Machine Learning

Hal Daumé III

Computer Science
University of Maryland

me@hal3.name

CS 421: Introduction to Artificial Intelligence

8 May 2012

Many slides courtesy of
Dan Klein, Stuart Russell,
or Andrew Moore

Announcements

- Final exam review
  - Please vote for a time slot:
    - http://www.when2meet.com/?439876-EJD1w
  - Will decide by class Thursday!

Machine Learning

- Up until now: how to reason in a model and how to make optimal decisions
- Machine learning: how to select a model on the basis of data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering)

Example: Spam Filter

- Input: email
- Output: spam/ham
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham”
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails
- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images
- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …


0


1


2


1


??

Other Classification Tasks

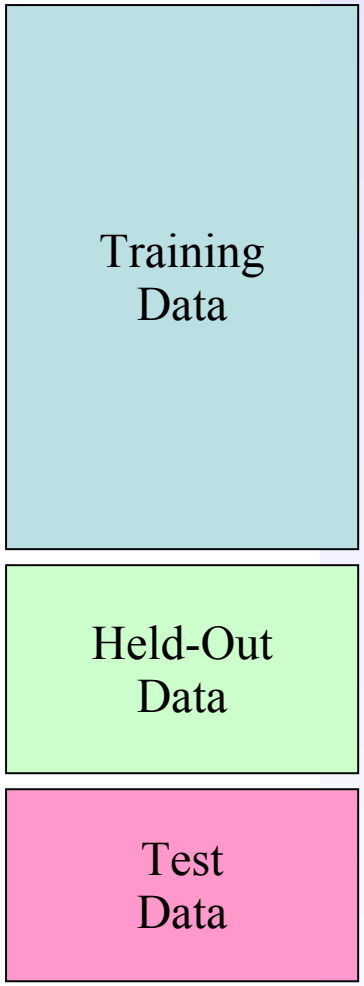
- In classification, we predict labels y (classes) for inputs x
- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - … many more
- Classification is an important commercial technology!

Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!

- Evaluation
  - Accuracy: fraction of instances predicted correctly

- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well



Bayes Nets for Classification

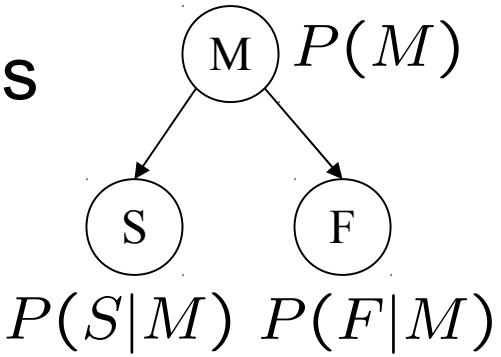
- One method of classification:
  - Use a probabilistic model!
  - Features are observed random variables F_i
  - Y is the query variable
  - Use probabilistic inference to compute most likely Y

$$y = \text{argmax}_y \; P(y|f_1 \ldots f_n)$$

- You already know how to do this inference

Simple Classification

- Simple example: two binary features



$P(m|s,f)$ ← direct estimate

$$P(m|s,f) = \frac{P(s,f|m)P(m)}{P(s,f)}$$ ← Bayes estimate (no assumptions)

$$P(m|s,f) = \frac{P(s|m)P(f|m)P(m)}{P(s,f)}$$ ← Conditional independence

$$+ \begin{cases} P(m,s,f) = P(s|m)P(f|m)P(m) \\ P(\bar{m},s,f) = P(s|\bar{m})P(f|\bar{m})P(\bar{m}) \end{cases}$$

General Naïve Bayes

- A general *naive Bayes* model:

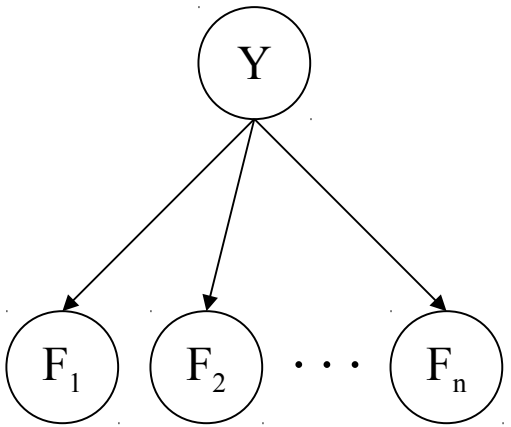
$|Y| \times |F|^n$ parameters

$$P(Y, F_1 \ldots F_n) =$$

$$P(Y) \prod_i P(F_i|Y)$$

$|Y|$ parameters

$n \times |F| \times |Y|$ parameters



- We only specify how each feature depends on the class
- Total number of parameters is *linear* in n

