Supervised Classification

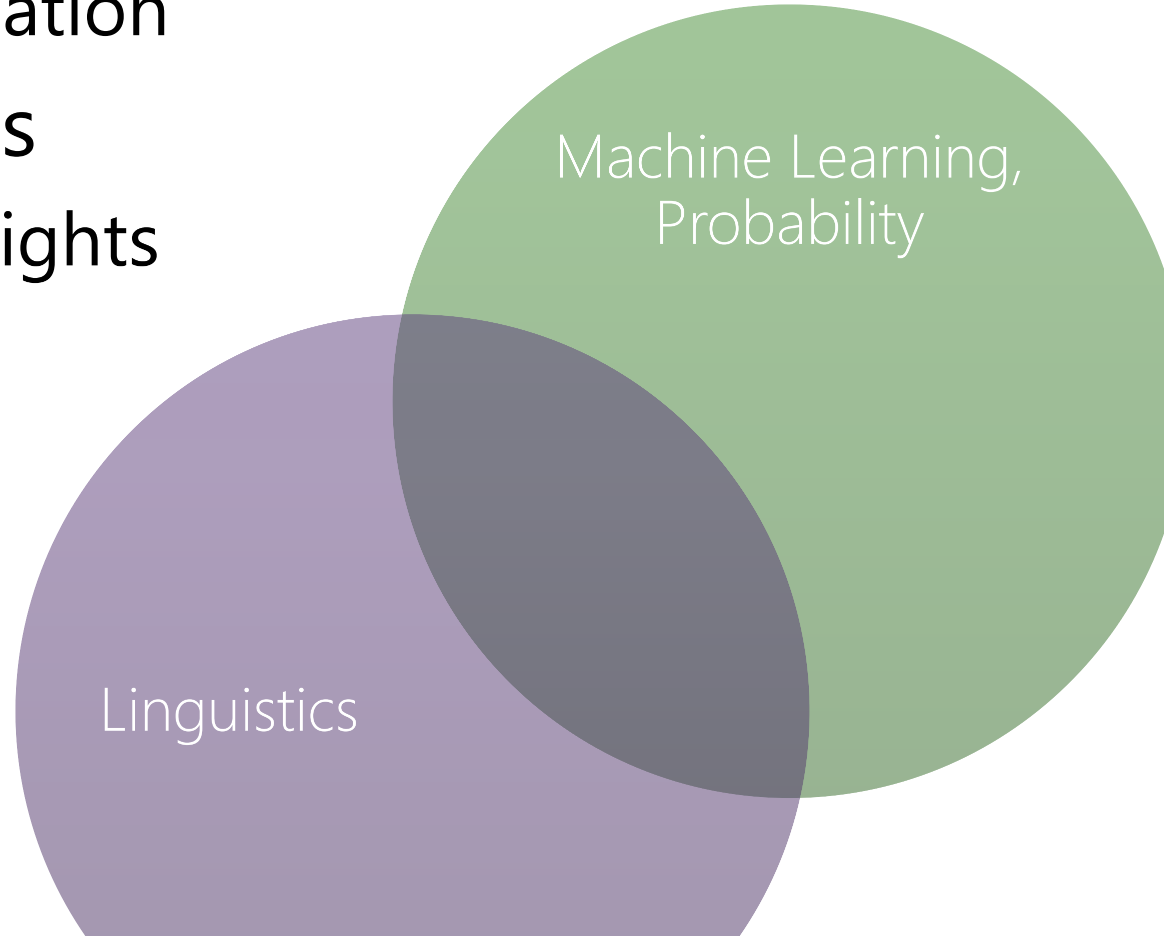
CMSC 723 / LING 723 / INST 725

MARINE CARPUAT
marine@cs.umd.edu

Some slides by Graham Neubig , Jacob Eisenstein

Last time

- Text classification problems
  - and their evaluation
- Linear classifiers
  - Features & Weights
  - Bag of words
  - Naïve Bayes



