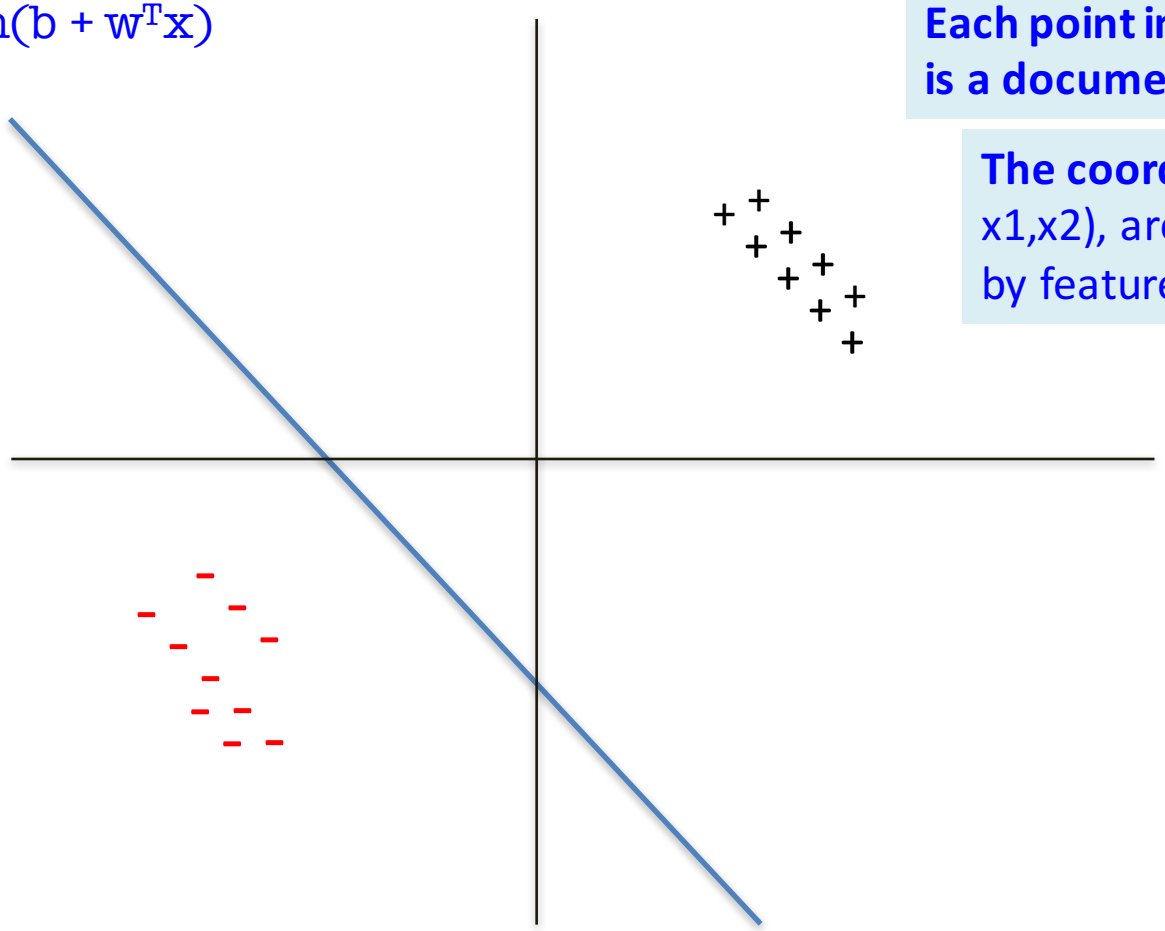
if $p_i = q_i$ then $w_i = 0$ and the feature is irrelevant

Linear Classifiers

- Linear threshold functions
  - Associate a weight (w_i) with each feature (x_i)
  - **Prediction**: $\text{sign}(b + w^T x) = \text{sign}(b + \Sigma\, w_i\, x_i)$
    - $b + w^T x \geq 0$ predict y=1
    - Otherwise, predict y=-1
- *NB is a linear threshold function*
  - Weight vector (w) is assigned by computing conditional probabilities
- *In fact, Linear threshold functions are a very popular representation!*