Inference for Naïve Bayes

- Goal: compute posterior over causes
  - Step 1: get joint probability of causes and evidence

$$P(Y, f_1 \dots f_n) =$$

$$\begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(f_1) \prod_i P(f_i|c_1) \\ P(f_2) \prod_i P(f_i|c_2) \\ \vdots \\ P(f_k) \prod_i P(f_i|c_k) \end{bmatrix}$$

  - Step 2: get probability of evidence

$$\frac{\begin{bmatrix} P(f_1) \prod_i P(f_i|c_1) \\ P(f_2) \prod_i P(f_i|c_2) \\ \vdots \\ P(f_k) \prod_i P(f_i|c_k) \end{bmatrix}}{P(f_1 \dots f_n)} \quad +$$

  - Step 3: renormalize

$$\Downarrow$$

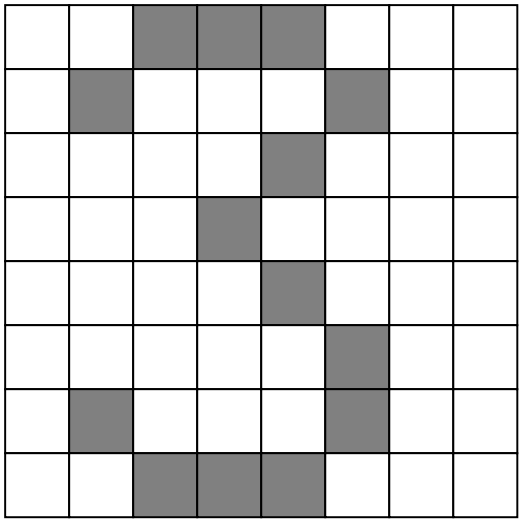
$$P(Y|f_1 \dots f_n)$$

General Naïve Bayes

- What do we need in order to use naïve Bayes?
  - Inference (you know this part)
    - Start with a bunch of conditionals, P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the *parameters* of the model and denoted by $\boldsymbol{\theta}$
    - Up until now, we assumed these appeared by magic, but…
    - …they typically come from training data: we'll look at this now

A Digit Recognizer

- Input: pixel grids



- Output: a digit 0-9

Naïve Bayes for Digits

- Simple version:
  - One feature F_{ij} for each grid position <i,j>
  - Possible feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

  - Here: lots of features, each is binary
- Naïve Bayes model:

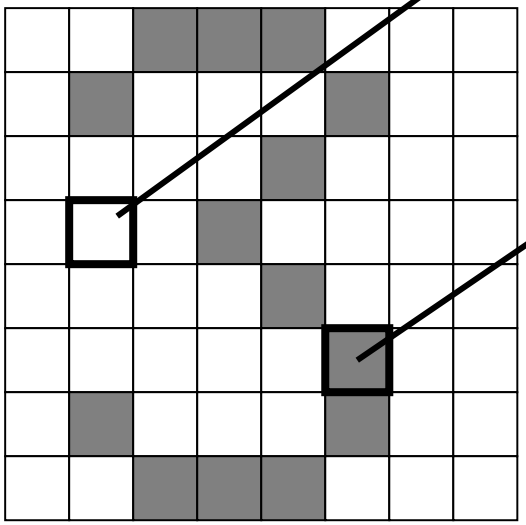
$$P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

- What do we need to learn?

Examples: CPTs

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Parameter Estimation

- Estimating distribution of random variables like X or X | Y
- *Empirically:* use training data
  - For each outcome x, look at the *empirical rate* of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{\text{ML}}(\text{r}) = 1/3$$

  - This is the estimate that maximizes the *likelihood of the data*

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

- *Elicitation:* ask a human!
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating

A Spam Filter

- Naïve Bayes spam filter
- Data:
  - Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- Classifiers
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Naïve Bayes for Text

- Bag-of-Words Naïve Bayes:
  - Predict unknown class label (spam vs. ham)
  - Assume evidence features (e.g. the words) are independent
  - Warning: subtly different assumptions than before!
- Generative model

$$P(C, W_1 \dots W_n) = P(Y) \prod_i P(W_i|C)$$

Word at position i, not ith word in the dictionary!

- Tied distributions and bag-of-words
  - Usually, each variable gets its own conditional probability distribution P(F|Y)
  - In a bag-of-words model
    - Each position is identically distributed
    - All positions share the same conditional probs P(W|C)
    - Why make this assumption?

Example: Spam Filtering

- Model:

$$P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$$

- What are the parameters?

$P(C)$

```
ham : 0.66
spam: 0.33
```

$P(W|\text{spam})$

```
the :  0.0156
to  :  0.0153
and :  0.0115
of  :  0.0095
you :  0.0093
a   :  0.0086
with:  0.0080
from:  0.0075
...
```

$P(W|\text{ham})$

```
the :  0.0210
to  :  0.0133
of  :  0.0119
2002:  0.0110
with:  0.0108
from:  0.0107
and :  0.0105
a   :  0.0100
...
```

- Where do these tables come from?