Today

- 3 linear classifiers
  - Naïve Bayes
  - Perceptron
  - (Logistic Regression)

- Bag of words vs. rich feature sets
- Generative vs. discriminative models
- Bias-variance tradeoff

Naïve Bayes Recap

- Define $\mathrm{p}(\boldsymbol{x}, \boldsymbol{y})$ via a *generative model*
- Prediction: $\hat{y} = \arg\max_y \mathrm{p}(\boldsymbol{x}_i, y)$
- Learning:

$$
\begin{aligned}
\boldsymbol{\theta} &= \arg\max_{\boldsymbol{\theta}} \mathrm{p}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) \\
\mathrm{p}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) &= \prod_i \mathrm{p}(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) = \prod_i \mathrm{p}(\boldsymbol{x}_i | y_i) \mathrm{p}(y_i) \\
\phi_{y,j} &= \frac{\sum_{i:Y_i=y} x_{ij}}{\sum_{i:Y_i=y} \sum_j x_{ij}} \\
\mu_y &= \frac{\text{count}(Y = y)}{N}
\end{aligned}
$$

This gives the maximum likelihood estimator (MLE; same as relative frequency estimator)

The Naivety of Naïve Bayes

$$\log \mathbf{p}(y_i, \boldsymbol{x}_i) = \log \mathbf{p}(\boldsymbol{x}_i \mid y_i) + \log \mathbf{p}(y_i)$$

$$= \sum_j \log \mathbf{p}(x_{i,j} \mid y_i) + \log \mathbf{p}(y_i)$$

Conditional independence assumption

Naïve Bayes: Example

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no originality

Smoothing

- Goal: assign some probability mass to events that were not seen during training
- One method: “add alpha” smoothing
  - Often, alpha = 1

$$\phi_{y,j} = \frac{\alpha + \sum_{i:Y_i=y} x_{i,j}}{\sum_{j'=1}^{V}\left(\alpha + \sum_{i:Y_i=y} x_{i,j'}\right)} = \frac{\alpha + \mathbf{count}(y,j)}{V\alpha + \sum_{j'=1}^{V} \mathbf{count}(y,j')}$$

Multinomial Naïve Bayes: Learning in Practice

- From training corpus, extract *Vocabulary*
- Calculate $P(y_j)$ terms
  - For each y_j in Y do

    $docs_j \leftarrow$ all docs with class $= y_j$

    $$P(y_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid y_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

    $$P(w_k \mid y_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Bias Variance trade-off

- **Variance** of a classifier
  - How much its decisions are affected by small changes in training sets
  - Lower variance = smaller changes
- **Bias** of a classifier
  - How accurate it is at modeling different training sets
  - Lower bias = more accurate

- High variance classifiers tend to **overfit**
- High bias classifiers tend to **underfit**

Bias Variance trade-off

- Impact of smoothing
  - Lowers variance
  - Increases bias (toward uniform probabilities)

Naïve Bayes

- A linear classifier whose weights can be interpreted as parameters of a probabilistic model
- Pros
  - parameters are easy to estimate from data:
    "count and normalize" (and smooth)
- Cons
  - requires making a conditional independence assumption
  - which does not hold in practice

Today

- 3 linear classifiers
  - Naïve Bayes
  - Perceptron
  - Logistic Regression

- Bag of words vs. rich feature sets
- Generative vs. discriminative models
- Bias-variance tradeoff
  - Smoothing, regularization

Beyond Bag of Words for classification tasks

Given an introductory sentence in Wikipedia

predict whether the article is about a person

Given		Predict
Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods.	→	Yes!
Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.	→	No!

Designing features

Contains "priest" → probably person!

Contains "(<#>-<#>)" → probably person!

Gonso was a Sanron sect priest (754 – 827) in the late Nara and early Heian periods .

Contains "site" → probably not person!

Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura , Maizuru City , Kyoto Prefecture .

Contains "Kyoto Prefecture" → probably not person!

