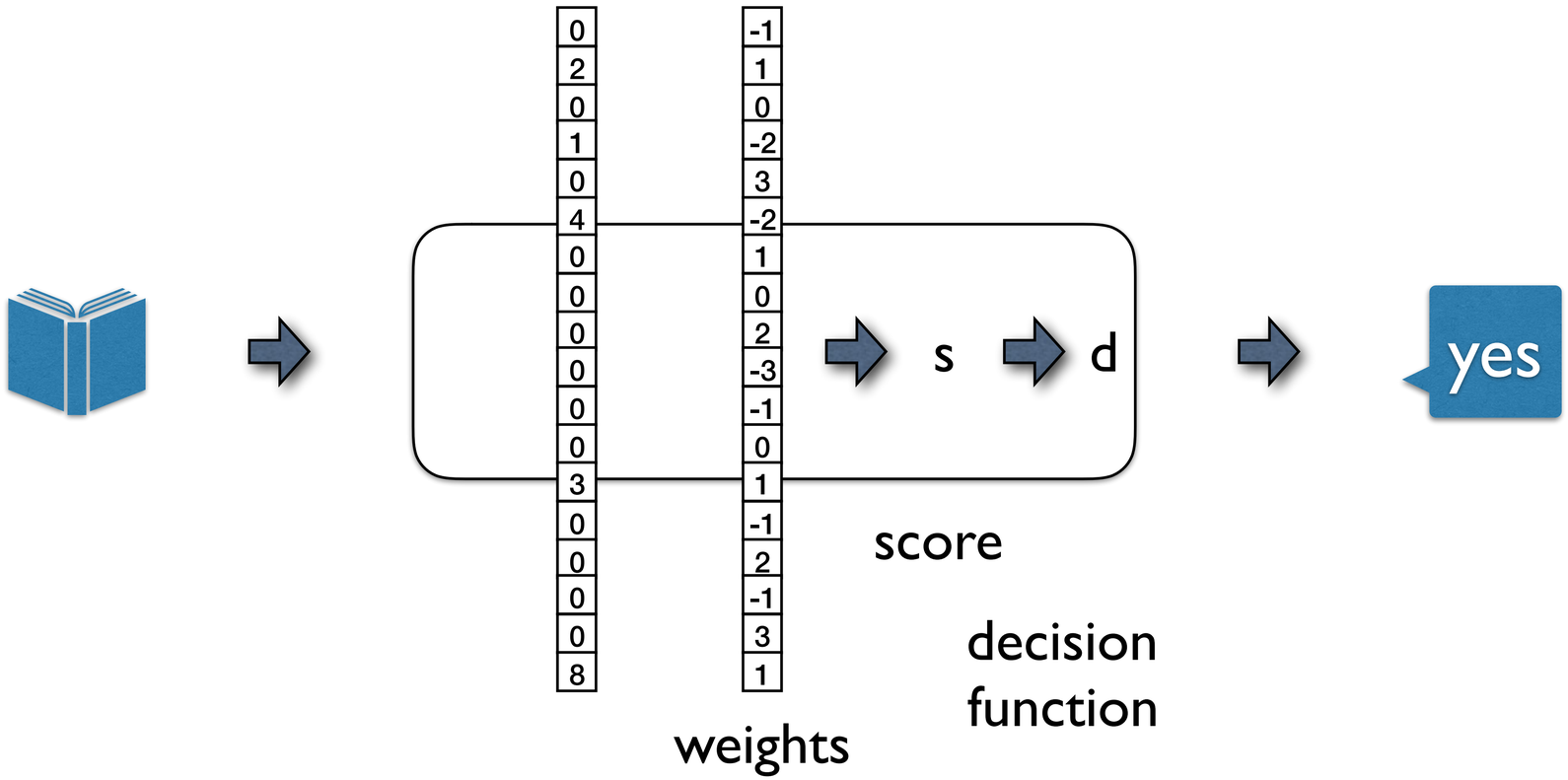
ANLP

07 - multicls. classification (classf., part II)

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

classification



Generative vs. Discriminative Models

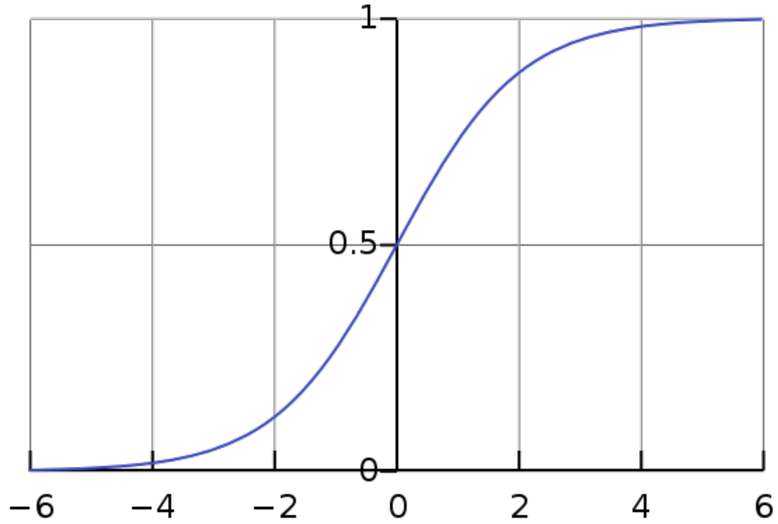
- Generative models: $P(x, y)$
  - Bayes nets / graphical models / Naive Bayes
  - Some of the model capacity goes to explaining the distribution of *x*; prediction uses Bayes rule post-hoc
- Discriminative models: $P(y|x)$
  - SVMs, logistic regression, CRFs, most neural networks
  - Model is trained to be good at prediction, but doesn't model *x*
- Up next: discriminative classifiers
- We'll come back to this distinction throughout this class

Logistic Regression

Logistic Regression

$$P(y = +|x) = \text{logistic}(w^\top x)$$

$$P(y = +|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)}$$



- To learn weights: maximize discriminative log likelihood of data (log P(y|x))

$$\mathcal{L}(\{x_j, y_j\}_{j=1,\ldots,n}) = \sum_j \log P(y_j|x_j)$$ corpus-level LL

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j)$$ one (positive) example LL

$$= \sum_{i=1}^{n} w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

sum over features

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \sum_{i=1}^{n} w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = x_{ji} - \frac{\partial}{\partial w_i} \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$= x_{ji} - \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)} \frac{\partial}{\partial w_i}\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

deriv of log

$$= x_{ji} - \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)} x_{ji} \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)$$

deriv of exp

$$= x_{ji} - x_{ji} \frac{\exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)} = x_{ji}(1 - P(y_j = +|x_j))$$

Logistic Regression

- Gradient of w_i on positive example $= x_{ji}(1 - P(y_j = +|x_j))$

  If P(+) is close to 1, make very little update

  Otherwise make w_i look more like x_{ji}, which will increase P(+)

- Gradient of w_i on negative example $= x_{ji}(-P(y_j = +|x_j))$

  If P(+) is close to 0, make very little update

  Otherwise make w_i look less like x_{ji}, which will decrease P(+)

- Let $y_j = 1$ for positive instances, $y_j = 0$ for negative instances.
- Can combine these gradients as $x_j(y_j - P(y_j = 1|x_j))$

Example

(1) *this movie was great! would watch again* +

(2) *I expected a great movie and left happy* +

(3) *great potential but ended up being a flop* —

$f(x_1)$ = [1 1]

$f(x_2)$ = [1 1]

$f(x_3)$ = [1 0]

[contains *great*] [contains *movie*]

position 0 position 1

w = [0, 0] ⟶ $P(y = 1 \mid x_1)$ = exp(0)/(1 + exp(0)) = 0.5 ⟶ g = [0.5, 0.5]

w = [0.5, 0.5] ⟶ $P(y = 1 \mid x_2)$ = logistic(1) ≈ 0.75 ⟶ g = [0.25, 0.25]

w = [0.75, 0.75] ⟶ $P(y = 1 \mid x_3)$ = logistic(0.75) ≈ 0.67 ⟶ g = [-0.67, 0]

w = [0.08, 0.75] …

$$P(y = +|x) = \text{logistic}(w^\top x)$$

$$x_j(y_j - P(y_j = 1|x_j))$$

Regularization

- Can end up making extreme updates to fit the training data

$w_{funds} = +1000$

$w_{transfer} = -900$

$w_{send} = +742$

$w_{the} = +203$

- All examples have P(correct) > 0.999, but classifier does crazy things on new examples

Regularization

- Regularizing an objective can mean many things, including an L2-norm penalty to the weights:

$$\sum_{j=1}^{m} \mathcal{L}(x_j, y_j) - \lambda \|w\|_2^2$$

- Keeping weights small can prevent overfitting
- For most of the NLP models we build, explicit regularization isn't necessary
  - Early stopping
  - Large numbers of sparse features are hard to overfit in a really bad way
  - For neural networks: dropout and gradient clipping

Logistic Regression: Summary

- Model

$$P(y = +|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)}$$

- Inference

$$\text{argmax}_y P(y|x)$$

$$P(y = 1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0$$

- Learning: gradient ascent on the (regularized) discriminative log-likelihood

a.k.a. Maximum Entropy

- in NLP, logistic regression used to be known as Maximum Entropy Classifier

- (from all the models that fit the data, choose the one that has the maximum entropy = makes the fewest assumptions not validated by data)

Perceptron/SVM

invented in the 1950s

- Frank Rosenblatt, 1958
- explicitly inspired by what was known then about neurons
- … based on Rosenblatt's statements, The New York Times [in 1958] reported the perceptron to be "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.“ [https://en.wikipedia.org/wiki/Perceptron]