Linear Classifiers

$\text{sign}(b + w^T x)$

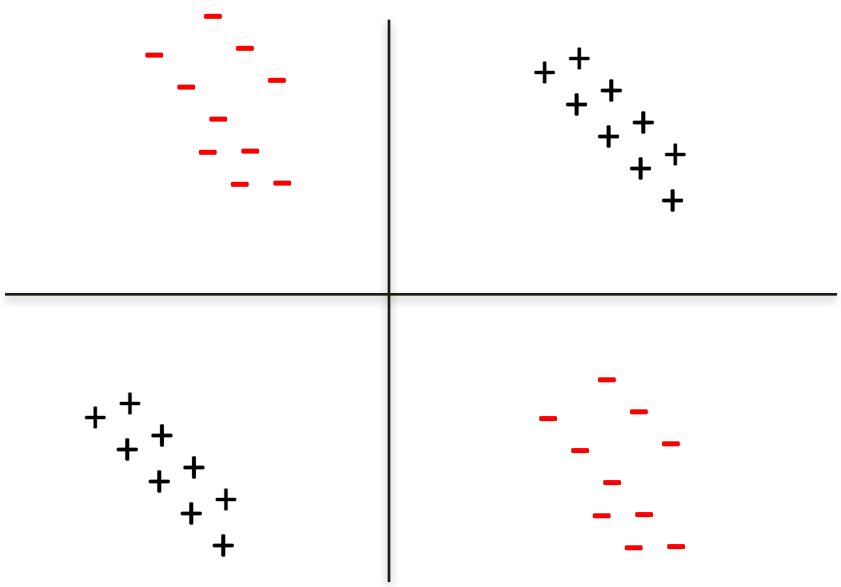


Each point in this space is a document.

The coordinates (e.g., x1,x2), are determined by feature activations

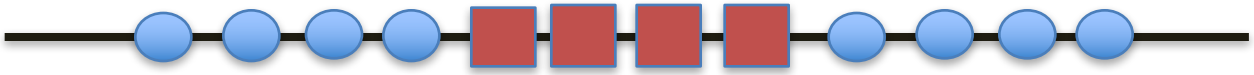
Expressivity

- Linear functions are quite expressive
  - *Exists a linear function that is consistent with the data*
- A famous negative examples (XOR):