Predicting requires combining information

- Given features and weights

$$w_{\text{contains “priest”}} = 2 \qquad w_{\text{contains “(<\#>-<\#>)”}} = 1$$

$$w_{\text{contains “site”}} = -3 \qquad w_{\text{contains “Kyoto Prefecture”}} = -1$$

- Predicting for a **new example**:
  - If (sum of weights > 0), “yes”; otherwise “no”

Kuya (903-972) was a priest born in Kyoto Prefecture. → $2 + -1 + 1 = 2$

Formalizing binary classification with linear models

$$
\begin{aligned}
y &= \text{sign}\left(\mathbf{w}\cdot\boldsymbol{\varphi}(x)\right) \\
&= \text{sign}\left(\sum_{i=1}^{I} w_i \cdot \varphi_i(x)\right)
\end{aligned}
$$

- **x**: the input
- **φ(x)**: vector of feature functions $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}$
- **w**: the weight vector $\{w_1, w_2, \ldots, w_I\}$
- **y**: the prediction, +1 if "yes", -1 if "no"
  - (sign(v) is +1 if v >= 0, -1 otherwise)

Example feature functions: Unigram features

- Number of times a particular word appears
  - i.e. bag of words

$$x = \text{A site , located in Maizuru , Kyoto}$$

$$\varphi_{\text{unigram "A"}}(x) = 1 \quad \varphi_{\text{unigram "site"}}(x) = 1 \quad \varphi_{\text{unigram ","}}(x) = 2$$

$$\varphi_{\text{unigram "located"}}(x) = 1 \quad \varphi_{\text{unigram "in"}}(x) = 1$$

$$\varphi_{\text{unigram "Maizuru"}}(x) = 1 \quad \varphi_{\text{unigram "Kyoto"}}(x) = 1$$

$$\varphi_{\text{unigram "the"}}(x) = 0 \quad \varphi_{\text{unigram "temple"}}(x) = 0$$