Perceptron

- Simple error-driven learning approach similar to logistic regression

- Decision rule: $w^\top x > 0$
  - If incorrect: if positive, $w \leftarrow w + x$

    if negative, $w \leftarrow w - x$

Logistic Regression

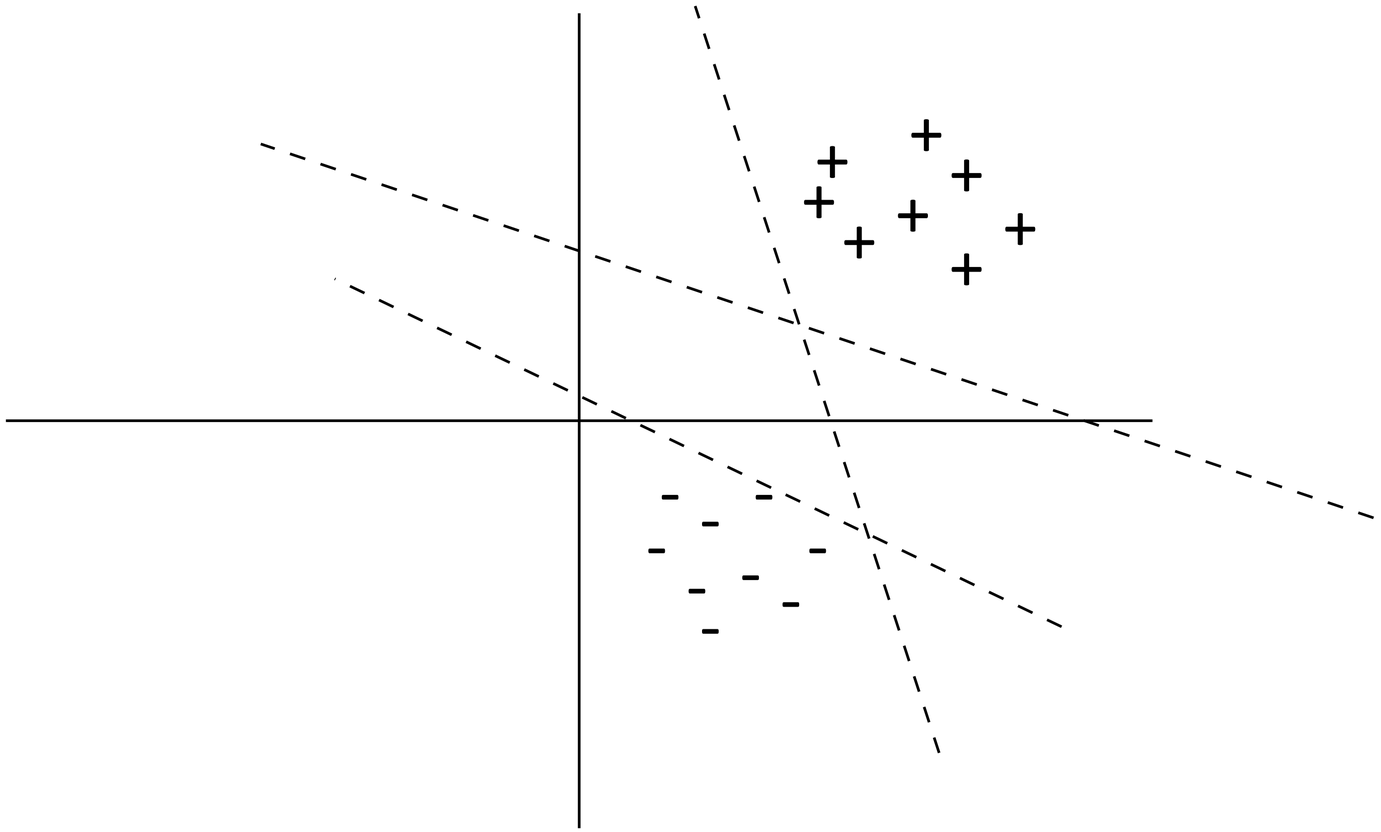
$$w \leftarrow w + x(1 - P(y = 1|x))$$

$$w \leftarrow w - xP(y = 1|x)$$

- Guaranteed to eventually separate the data if the data are separable

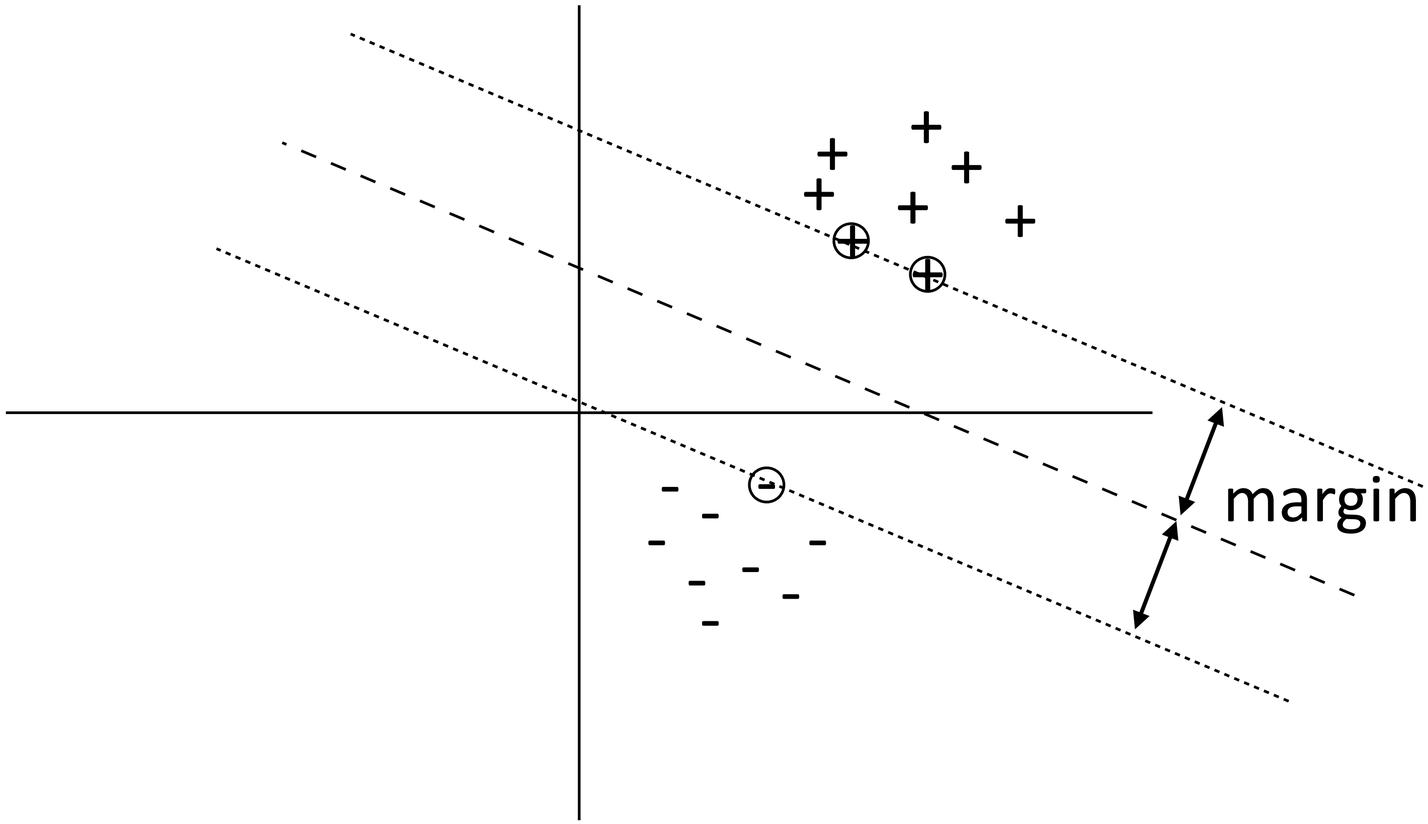
Support Vector Machines

- Many separating hyperplanes — is there a best one?



Support Vector Machines

- Many separating hyperplanes — is there a best one?



Support Vector Machines

- Constraint formulation: find *w* via following quadratic program:

Minimize $\|w\|_2^2$

s.t. $\forall j \quad w^\top x_j \geq 1 \text{ if } y_j = 1$

$w^\top x_j \leq -1 \text{ if } y_j = 0$

minimizing norm with fixed margin <=> maximizing margin

As a single constraint:

$$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1$$

- Generally no solution (data is generally non-separable) — need slack!