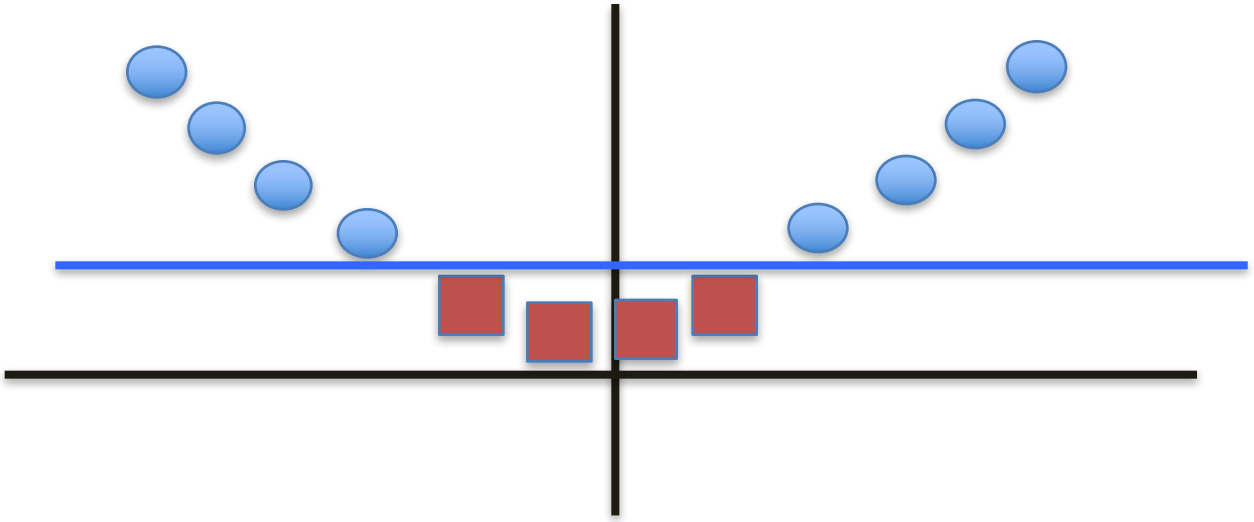


Expressivity

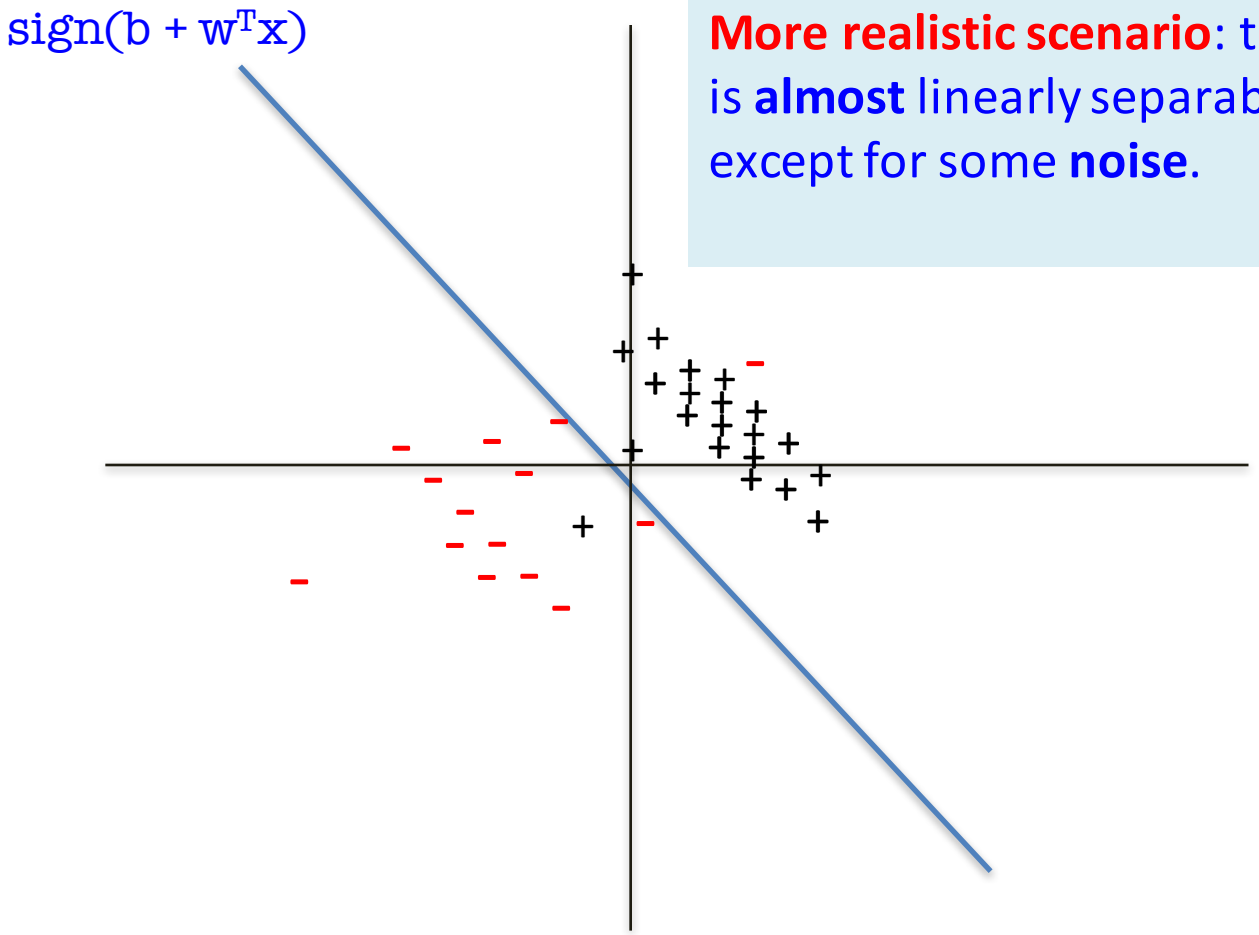
By transforming the feature space these functions can be made linear



Represent each point in 2D as (x,x^2)



Expressivity



More realistic scenario: the data is **almost** linearly separable, except for some **noise**.