Spam Example

Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4

P(spam | w) = 98.9

Example: Overfitting

$P(\text{features}, C = 2)$

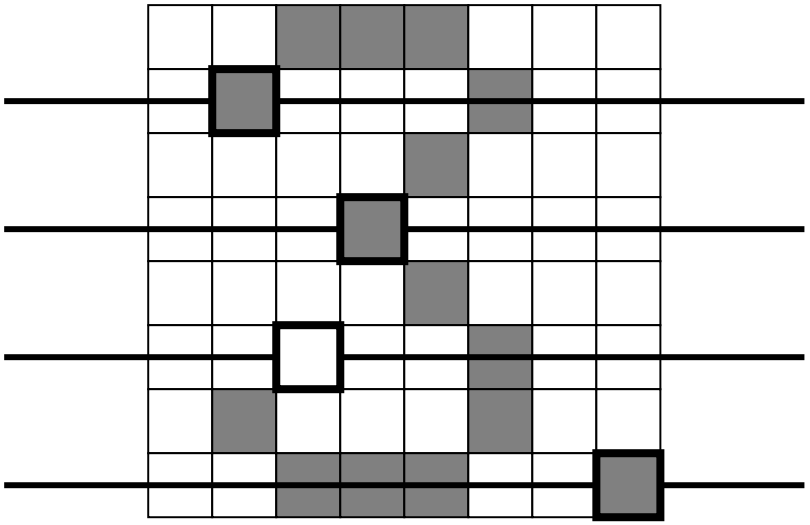
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

2 wins!!

Example: Spam Filtering

- Raw probabilities alone don't affect the posteriors; relative probabilities (odds ratios) do:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

What went wrong here?

Generalization and Overfitting

- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all?
  - In general, we can't go around giving unseen events zero probability

- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough

- To generalize better: we need to smooth or regularize the estimates

Estimation: Smoothing

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it's heads, what's the estimate for P(heads)?
  - What if I flip 10 times with 8 heads?
  - What if I flip 10M times with 8M heads?

- Basic idea:
  - We have some prior expectation about parameters (here, the probability of heads)
  - Given little evidence, we should skew towards our prior
  - Given a lot of evidence, we should listen to the data

Estimation: Smoothing

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_i P_\theta(X_i)$$

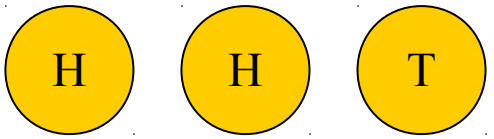
$$\Rightarrow \quad P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

- In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \quad \Rightarrow \quad ????$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Estimation: Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did



$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

  - Can derive this as a MAP estimate with *Dirichlet priors* (see cs5350)

Estimation: Laplace Smoothing

- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large
- Another option: linear interpolation
  - Also get P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from P(X)

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha)\hat{P}(x)$$

  - What if α is 0? 1?

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

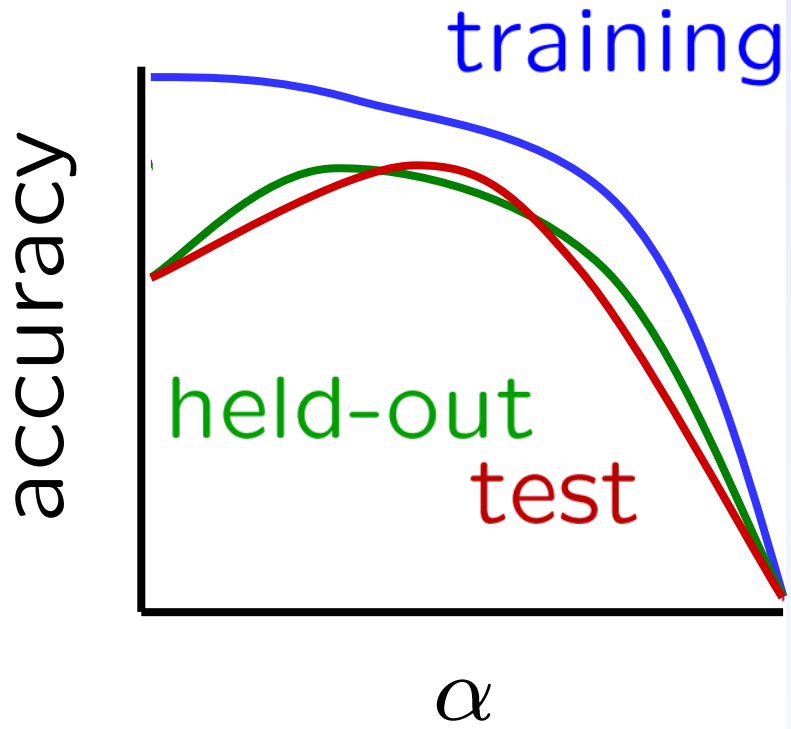
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```

Do these make more sense?