N-Slack SVMs

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j \qquad \forall j \quad \xi_j \geq 0$

- The ξ_j are a "fudge factor" to make all constraints satisfied
- Take the gradient of the objective:

$$\frac{\partial}{\partial w_i}\xi_j = 0 \text{ if } \xi_j = 0 \qquad \frac{\partial}{\partial w_i}\xi_j = (2y_j - 1)x_{ji} \text{ if } \xi_j > 0$$

$$= x_{ji} \text{ if } y_j = 1, \ -x_{ji} \text{ if } y_j = 0$$

- Looks like the perceptron! But updates more frequently

Gradients on Positive Examples

Logistic regression

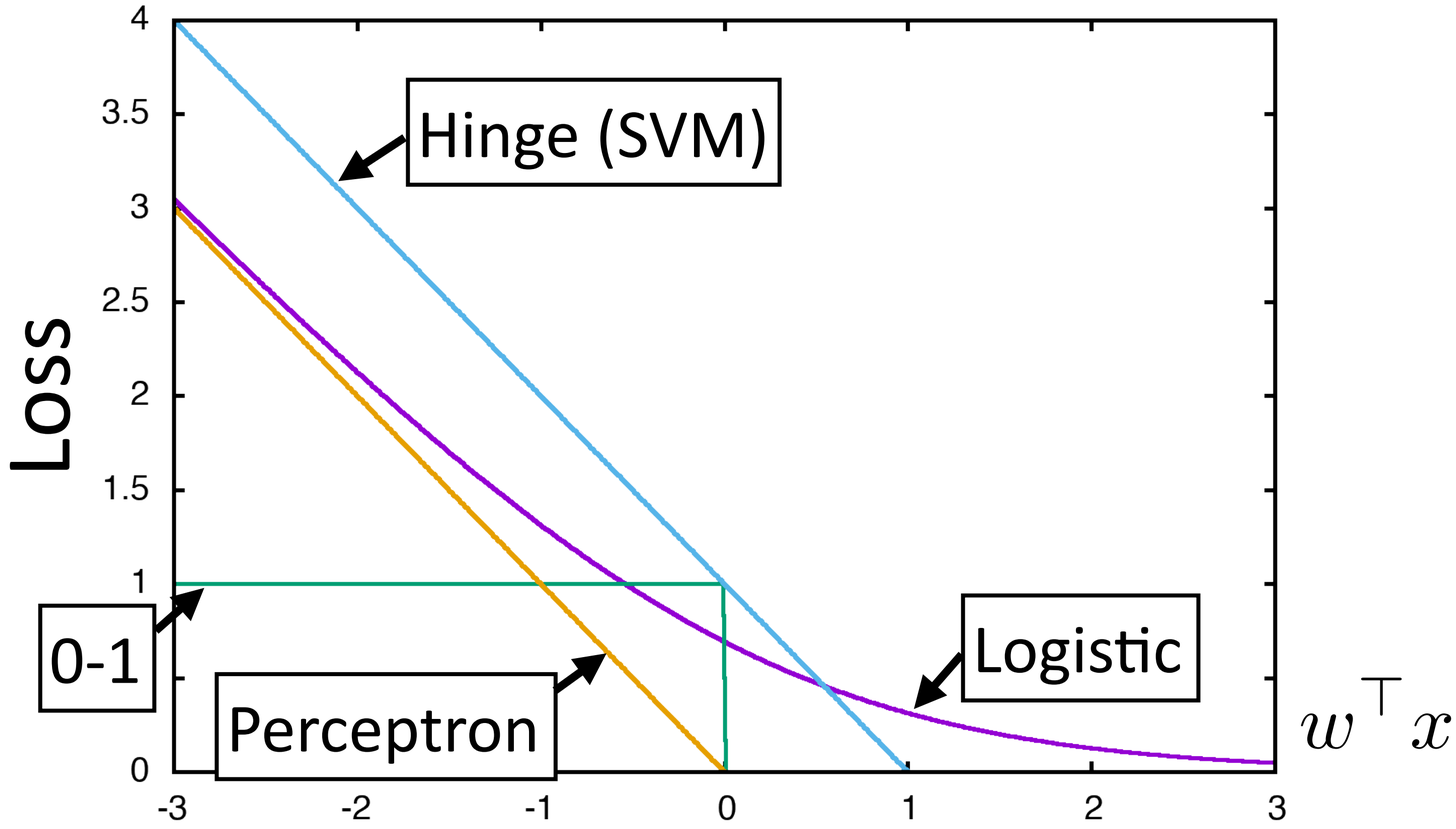
$$x(1 - \text{logistic}(w^\top x))$$

Perceptron

$$x \text{ if } w^\top x < 0, \text{ else } 0$$

SVM (ignoring regularizer)

$$x \text{ if } w^\top x < 1, \text{ else } 0$$



Comparing Gradient Updates (Reference)

Logistic regression (unregularized)

$x(y - P(y = 1|x))$ = $x(y - \text{logistic}(w^\top x))$

y = 1 for pos,
0 for neg

Perceptron

$(2y - 1)x$ if classified incorrectly

0 else

SVM

$(2y - 1)x$ if not classified correctly with margin of 1

0 else

Optimization

Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - “First-order” technique: only relies on having gradient
  - Can avg gradient over a few examples and apply update once (minibatch)
  - Setting step size is hard (decrease when held-out performance worsens?)

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

- Newton’s method
  - Second-order technique
  - Optimizes quadratic instantly

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

Inverse Hessian: *n* x *n* mat, expensive!

- Quasi-Newton methods: L-BFGS, etc. approximate inverse Hessian

Sentiment Analysis

Sentiment Analysis

this movie was great! would watch again +

the movie was gross and overwrought, but I liked it +

this movie was not really very enjoyable —

- Bag-of-words doesn't seem sufficient (discourse structure, negation)
- There are some ways around this: extract bigram feature for "*not* X" for all X following the *not*

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (2002)

Sentiment Analysis

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	"	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

- Simple feature sets can do pretty well!

Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (2002)

Sentiment Analysis

Method	RT-s	MPQA
MNB-uni	77.9	85.3
MNB-bi	**79.0**	**86.3**
SVM-uni	76.2	86.1
SVM-bi	77.7	**86.7**
NBSVM-uni	**78.1**	85.3
NBSVM-bi	**79.4**	**86.3**
RAE	76.8	85.7
RAE-pretrain	77.7	**86.4**
Voting-w/Rev.	63.1	81.7
Rule	62.9	81.8
BoF-noDic.	75.7	81.8
BoF-w/Rev.	76.4	84.1
Tree-CRF	77.3	86.1
BoWSVM	–	–
Kim (2014) CNNs	**81.5**	**89.5**

Naive Bayes is doing well!

Ng and Jordan (2002) — NB can be better for small data

Before neural nets had taken off — results weren't that great

Wang and Manning (2012)

Sentiment Analysis

- Stanford Sentiment Treebank (SST) binary classification
- Best systems now: large pretrained networks
- 90 -> 97 over the last 2 years

Model	Accuracy	Paper / Source	Code
XLNet-Large (ensemble) (Yang et al., 2019)	96.8	XLNet: Generalized Autoregressive Pretraining for Language Understanding	Official
MT-DNN-ensemble (Liu et al., 2019)	96.5	Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding	Official
Snorkel MeTaL(ensemble) (Ratner et al., 2018)	96.2	Training Complex Models with Multi-Task Weak Supervision	Official
MT-DNN (Liu et al., 2019)	95.6	Multi-Task Deep Neural Networks for Natural Language Understanding	Official
Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)	94.9	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	Official
...			
Neural Semantic Encoder (Munkhdalai and Yu, 2017)	89.7	Neural Semantic Encoders	
BLSTM-2DCNN (Zhou et al., 2017)	89.5	Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling	

https://github.com/sebastianruder/NLP-progress/blob/master/english/sentiment_analysis.md

Recap

- Logistic regression: $P(y=1|x) = \frac{\exp\left(\sum_{i=1}^{n} w_i x_i\right)}{\left(1+\exp\left(\sum_{i=1}^{n} w_i x_i\right)\right)}$

  Decision rule: $P(y=1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0$

  Gradient (unregularized): $x(y - P(y=1|x))$

- SVM:

  Decision rule: $w^\top x \geq 0$

  (Sub)gradient (unregularized): 0 if correct with margin of 1, else $x(2y-1)$

Recap

- Logistic regression, SVM, and perceptron are closely related
- SVM and perceptron inference require taking maxes, logistic regression has a similar update but is “softer” due to its probabilistic nature
- All gradient updates: “make it look more like the right thing and less like the wrong thing”
- Next step: multiclass classification

Recap: Binary Classification

- Logistic regression: $P(y=1|x) = \frac{\exp\left(\sum_{i=1}^{n} w_i x_i\right)}{\left(1+\exp\left(\sum_{i=1}^{n} w_i x_i\right)\right)}$

  Decision rule: $P(y=1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0$

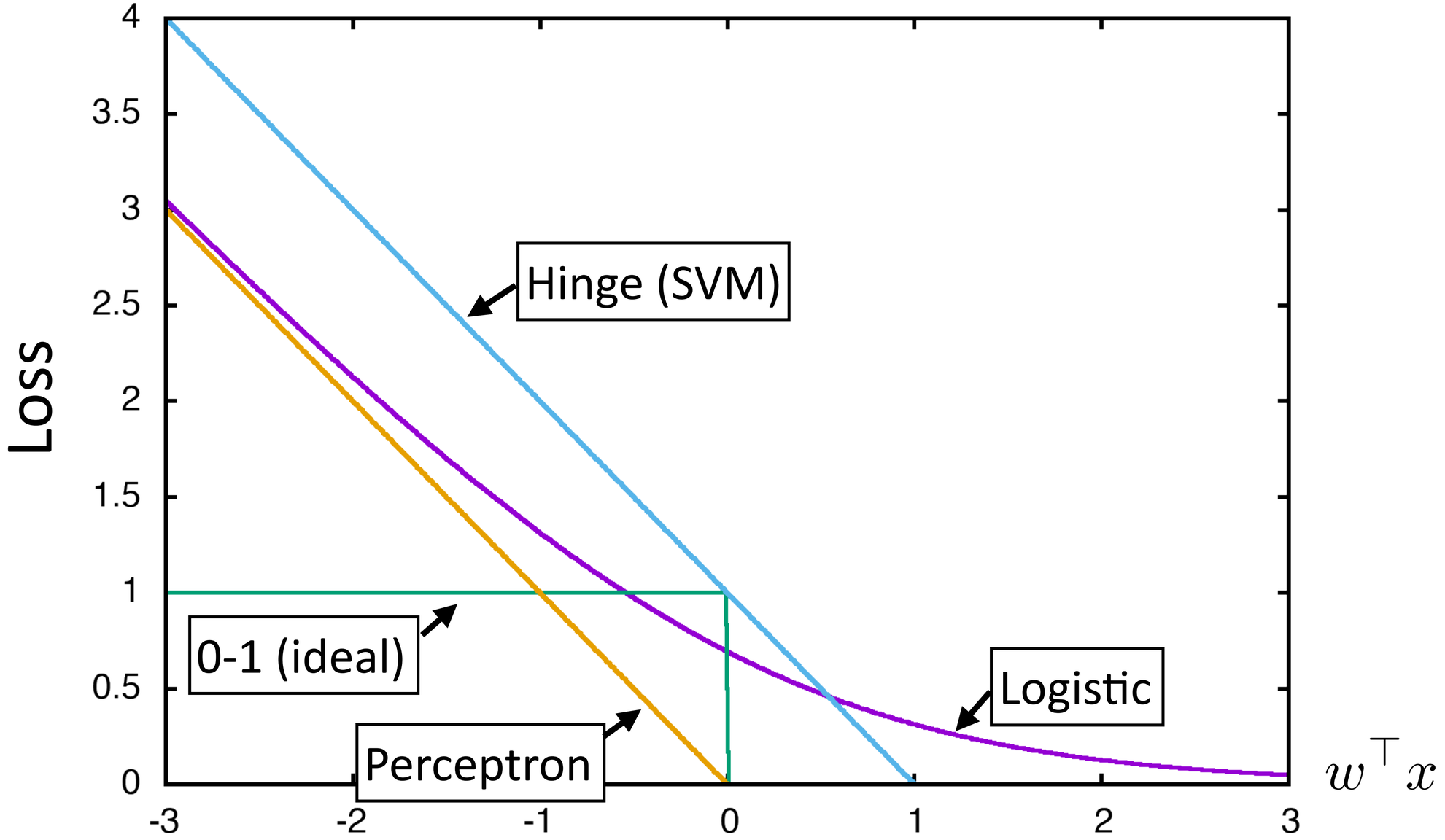
  Gradient (unregularized): $x(y - P(y=1|x))$

