CS388: Natural Language Processing

Lecture 2: Binary Classification

credit: Machine Learning Memes on Facebook

Greg Durrett

TEXAS
The University of Texas at Austin

Some slides adapted from Vivek Srikumar, University of Utah

Administrivia

- Course enrollment
- Course website: slides, readings, office hours, syllabus
- Mini 1 out, due Tuesday
- Greg’s office hours on Thursday are rescheduled to 9am-10am

This Lecture

- Linear classification fundamentals

- Three discriminative models: logistic regression, perceptron, SVM
  - Different motivations but very similar update rules / inference!

- Optimization

- Sentiment analysis

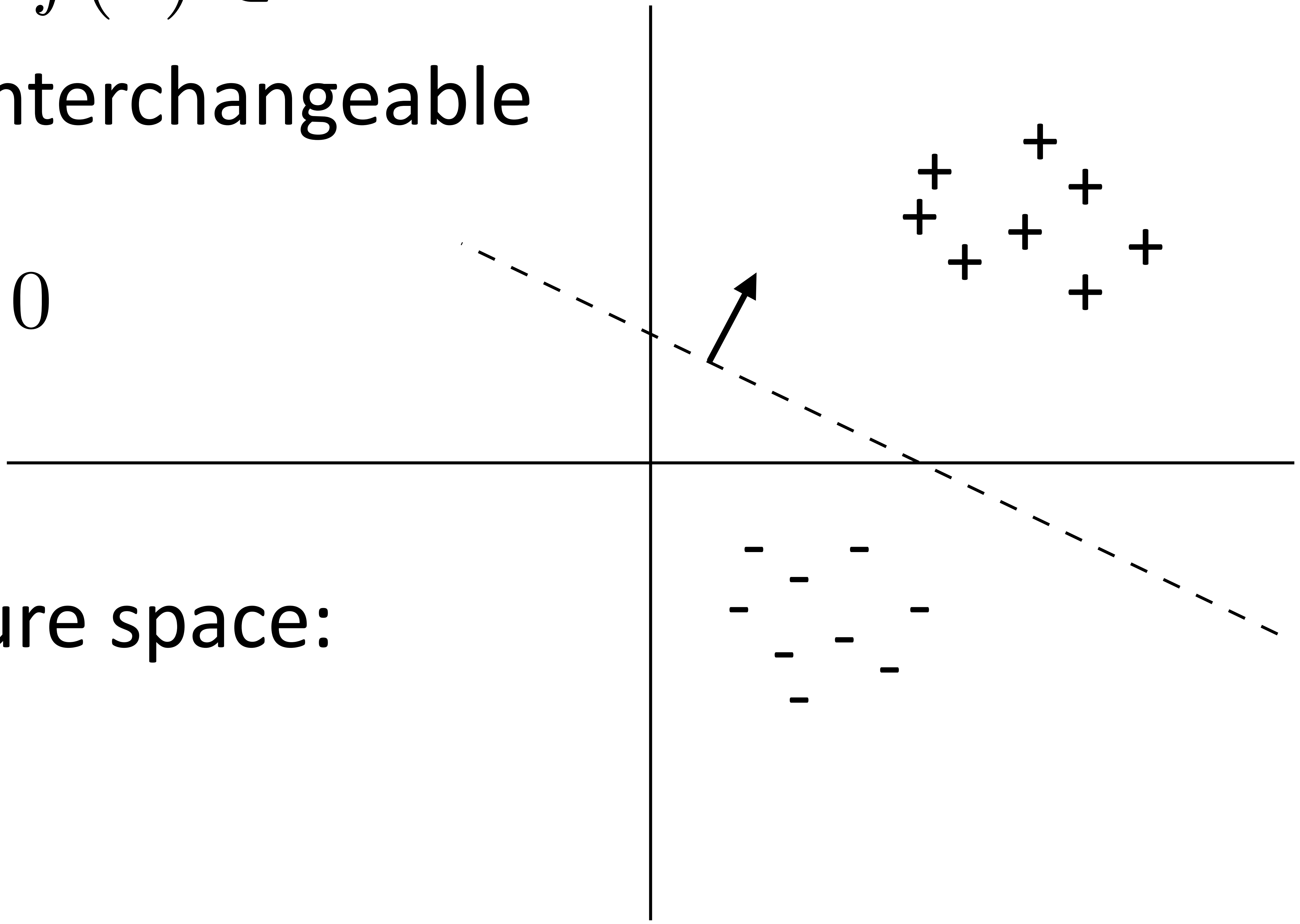
Classification

Classification

- Datapoint x with label $y \in \{0, 1\}$

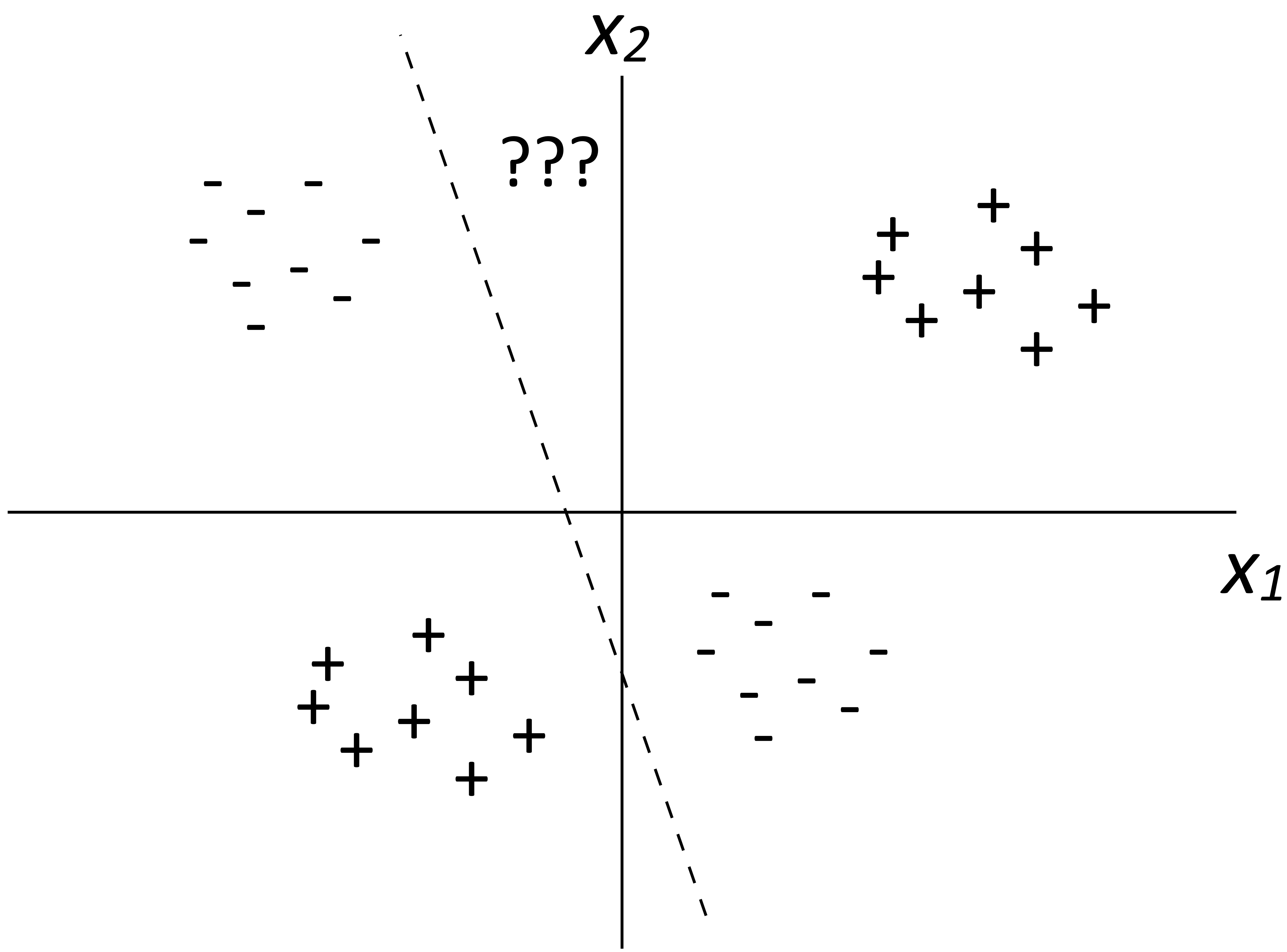
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
  but in this lecture $f(x)$ and x are interchangeable