Features

- So far we have discussed BoW representation
  - *In fact, you can use a very rich representation*
- **Broader definition**
  - Functions mapping attributes of the input to a Boolean/categorical/numeric value

$$\phi_1(x) = \begin{cases} 1 & x_1 \ is \ capitalized \\ 0 & otherwise \end{cases}$$

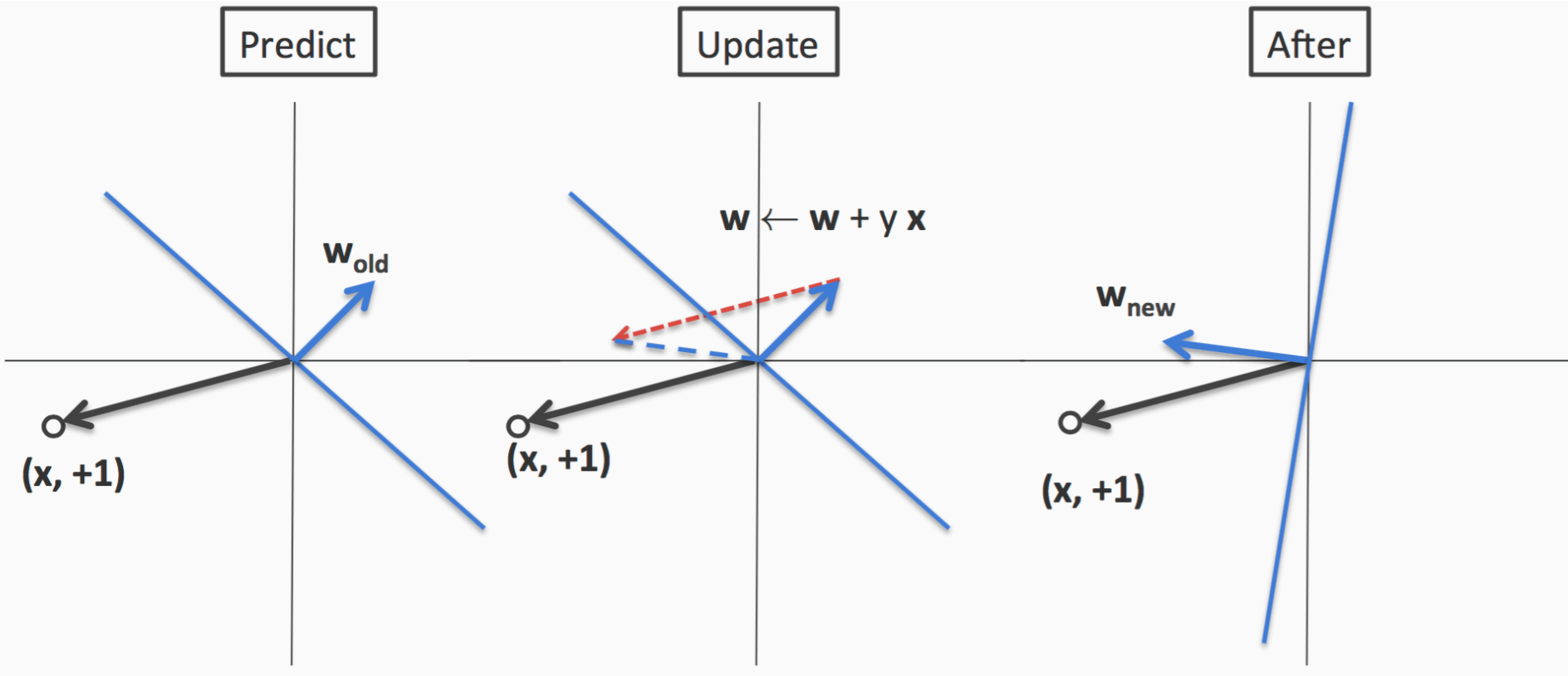
$$\phi_k(x) = \begin{cases} 1 & x \ contains \ "good" \ more \ than \ twice \\ 0 & otherwise \end{cases}$$