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(Y|X), P(Y)
  - Hyperparameters, like the amount of smoothing to do: k, α
- Where to learn?
  - Learn parameters from training data
  - Must tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data



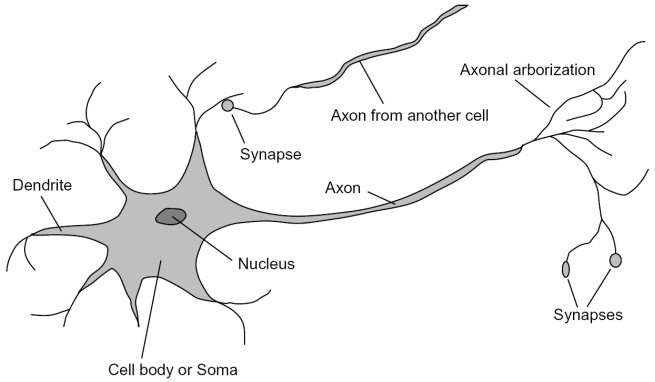
Generative vs. Discriminative

- Generative classifiers:
  - E.g. naïve Bayes
  - We build a causal model of the variables
  - We then query that model for causes, given evidence

- Discriminative classifiers:
  - E.g. perceptron (next)
  - No causal model, no Bayes rule, often no probabilities
  - Try to predict output directly
  - Loosely: mistake driven rather than model driven

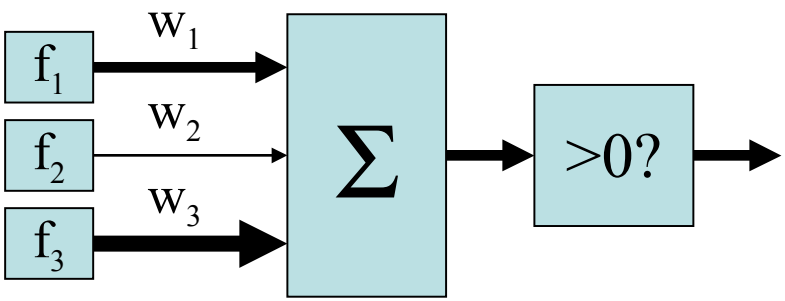
The Binary Perceptron

- Inputs are feature values
- 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)

$$w \cdot f(x)$$

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
...	

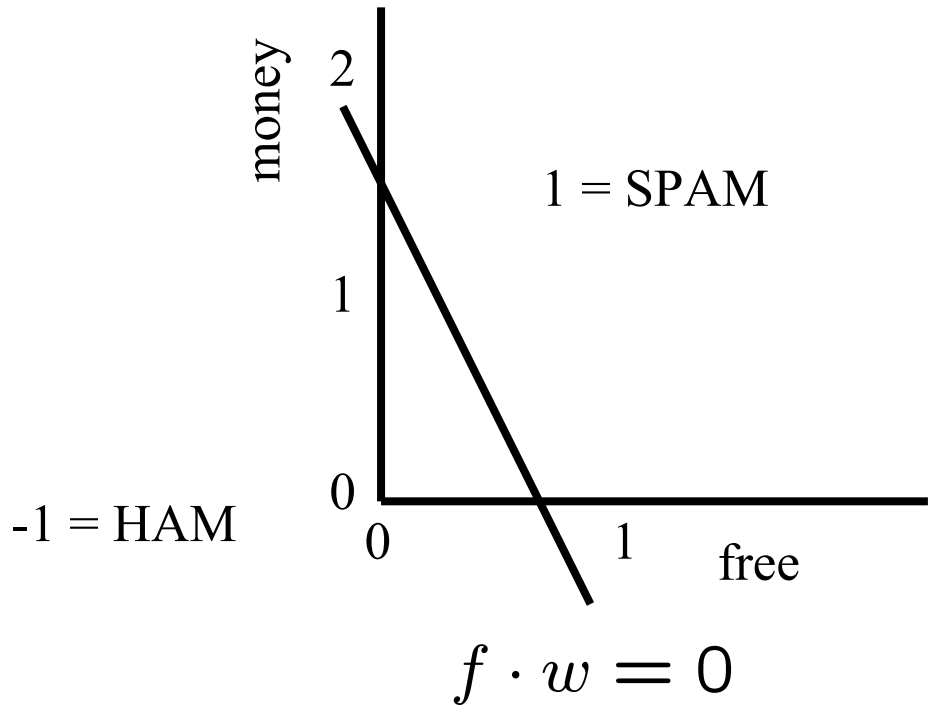
$$(1)(-3) + (1)(4) + (1)(2) + (0)(0) + \ldots = 3$$