- SVM: quadratic program to minimize weight vector norm w/slack

  Decision rule: $w^\top x \geq 0$

  (Sub)gradient (unregularized): 0 if correct with margin of 1, else $x(2y-1)$

Loss Functions



This Lecture

- Multiclass fundamentals
- Feature extraction
- Multiclass logistic regression
- Multiclass SVM
- Generative models revisited

Multiclass Fundamentals

Text Classification

A Cancer Conundrum: Too Many Drug Trials, Too Few Patients

Breakthroughs in immunotherapy and a rush to develop profitable new treatments have brought a crush of clinical trials scrambling for patients.

By GINA KOLATA

→ Health

Yankees and Mets Are on Opposite Tracks This Subway Series

As they meet for a four-game series, the Yankees are playing for a postseason spot, and the most the Mets can hope for is to play spoiler.

By FILIP BONDY

→ Sports

~20 classes

Image Classification

Dog

Car

- Thousands of classes (ImageNet)

Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.

- 4,500,000 classes (all articles in Wikipedia)

Reading Comprehension

One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

3) Where did James go after he went to the grocery store?

A) his deck
B) his freezer
C) a fast food restaurant
D) his room

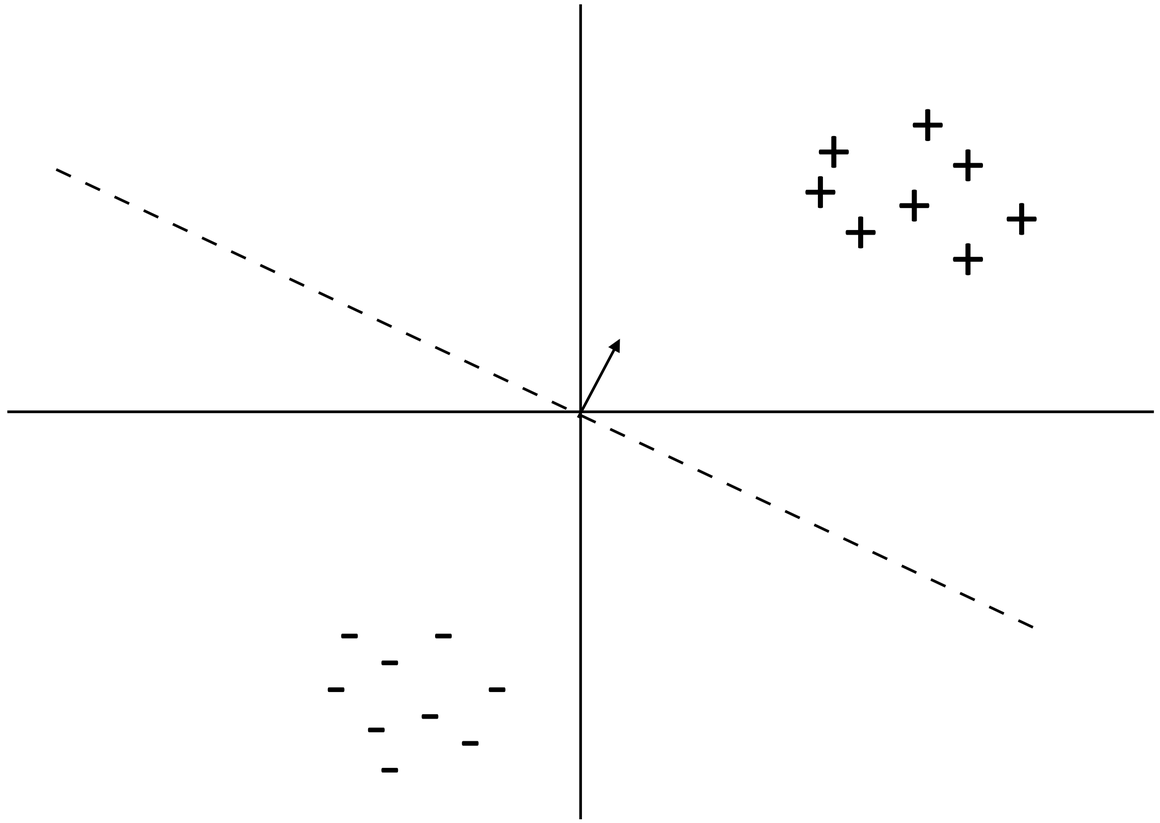
After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle. 

- Multiple choice questions, 4 classes (but classes change per example)

Richardson (2013)