- **Question**: *assume that you have a lexicon, containing positive and negative sentiment words. How can you use it to improve over BoW?*

Perceptron

- One of the earliest learning algorithms
  - Introduced by Rosenblatt 1958 to model neural learning
- **Goal**: directly search for a separating hyperplane
  - If one exists, perceptron will find it
  - If not, …
- **Online** algorithm
  - Considers one example at a time (NB – looks at entire data)
- **Error driven** algorithm
  - Updates the weights only when a mistake is made

Perceptron Intuition



Perceptron

- We learn f:X→ {-1,+1} represented as f =sgn{w•x)
- Where X= $\{0,1\}^n$ or X= R^n and w ^{2} R^n
- Given Labeled examples: $\{(x_1, y_1), (x_2, y_2),\ldots(x_m, y_m)\}$

1. Initialize w=0 $\in \mathbf{R}^n$
2. Cycle through all examples

   a. **Predict** the label of instance x to be y' = sgn{w•x)

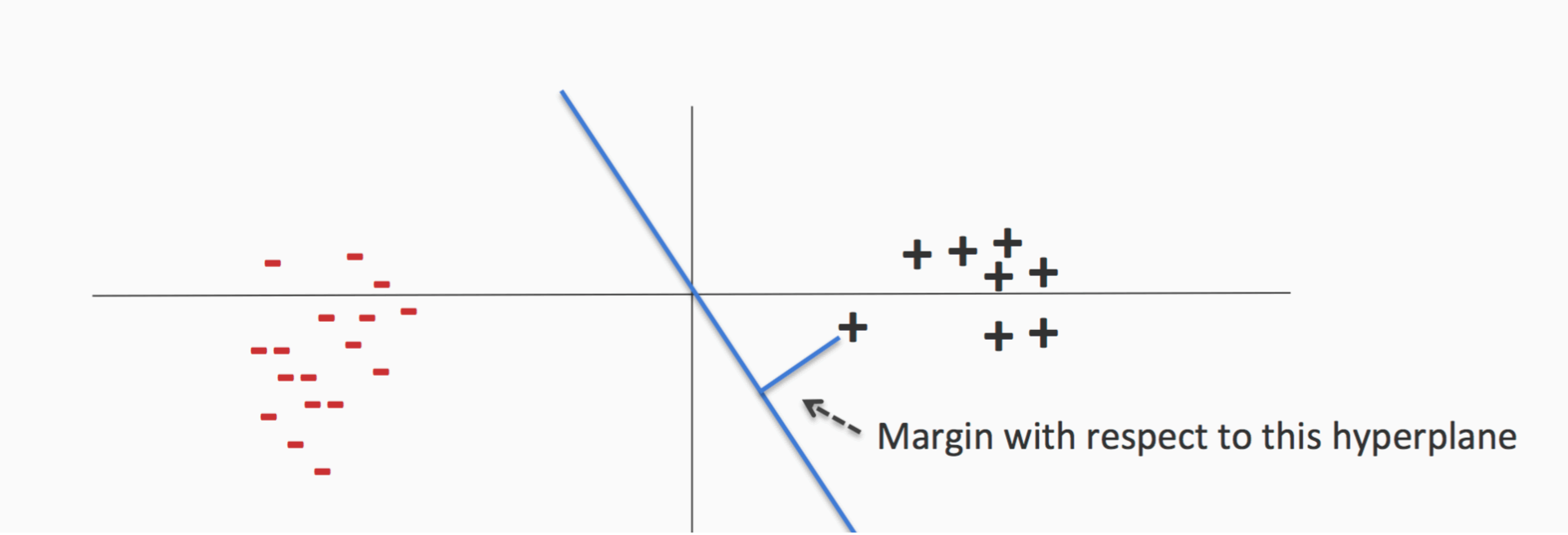
   b. If ***y'≠y***, **update** the weight vector:

   w = w + r y x (r - a constant, learning rate)

   Otherwise, if **y'=y**, leave weights unchanged.