Binary Decision Rule

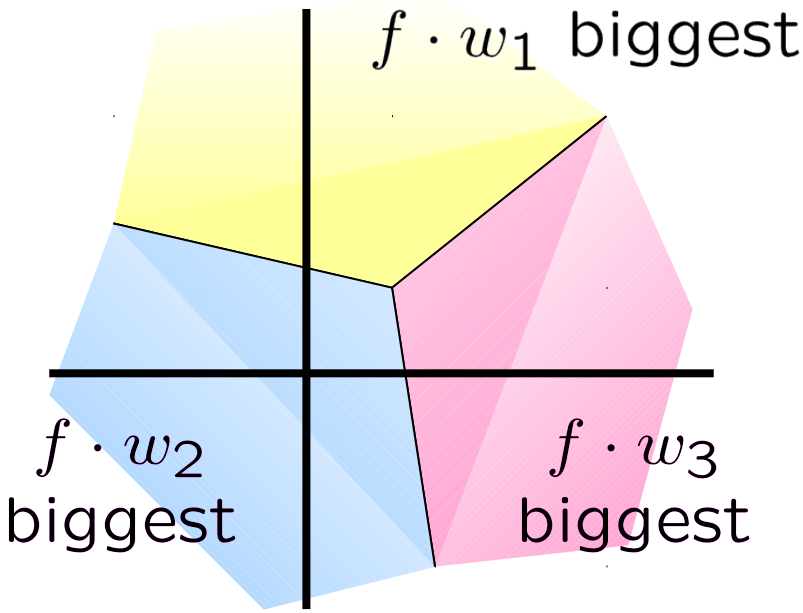
- In the space of feature vectors
  - Any weight vector is a hyperplane
  - One side will be class 1
  - Other will be class -1

w

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



Multiclass Decision Rule



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

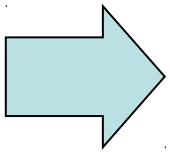
$$\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)$$

- Highest activation wins

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

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
...
```

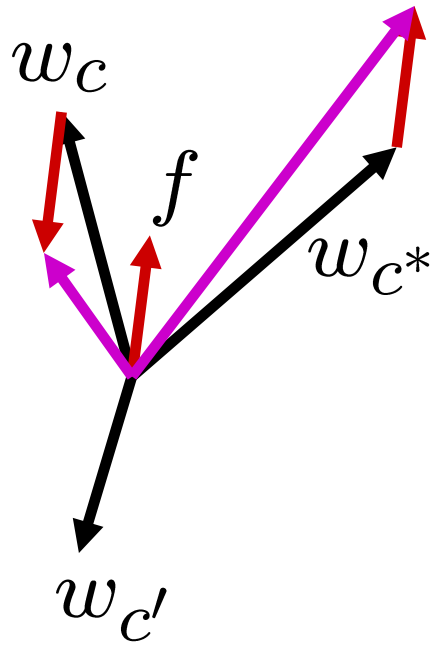
The Perceptron Update Rule

- 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



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

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

Example

"win the vote"

"win the election"

"win the game"

w_{SPORTS}

```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

$w_{POLITICS}$

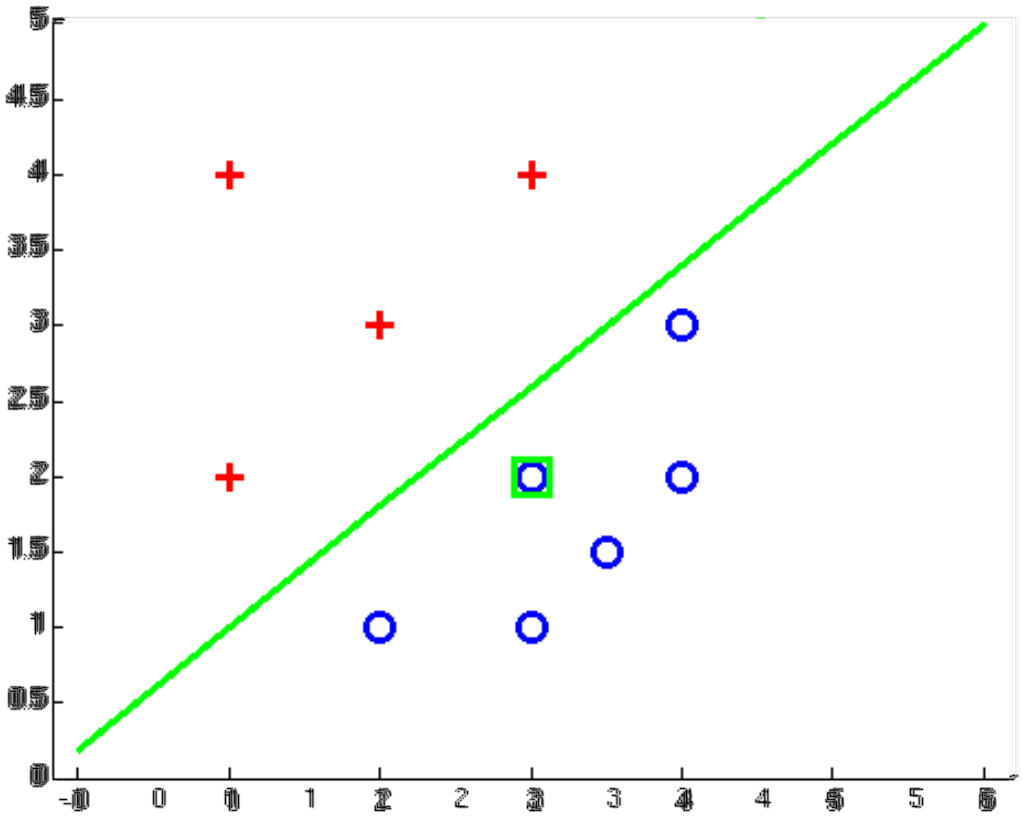
```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

w_{TECH}