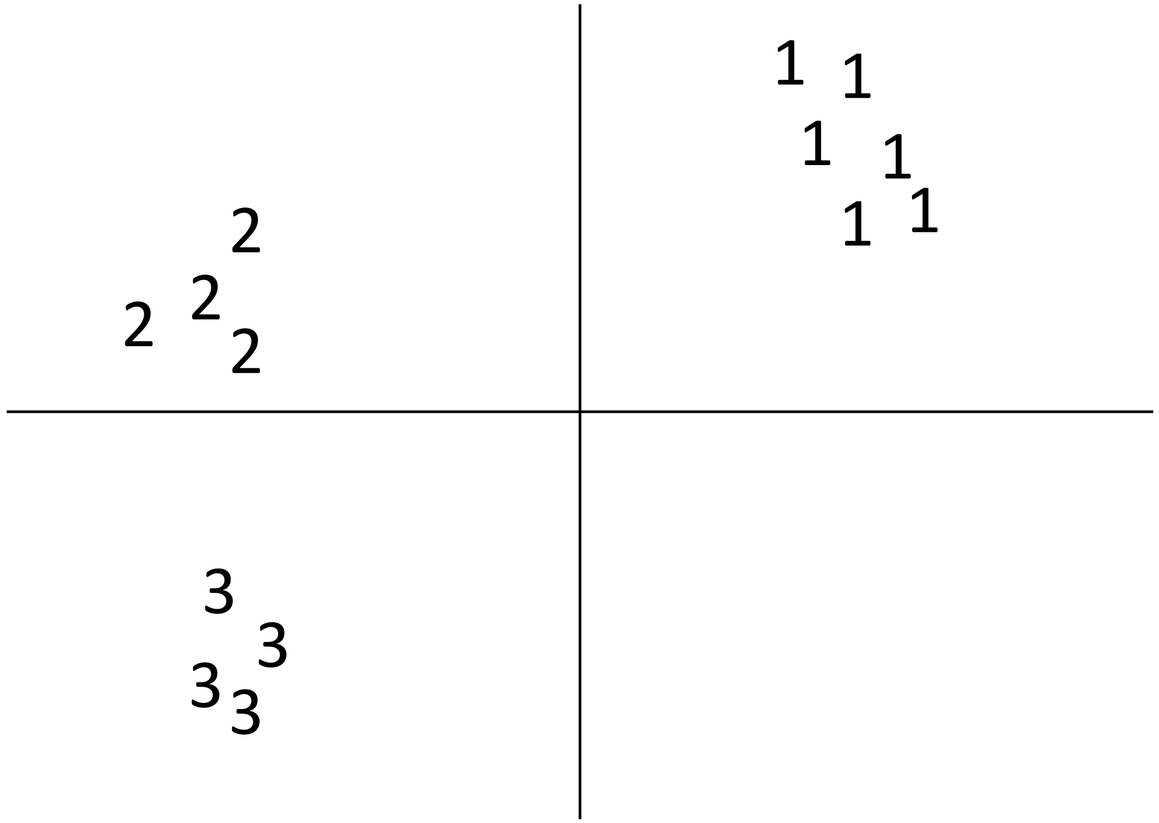
Binary Classification

- Binary classification: one weight vector defines positive and negative classes



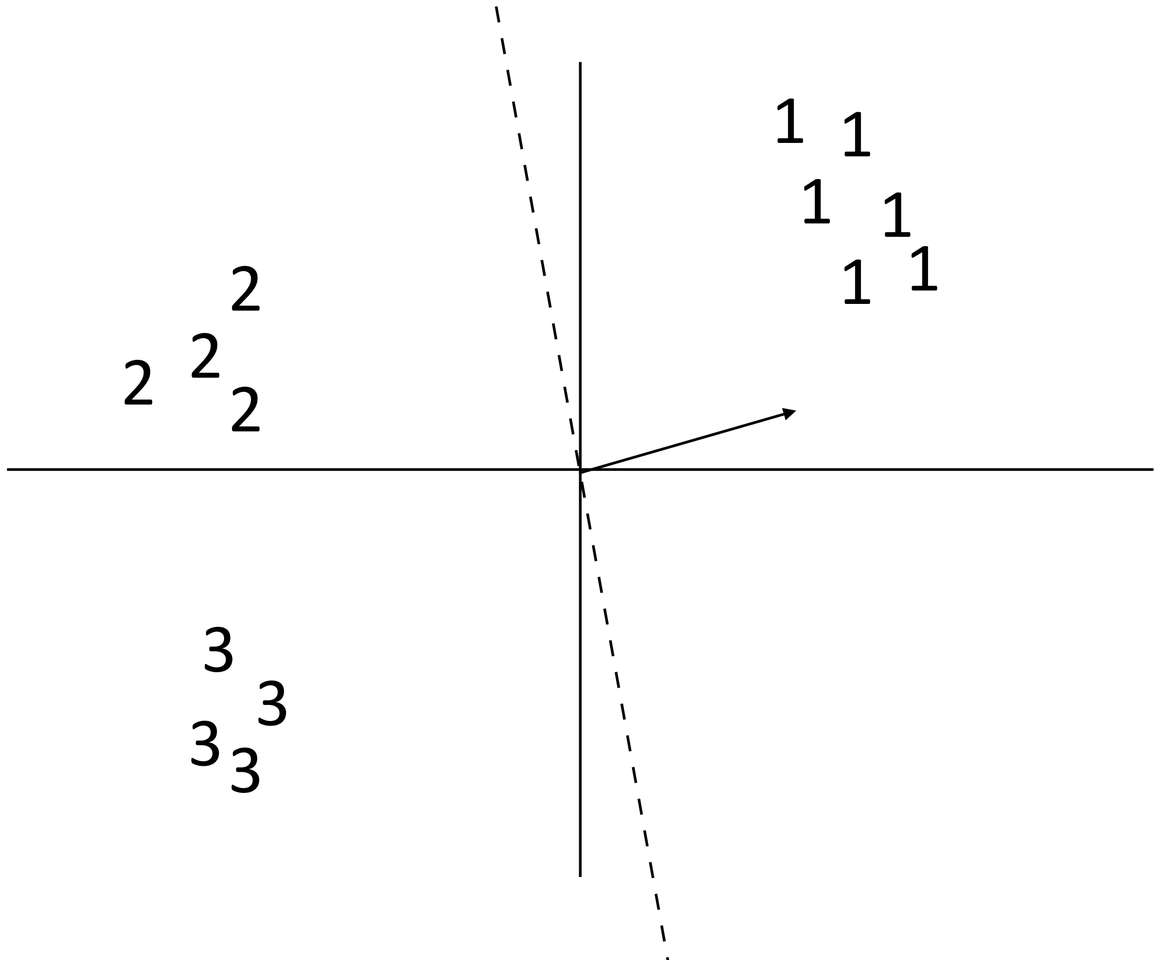
Multiclass Classification

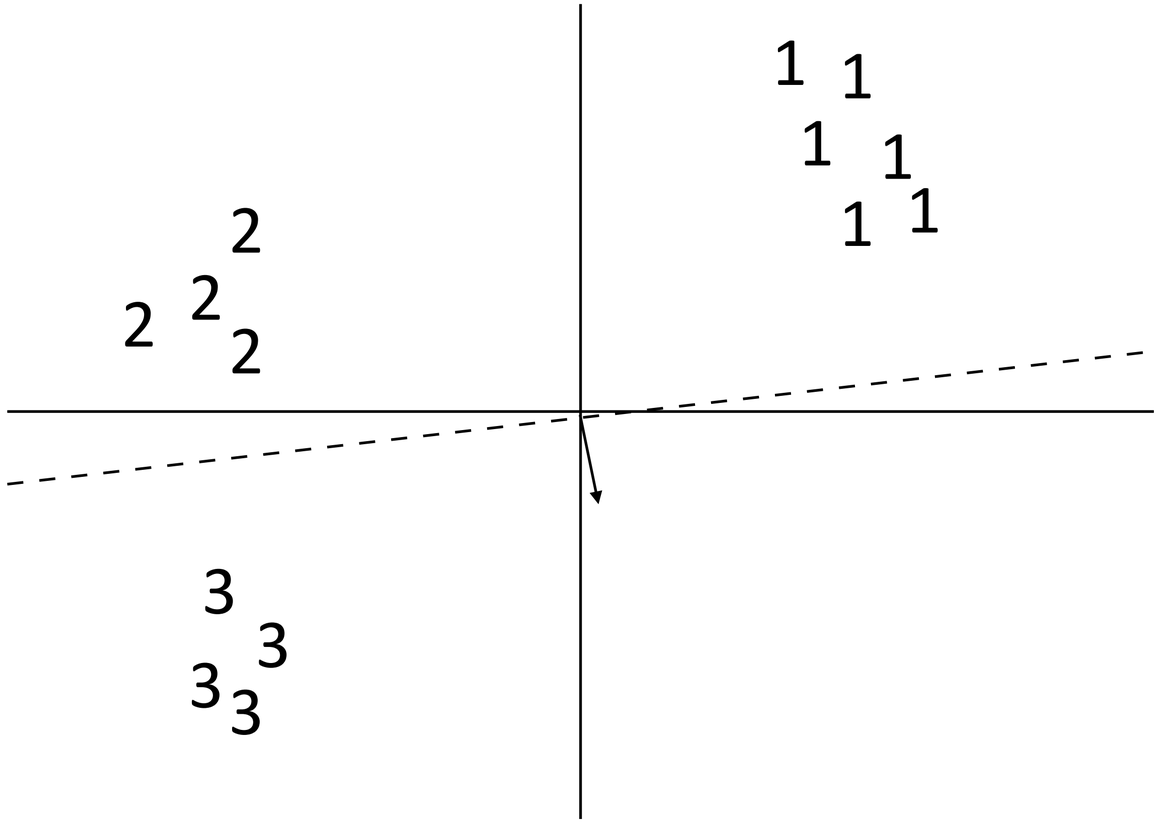
- Can we just use binary classifiers here?



Multiclass Classification

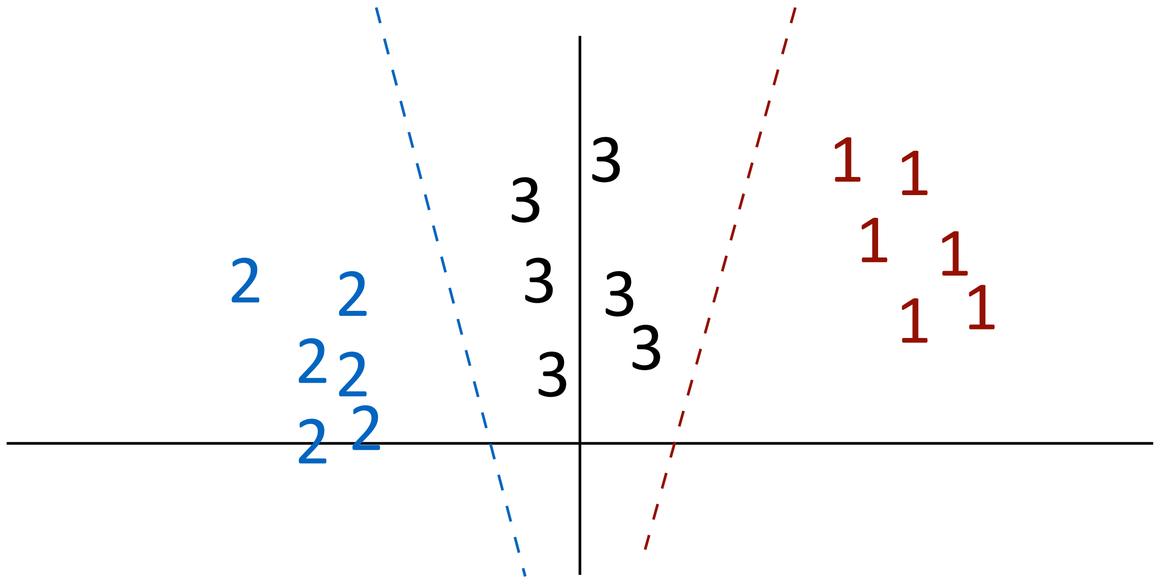
- One-vs-all: train *k* classifiers, one to distinguish each class from all the rest
- How do we reconcile multiple positive predictions? Highest score?