…

The rest are all 0

An **online** learning algorithm

```
create map w
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```

Perceptron weight update

$$\mathbf{w} \leftarrow \mathbf{w} + y\,\boldsymbol{\varphi}(x)$$

- If y = 1, increase the weights for features in $\boldsymbol{\varphi}(x)$
- If y = -1, decrease the weights for features in $\boldsymbol{\varphi}(x)$

Example: initial update

- Initialize **w**=**0**

x = A site , located in Maizuru , Kyoto   y = -1

$$\boldsymbol{w} \cdot \boldsymbol{\varphi}(x) = 0 \qquad y' = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{\varphi}(x)) = 1$$

↓

$$y' \neq y$$

↓

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{\varphi}(x)$$

↓

$w_{\text{unigram "Maizuru"}} = -1$

$w_{\text{unigram ","}} = -2$

$w_{\text{unigram "in"}} = -1$

$w_{\text{unigram "Kyoto"}} = -1$

$w_{\text{unigram "A"}} = -1$

$w_{\text{unigram "site"}} = -1$

$w_{\text{unigram "located"}} = -1$

Example: second update

$\mathbf{x}$ = Shoken , monk born in Kyoto $\quad y = 1$

-2 -1 -1

$$\boldsymbol{w} \cdot \varphi(x) = -4 \qquad y' = \text{sign}(\boldsymbol{w} \cdot \varphi(x)) = -1$$

$$y' \neq y$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\varphi(x)$$

$w_{\text{unigram “Maizuru”}} = -1$
$w_{\text{unigram “,”}} = -1$
$w_{\text{unigram “in”}} = 0$
$w_{\text{unigram “Kyoto”}} = 0$

$w_{\text{unigram “A”}} = -1$
$w_{\text{unigram “site”}} = -1$
$w_{\text{unigram “located”}} = -1$

$w_{\text{unigram “Shoken”}} = 1$
$w_{\text{unigram “monk”}} = 1$
$w_{\text{unigram “born”}} = 1$

Perceptron

- A linear model for classification
- An algorithm to learn feature weights given labeled data
  - online algorithm
  - error-driven
  - Does it converge?
    - See “A Course In Machine Learning” Ch.3

Multiclass perceptron

$$\hat{y} = \arg\max_{y} \boldsymbol{\theta}^{\top} \mathbf{f}(\mathbf{x}, y)$$

Algorithm 1 Perceptron learning algorithm

procedure PERCEPTRON($\boldsymbol{x}_{1:N}, y_{1:N}$)
    repeat
        Select an instance i
        $\hat{y} \leftarrow \arg\max_y \boldsymbol{\theta}_t^{\top} \boldsymbol{f}(\boldsymbol{x}_i, y)$
        if $\hat{y} \neq y_i$ **then**
            $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, \hat{y})$
        else
            do nothing
    until tired

Bias Variance trade off

- How do we decide when to stop?
  - Accuracy on held out data
  - Early stopping

- Averaged perceptron
  - Improves generalization

Averaged perceptron

Algorithm 2 Averaged perceptron learning algorithm

procedure AVG-PERCEPTRON($\boldsymbol{x}_{1:N}, y_{1:N}$)
  repeat
    Select an instance i
    $\hat{y} \leftarrow \arg\max_y \boldsymbol{\theta}_t^\top \boldsymbol{f}(\boldsymbol{x}_i, y)$
    if $\hat{y} \neq y_i$ **then**
      $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, \hat{y})$