```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

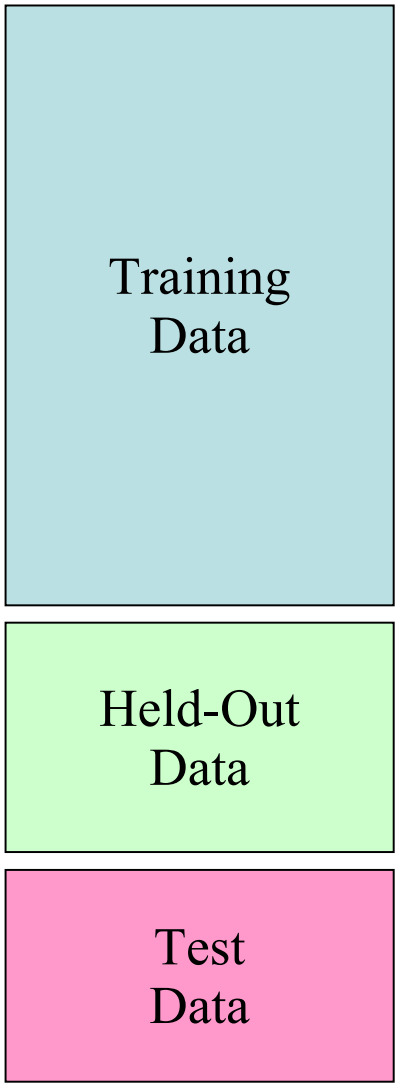
Examples: Perceptron

- Separable Case



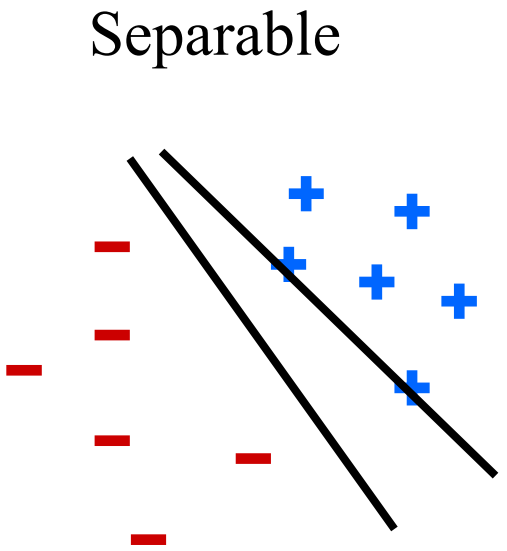
Mistake-Driven Classification

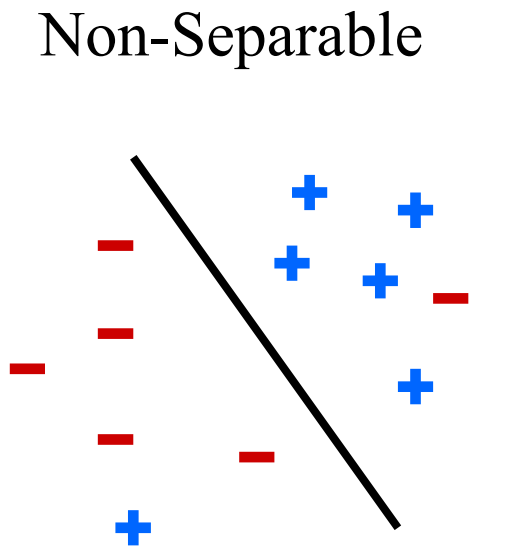
- In naïve Bayes, parameters:
  - From data statistics
  - Have a causal interpretation
  - One pass through the data
- For the perceptron parameters:
  - From reactions to mistakes
  - Have a discriminative interpretation
  - Go through the data until held-out accuracy maxes out