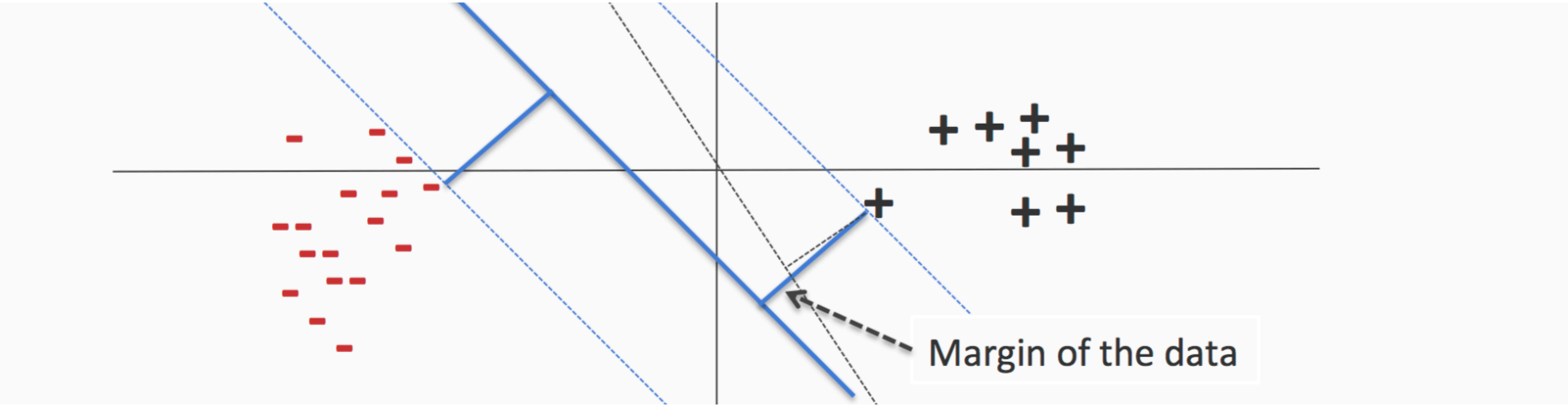
Margin

- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Margin

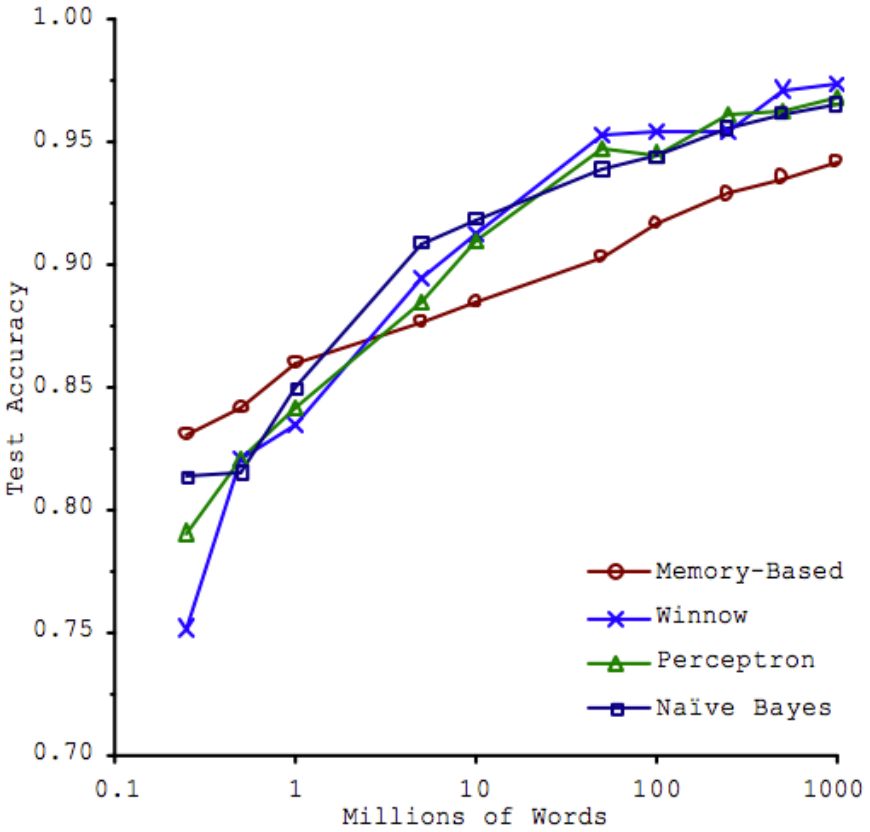
- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.