Multiclass Classification

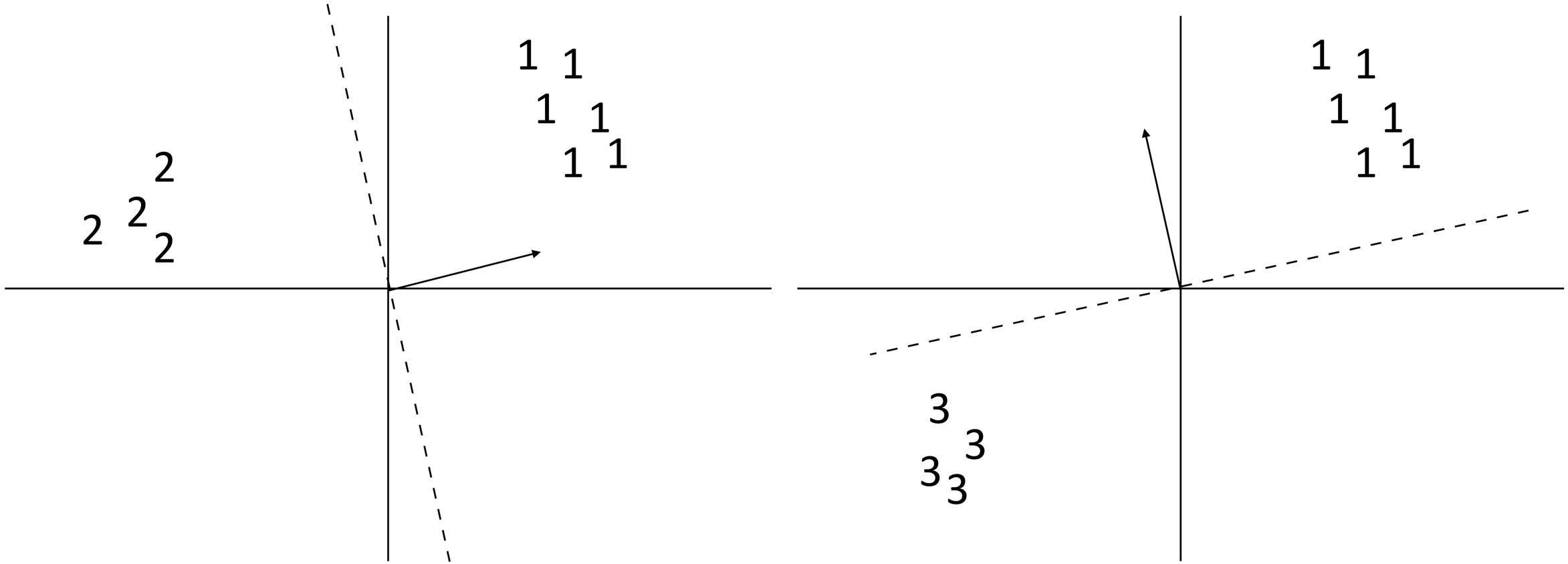
- Not all classes may even be separable using this approach



- Can separate 1 from 2+3 and 2 from 1+3 but not 3 from the others (with these features)

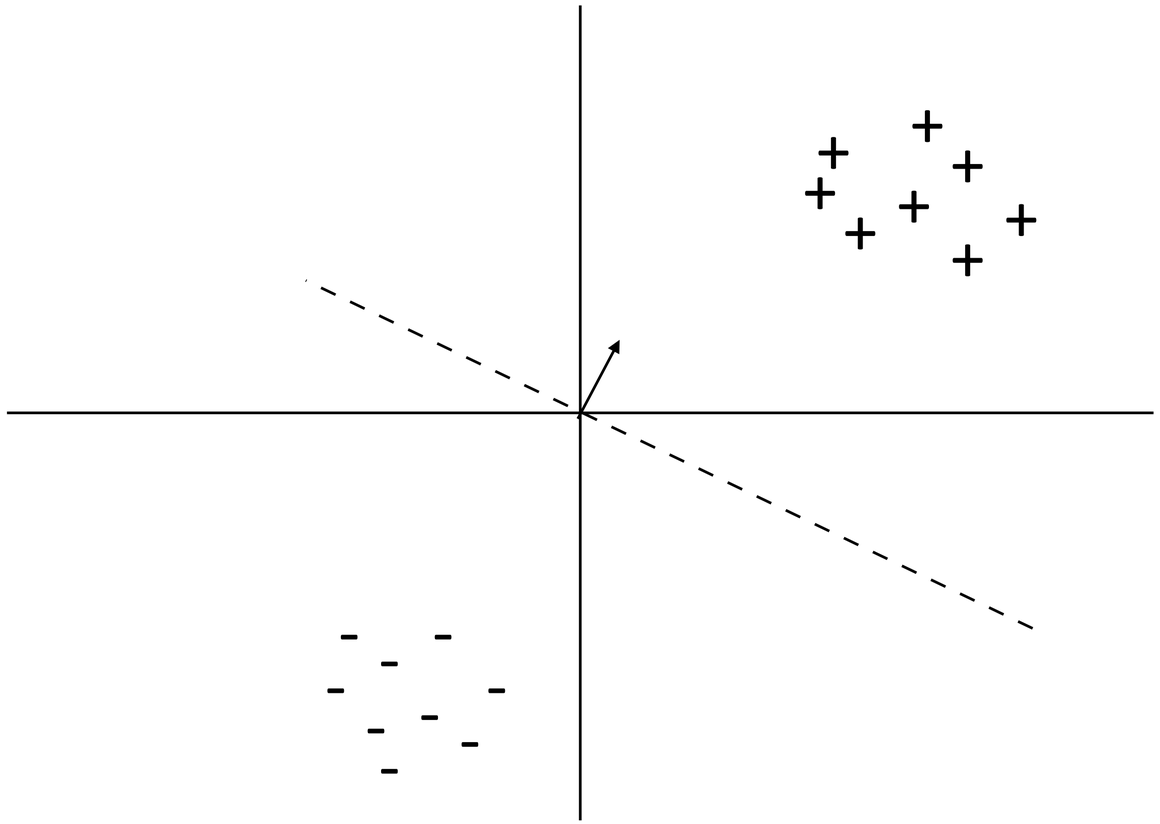
Multiclass Classification

- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes
- Again, how to reconcile?

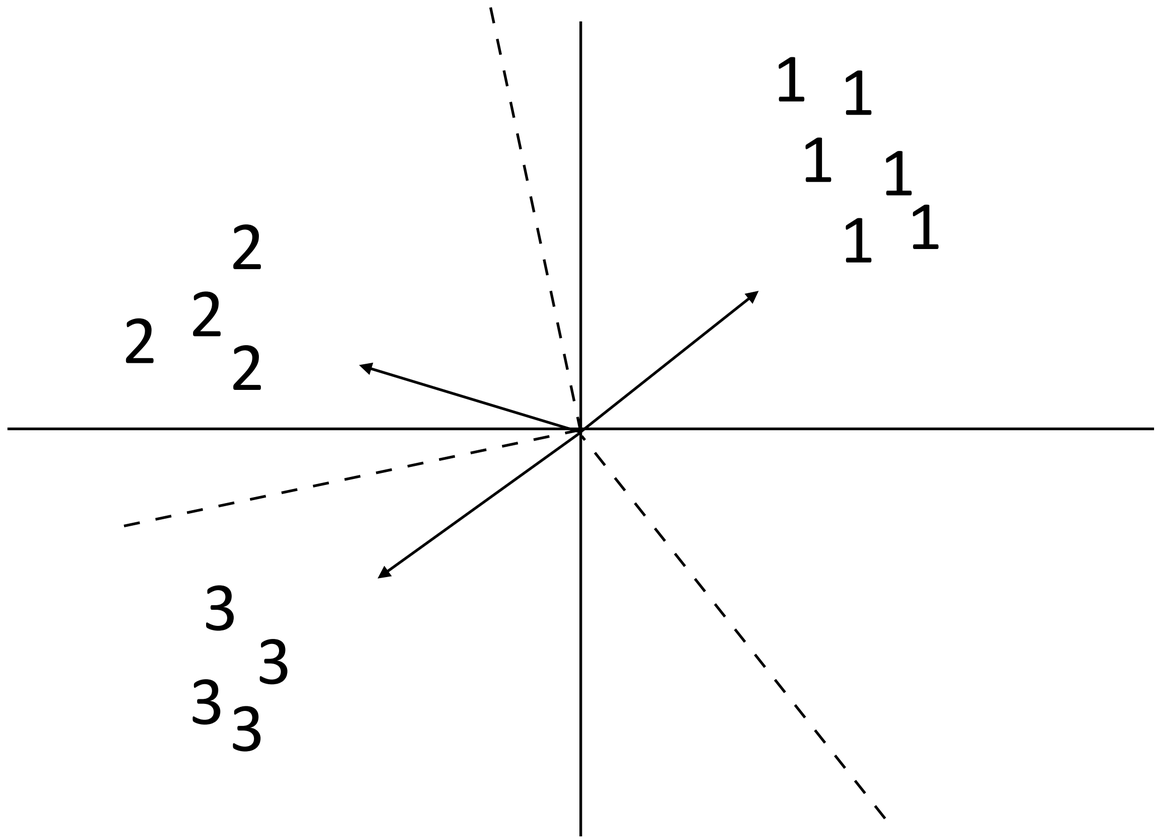


Multiclass Classification

- Binary classification: one weight vector defines both classes



- Multiclass classification: different weights and/or features per class



Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees
- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$ ← features depend on choice of label now! note: this isn't the gold label
  - Multiple feature vectors, one weight vector
- Can also have one weight vector per class: $\text{argmax}_{y \in \mathcal{Y}} w_y^\top f(x)$

Different Weights vs. Different Features

- Different features: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$
  - Suppose $\mathcal{Y}$ is a structured label space (part-of-speech tags for each word in a sentence). *f*(*x*,*y*) extracts features over shared parts of these
- Different weights: $\text{argmax}_{y \in \mathcal{Y}} w_y^\top f(x)$
  - Generalizes to neural networks: *f*(*x*) is the first *n*-1 layers of the network, then you multiply by a final linear layer at the end
- For linear multiclass classification with discrete classes, these are identical