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

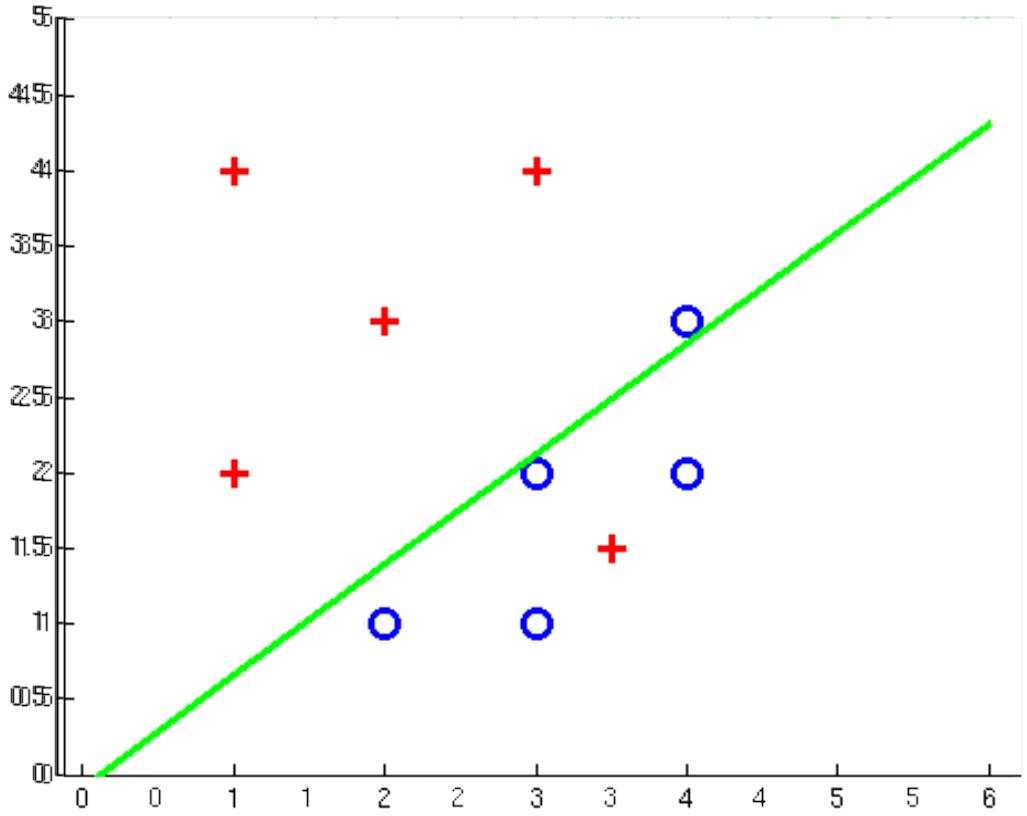




$$\text{mistakes} < \frac{1}{\delta^2}$$

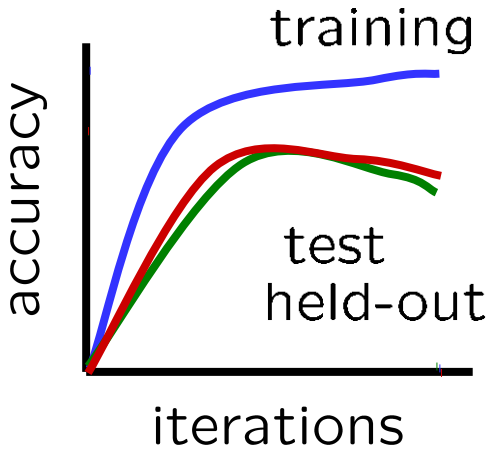
Examples: Perceptron

- Non-Separable Case

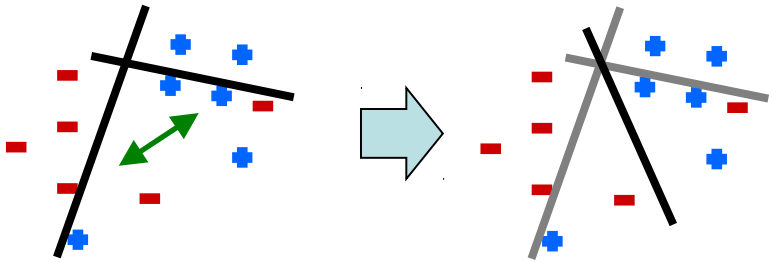


Issues with Perceptrons

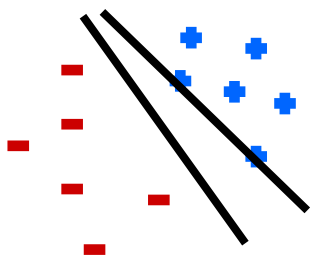
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn't quite as bad as overfitting, but is similar