Mistake Bound for Perceptron

- Let $D=\{(x_i, y_i)\}$ be a labeled dataset that is separable
- Let $||x_i|| < R$ for all examples.
- Let γ be the margin of the dataset D.
- Then, the perceptron algorithm will make at most R^2/γ^2 mistakes on the data.

Practical Example



Task: **context sensitive spelling**
{principle, principal},{weather,whether}.

Source: Scaling to very very large corpora for natural language disambiguation
Michele Banko, Eric Brill. Microsoft Research, Redmond, WA. 2001.

Deceptive Reviews

Which of these two hotel reviews is deceptive opinion spam?

My husband and I stayed at the James Chicago Hotel for our anniversary. This place is fantastic! We knew as soon as we arrived we made the right choice! The rooms are BEAUTIFUL and the staff very attentive and wonderful!! The area of the hotel is great, since I love to shop I couldn't ask for more!! We will definatly be back to Chicago and we will for sure be back to the James Chicago.

I have stayed at many hotels traveling for both business and pleasure and I can honestly stay that The James is tops. The service at the hotel is first class. The rooms are modern and very comfortable. The location is perfect within walking distance to all of the great sights and restaurants. Highly recommend to both business travellers and couples.

What should your learning algorithm look at?

Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Ott etal. ACL 2011

Deception Classification

			TRUTHFUL			DECEPTIVE		
Approach	**Features**	**Accuracy**	**P**	**R**	**F**	**P**	**R**	**F**
GENRE IDENTIFICATION	POS_{SVM}	73.0%	75.3	68.5	71.7	71.1	77.5	74.2
PSYCHOLINGUISTIC DECEPTION DETECTION	$LIWC_{SVM}$	76.8%	77.2	76.0	76.6	76.4	77.5	76.9
TEXT CATEGORIZATION	$UNIGRAMS_{SVM}$	88.4%	89.9	86.5	88.2	87.0	90.3	88.6
	$BIGRAMS^{+}_{SVM}$	89.6%	90.1	89.0	89.6	89.1	90.3	89.7
	$LIWC{+}BIGRAMS^{+}_{SVM}$	**89.8%**	89.8	**89.8**	**89.8**	**89.8**	89.8	**89.8**
	$TRIGRAMS^{+}_{SVM}$	89.0%	89.0	89.0	89.0	89.0	89.0	89.0
	$UNIGRAMS_{NB}$	88.4%	**92.5**	83.5	87.8	85.0	**93.3**	88.9
	$BIGRAMS^{+}_{NB}$	88.9%	89.8	87.8	88.7	88.0	90.0	89.0
	$TRIGRAMS^{+}_{NB}$	87.6%	87.7	87.5	87.6	87.5	87.8	87.6
HUMAN / META	JUDGE 1	**61.9%**	57.9	87.5	**69.7**	74.4	36.3	48.7
	JUDGE 2	56.9%	53.9	**95.0**	68.8	**78.9**	18.8	30.3
	SKEPTIC	60.6%	**60.8**	60.0	60.4	60.5	**61.3**	**60.9**

Summary

- Classification is a basic tool for NLP
  - *E.g., What is the topic of a document?*
- *Classifier:* ***mapping from input to label***
  - *Label: Binary or Categorical*
- We saw two simple learning algorithms for finding the parameters of linear classification functions
  - Naïve Bayes and Perceptron
- **Next**:
  - More sophisticated algorithms
  - Applications (or – *how to get it to work!*)

Questions?