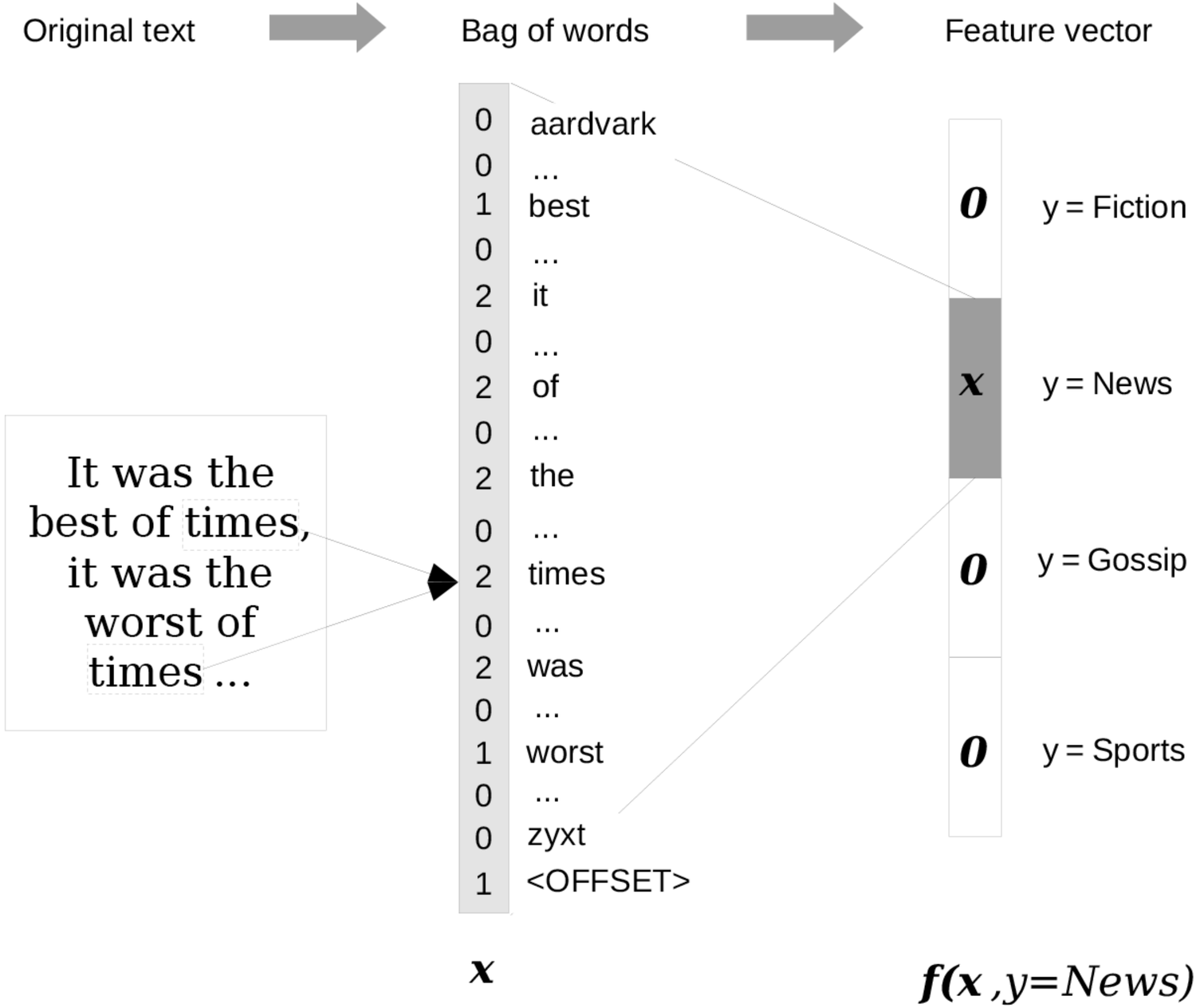


Figure 2.1
The bag-of-words and feature vector representations, for a hypothetical text classification task.

Feature Extraction

Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

too many drug trials, too few patients → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

feature vector blocks for each label

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

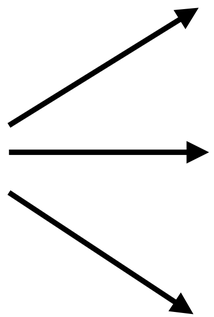
I[contains *drug* & label = Health]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

- Equivalent to having three weight vectors in this case
- We are NOT looking at the gold label! Instead looking at the candidate label

Making Decisions

too many drug trials, too few patients



Health

Sports

Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y = \text{Health}) = [1, 1, 0, 0, 0, 0, 0, 0, 0]$

$f(x, y = \text{Sports}) = [0, 0, 0, 1, 1, 0, 0, 0, 0]$

"word drug in Science article" = +1.1

$w = [+2.1, +2.3, -5, -2.1, -3.8, +5.2, +1.1, -1.7, -1.3]$

$w^\top f(x, y)$ = Health: +4.4 Sports: -5.9 Science: -0.6

argmax

Feature Representation Revisited

this movie was great! would watch again Positive

[contains *the*] [contains *a*] [contains *was*] [contains *movie*] [contains *film*] ...

position 0 position 1 position 2 position 3 position 4

- Bag-of-words features are position-insensitive
- What about for tasks like classifying a word as a given part-of-speech?

this movie was great! would watch again

- Want features extracted with respect to this particular position
- [curr word = *was*], [prev word = *movie*], [next word = *great*].
  - How many features?

Multiclass POS tagging

- Classify *blocks* as one of 36 POS tags

the router *blocks* *the packets*

NNS
VBZ
NN
DT
…

- Example *x*: sentence with a word (in this case, *blocks*) highlighted

- Extract features with respect to this word:

$f(x, y=\text{VBZ}) = $ I[curr_word=*blocks* & tag = VBZ],
I[prev_word=*router* & tag = VBZ]
I[next_word=*the* & tag = VBZ]
I[curr_suffix=*s* & tag = VBZ]

not saying that *the* is tagged as VBZ! saying that *the* follows the VBZ word

- Next two lectures: sequence labeling!

Multiclass Logistic Regression

Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

- Compare to binary:

$$P(y=1|x) = \frac{\exp(w^\top f(x))}{1+\exp(w^\top f(x))}$$

negative class implicitly had *f*(*x*, *y*=0) = the zero vector

- exp/sum(exp): also called *softmax*
- Training: maximize

$$\mathcal{L}(x,y) = \sum_{j=1}^{n} \log P(y_j^*|x_j)$$

$$= \sum_{j=1}^{n} \left(w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y)) \right)$$

Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \frac{\sum_y f_i(x_j, y) \exp(w^\top f(x_j, y))}{\sum_y \exp(w^\top f(x_j, y))}$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y[f_i(x_j, y)]$$

gold feature value

model's expectation of feature value

Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

too many drug trials, too few patients

$f(x, y = \text{Health}) = [1, 1, 0, 0, 0, 0, 0, 0, 0]$

$f(x, y = \text{Sports}) = [0, 0, 0, 1, 1, 0, 0, 0, 0]$

y^* = Health

$P_w(y|x)$ = [0.2, 0.5, 0.3]
(made up values)

gradient:

[1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.2 [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.5 [0, 0, 0, 1, 1, 0, 0, 0, 0]
— 0.3 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.8, 0.8, 0, -0.5, -0.5, 0, -0.3, -0.3, 0]

Logistic Regression: Summary

- Model: $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$
- Inference: $\text{argmax}_y P_w(y|x)$
- Learning: gradient ascent on the discriminative log-likelihood

$$f(x,y^*) - \mathbb{E}_y[f(x,y)] = f(x,y^*) - \sum_y [P_w(y|x) f(x,y)]$$

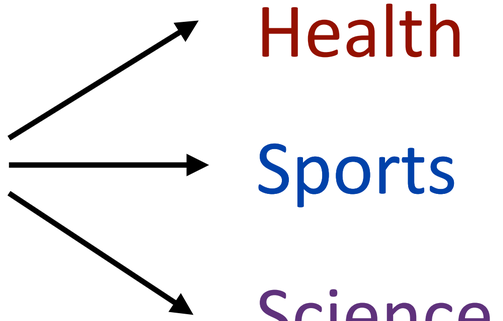
"towards gold feature value, away from expectation of feature value"

Multiclass SVM

Loss Functions

- Are all decisions equally costly?