      $\boldsymbol{m} \leftarrow \boldsymbol{m} + \boldsymbol{\theta}_{t+1}$
    else
      do nothing
  until tired
  $\overline{\boldsymbol{\theta}} \leftarrow \frac{1}{t}\boldsymbol{m}$

Learning as optimization: Loss functions

- Naïve Bayes chooses weights to maximize the joint likelihood of the training data (or log likelihood)

$$\log \mathrm{p}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \mathrm{p}(\boldsymbol{x}_i, y_i; \boldsymbol{\theta})$$

$$\ell_{\mathrm{NB}}(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i) = -\log \mathrm{p}(\boldsymbol{x}_i, y_i; \boldsymbol{\theta})$$

$$\hat{\boldsymbol{\theta}} = \operatorname*{arg\,min}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell_{\mathrm{NB}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$$

Perceptron Loss function

$$\ell_{\text{perceptron}}(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i) = \begin{cases} 0, & y_i = \mathbf{arg}\max_y \boldsymbol{\theta}^\top \boldsymbol{f}(x_i, y) \\ 1, & \text{otherwise} \end{cases}$$

- “0-1” loss

- Treats all errors equally
- Does not care about confidence of classification decision

Today

- 3 linear classifiers
  - Naïve Bayes
  - Perceptron
  - (Logistic Regression)

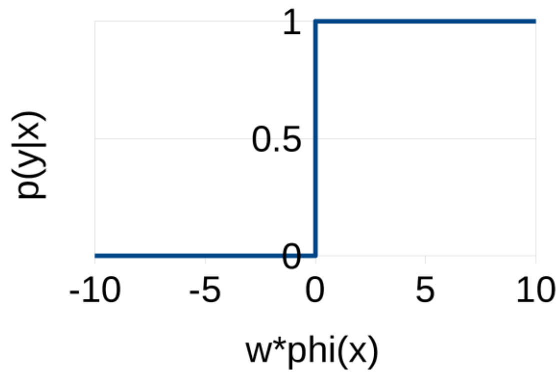
- Bag of words vs. rich feature sets
- Generative vs. discriminative models
- Bias-variance tradeoff

Perceptron & Probabilities

- What if we want a probability p(y|x)?
- The perceptron gives us a prediction y

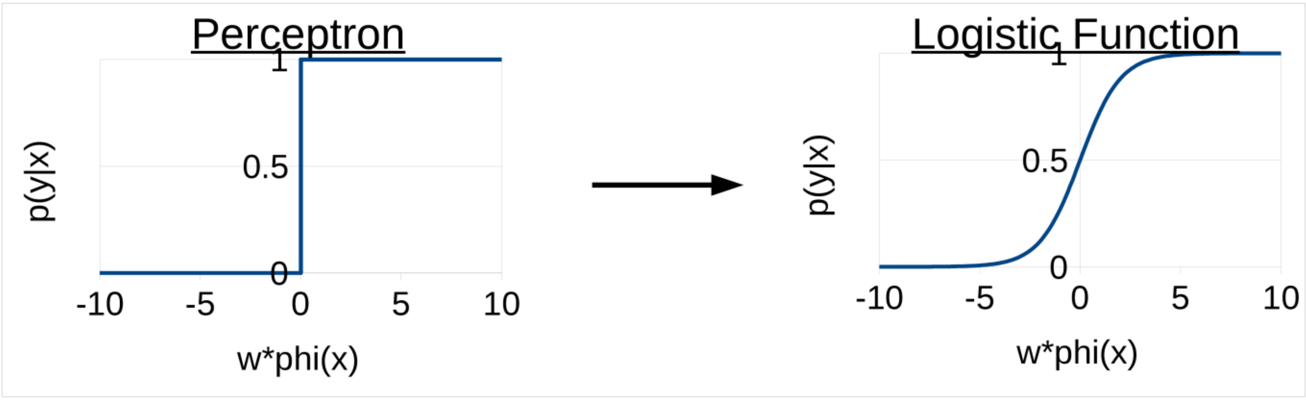
In other words:

$$P(y=1|x)=1 \text{ if } \mathbf{w}\cdot\boldsymbol{\varphi}(x)\geq 0$$
$$P(y=1|x)=0 \text{ if } \mathbf{w}\cdot\boldsymbol{\varphi}(x)<0$$



The logistic function

$$P(y=1|x)=\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}$$



- “Softer” function than in perceptron
- Can account for uncertainty
- Differentiable

Logistic regression: how to train?

- Train based on **conditional likelihood**
- Find parameters w that maximize conditional likelihood of all answers y_i given examples x_i

$$\hat{w} = \underset{w}{\operatorname{argmax}} \prod_i P(y_i | x_i ; w)$$

Stochastic gradient ascent (or descent)

- Online training algorithm for logistic regression
  - and other probabilistic models

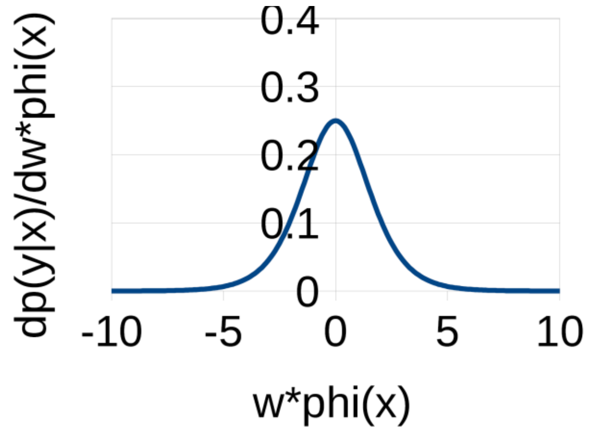
```
create map w
for I iterations
    for each labeled pair x, y in the data
        w += α * dP(y|x)/dw
```

- Update weights for every training example
- Move in direction given by gradient
- Size of update step scaled by learning rate

Gradient of the logistic function

$$\frac{d}{dw}P(y=1|x) = \frac{d}{dw}\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}$$

$$= \varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$



$$\frac{d}{dw}P(y=-1|x) = \frac{d}{dw}(1-\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}})$$