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

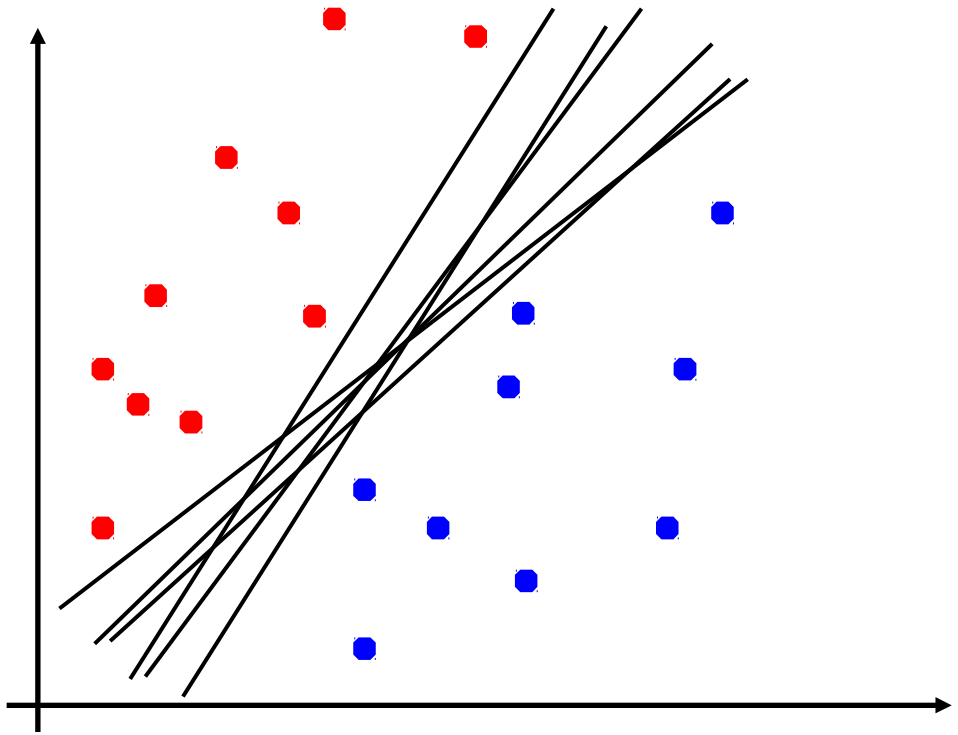


- Mediocre generalization: finds a "barely" separating solution



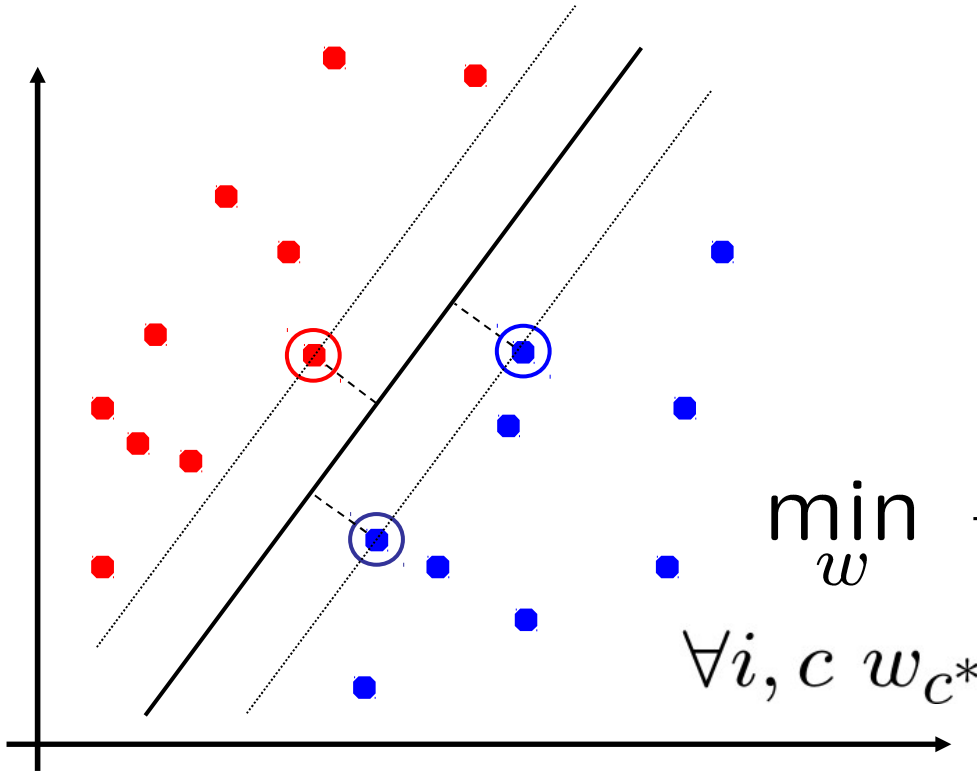
Linear Separators

- Which of these linear separators is optimal?



Support Vector Machines

- Maximizing the margin: good according to intuition and theory.
- Only support vectors matter; other training examples are ignorable.
- Support vector machines (SVMs) find the separator with max margin



$$\min_{w} \frac{1}{2}||w||^2$$

$$\forall i, c \; w_{c^*} \cdot f(x_i) \geq w_c \cdot f(x_i) + 1$$

Summary

- Naïve Bayes
  - Build classifiers using model of training data
  - Smoothing estimates is important in real systems
- Perceptrons:
  - Make less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data