too many drug trials, too few patients



Predicted Sports: bad error

Predicted Science: not so bad

- We can define a loss function $\ell(y, y^*)$

$\ell(\text{Sports}, \text{Health}) = 3$

$\ell(\text{Science}, \text{Health}) = 1$

Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

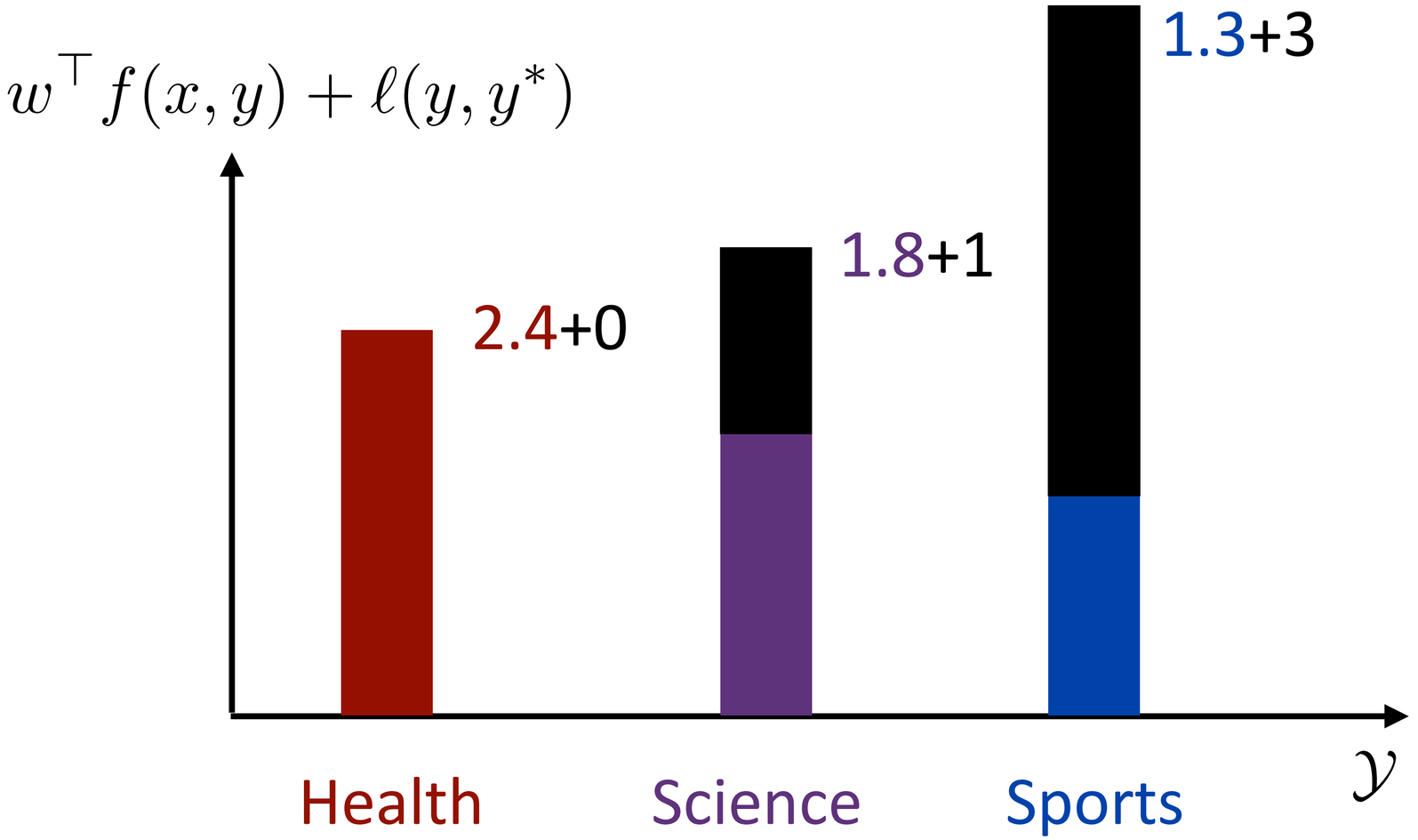
Correct prediction now has to beat every other class

Score comparison is more explicit now

The 1 that was here is replaced by a loss function

Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$



- Does gold beat every label + loss? No!
- Most violated constraint is Sports; what is ξ_j ?
- ξ_j = 4.3 - 2.4 = 1.9
- Perceptron would make no update here

Revisiting Generative vs. Discriminative Models

Learning in Probabilistic Models

- So far we have talked about discriminative classifiers (e.g., logistic regression which models P($y|x$))
- Cannot analytically compute optimal weights for such models, need to use gradient descent
- What about generative models?

Naive Bayes

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

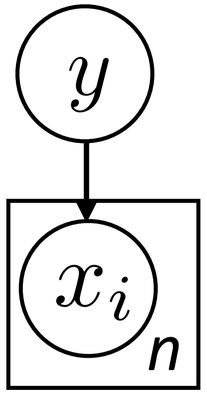
Bayes' Rule

constant: irrelevant for finding the max

$$\propto P(y)P(x|y)$$

"Naive" assumption:

$$= P(y)\prod_{i=1}^{n} P(x_i|y)$$



Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

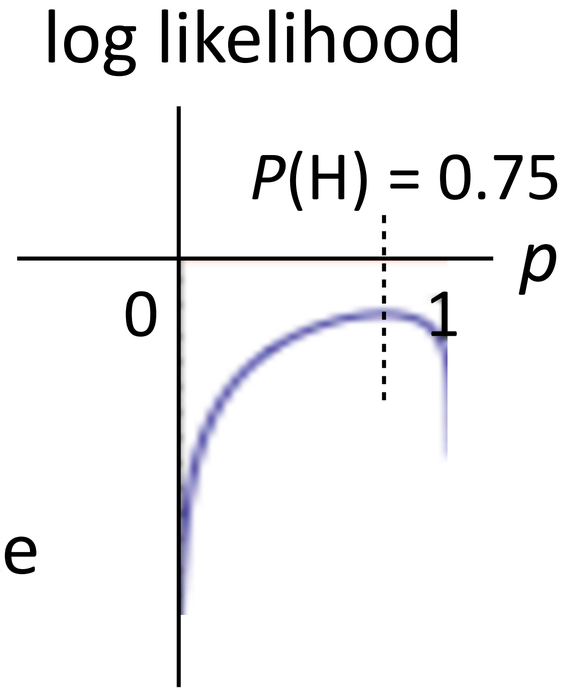
$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right]$$

data points (*j*) features (*i*) *i*th feature of *j*th example

Maximum Likelihood Estimation

- Imagine a coin flip which is heads with probability *p*
- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$
- Easier: maximize *log* likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3 \log p + \log(1-p)$$



- Maximum likelihood parameters for binomial/ multinomial = read counts off of the data + normalize

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right]$$

data points (*j*) features (*i*) *i*th feature of *j*th example

- Equivalent to maximizing logarithm of data likelihood:

$$\sum_{j=1}^{m} \log P(y_j, x_j) = \sum_{j=1}^{m} \left[\log P(y_j) + \sum_{i=1}^{n} \log P(x_{ji}|y_j) \right]$$

- Can do this by counting and normalizing distributions!

Classification Evaluation

Precision, Recall, F-Measure

Confusion matrix

- For each pair of classes <c_1,c_2> how many documents from c_1 were incorrectly assigned to c_2?
  - $c_{3,2}$: 90 wheat documents incorrectly assigned to poultry

Docs in test set	Assigned UK	Assigned poultry	Assigned wheat	Assigned coffee	Assigned interest	Assigned trade
True UK	95	1	13	0	1	0
True poultry	0	1	0	0	0	0
True wheat	10	90	0	1	0	0
True coffee	0	0	0	34	3	7
True interest	-	1	2	13	26	5
True trade	0	0	2	14	5	10

Per class evaluation

Recall:

Fraction of docs in class *i* classified correctly: $\frac{c_{ii}}{\sum_j c_{ij}}$

Precision:

Fraction of docs assigned class *i* that are actually about class *i*: $\frac{c_{ii}}{\sum_j c_{ji}}$

Accuracy: (1 - error rate)

Fraction of docs classified correctly: $\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one?
- **Macroaveraging**: Compute performance for each class, then average.
- **Microaveraging**: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

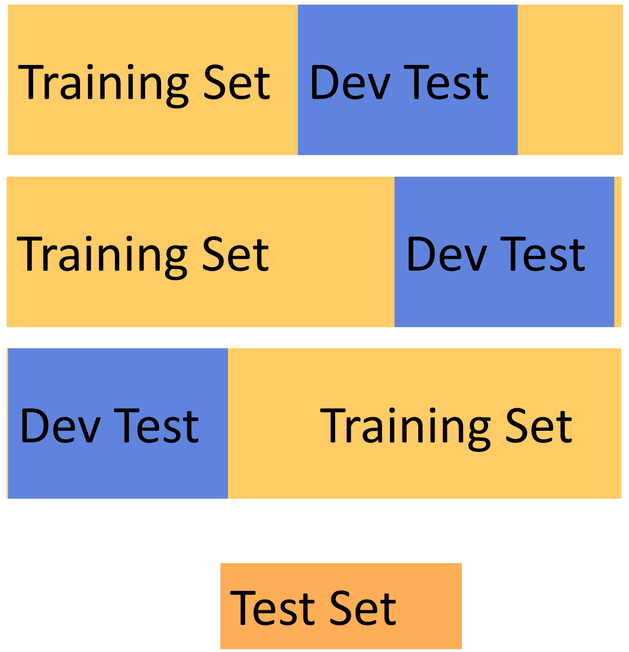
Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Microaveraged score is dominated by score on common classes

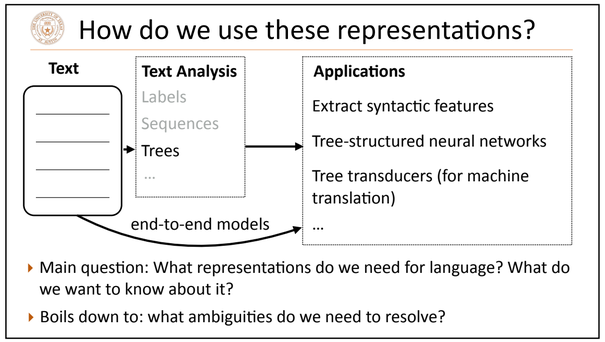
Crossvalidation

- Metric: P/R/F1 or Accuracy
- Unseen test set
  - avoid overfitting (‘tuning to the test set’)
  - more conservative estimate of performance
- Cross-validation over multiple splits
  - Handle sampling errors from different datasets
  - Pool results over each split
  - Compute pooled dev set performance



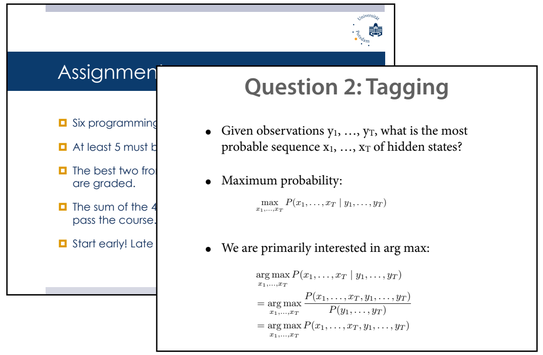
slide credits

slides that look like this



come from

CS388 given by Greg Durrett at U Texas, Austin



earlier editions of this class (ANLP), given by Tatjana Scheffler and Alexander Koller

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.