$$= -\varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$

Example: initial update

- Set α=1, initialize **w=0**

x = A site , located in Maizuru , Kyoto   y = -1

$$\mathbf{w}\cdot\varphi(x)=0 \qquad \frac{d}{d\mathbf{w}}P(y=-1|x) = -\frac{e^0}{(1+e^0)^2}\varphi(x)$$

$$= -0.25\,\varphi(x)$$

$$\mathbf{w} \leftarrow \mathbf{w} + -0.25\,\varphi(x)$$

$w_{\text{unigram "Maizuru"}}$ = -0.25
$w_{\text{unigram ","}}$ = -0.5
$w_{\text{unigram "in"}}$ = -0.25
$w_{\text{unigram "Kyoto"}}$ = -0.25
$w_{\text{unigram "A"}}$ = -0.25
$w_{\text{unigram "site"}}$ = -0.25
$w_{\text{unigram "located"}}$ = -0.25

Example: second update

x = Shoken , monk born in Kyoto   y = 1

-0.5   -0.25   -0.25

$$\boldsymbol{w} \cdot \boldsymbol{\varphi}(x) = -1 \qquad \frac{d}{d\,\boldsymbol{w}} P(y=1|x) = \frac{e^{1}}{(1+e^{1})^{2}} \boldsymbol{\varphi}(x)$$

$$= 0.196\,\boldsymbol{\varphi}(x)$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + 0.196\,\boldsymbol{\varphi}(x)$$

$w_{\text{unigram “Maizuru”}}$ = -0.25
$w_{\text{unigram “,”}}$ = -0.304
$w_{\text{unigram “in”}}$ = -0.054
$w_{\text{unigram “Kyoto”}}$ = -0.054

$w_{\text{unigram “A”}}$ = -0.25
$w_{\text{unigram “site”}}$ = -0.25
$w_{\text{unigram “located”}}$ = -0.25

$w_{\text{unigram “Shoken”}}$ = 0.196
$w_{\text{unigram “monk”}}$ = 0.196
$w_{\text{unigram “born”}}$ = 0.196

How to set the learning rate?

- Various strategies
  - decay over time

$$\alpha = \frac{1}{C + t}$$

Parameter

Number of samples

  - Use held-out test set, increase learning rate when likelihood increases

Some models are better then others…

- Consider these 2 examples

-1	he saw a bird in the park
+1	he saw a robbery in the park

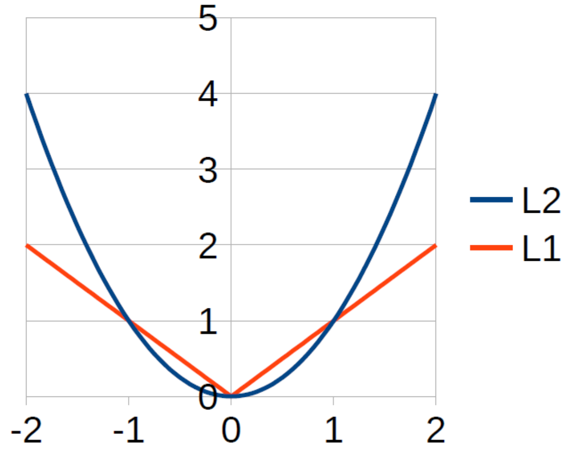
- Which of the 2 models below is better?

Classifier 1	Classifier 2
he +3	bird -1
saw -5	robbery +1
a +0.5	
bird -1	
robbery +1	
in +5	
the -3	
park -2	

Classifier 2 will probably generalize better!
It does not include irrelevant information
=> Smaller model is better

Regularization

- A penalty on adding extra weights
- L2 regularization: $\|w\|_2$
  - big penalty on large weights
  - small penalty on small weights
- L1 regularization: $\|w\|_1$
  - Uniform increase when large or small
  - Will cause many weights to become zero



L1 regularization in online learning

```
update_weights(w, phi, y, c)
★   for name, value in w:
★       if abs(value) < c:
★           w[name] = 0
★       else:
★           w[name] -= sign(value) * c
    for name, value in phi:
        w[name] += value * y
```

If abs. value < c, set weight to zero

If value > 0, decrease by c
If value < 0, increase by c

Today

- 3 linear classifiers
  - Naïve Bayes
  - Perceptron
  - (Logistic Regression)

- Bag of words vs. rich feature sets
- Generative vs. discriminative models
- Bias-variance tradeoff