- Linear decision rule: $w^\top f(x) + b > 0$
  $$w^\top f(x) > 0$$

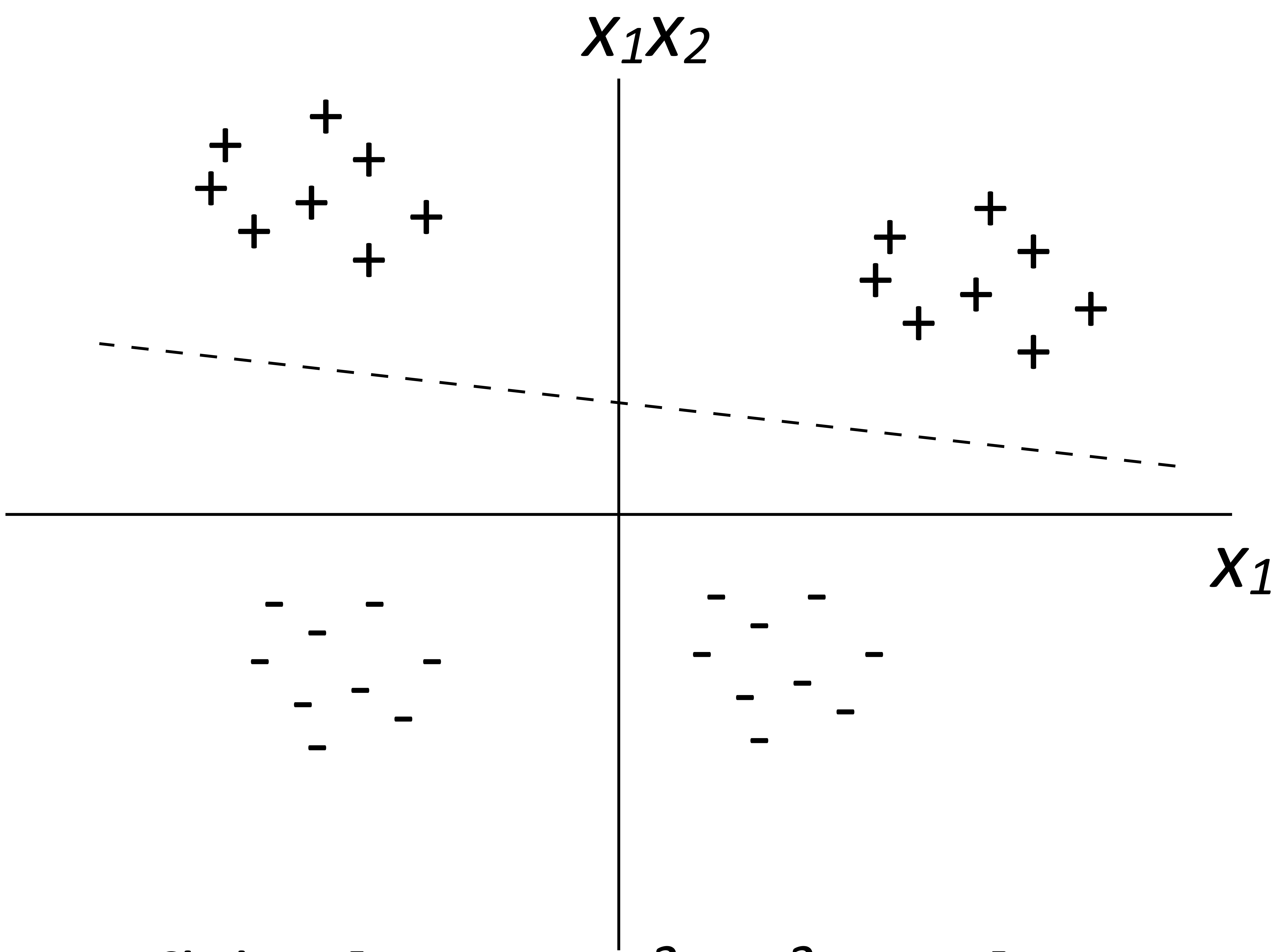


- Can delete bias if we augment feature space:
  $f(x)$ = [0.5, 1.6, 0.3]
  ↓
  [0.5, 1.6, 0.3, **1**]

Linear functions are powerful!



$f(x) = [x_1, x_2]$



$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

- "Kernel trick" does this for "free," but is too expensive to use in NLP applications, training is $O(n^2)$ instead of $O(n \cdot (\text{num feats}))$

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here: presence or absence of certain words (*great*, *awful*)
- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

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

position 0 position 1 position 2 position 3 position 4

$f(x)$ = [0 0 1 1 0 …

- Very large vector space (size of vocabulary), sparse features (how many?)
- Requires *indexing* the features (mapping them to axes)
- More sophisticated feature mappings possible (tf-idf), as well as lots of other features: n-grams, character n-grams, parts of speech, lemmas, …

Generative vs. Discriminative Modeling

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Generative models: probabilistic models of P(x,y)
  - Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

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

"proportional to"

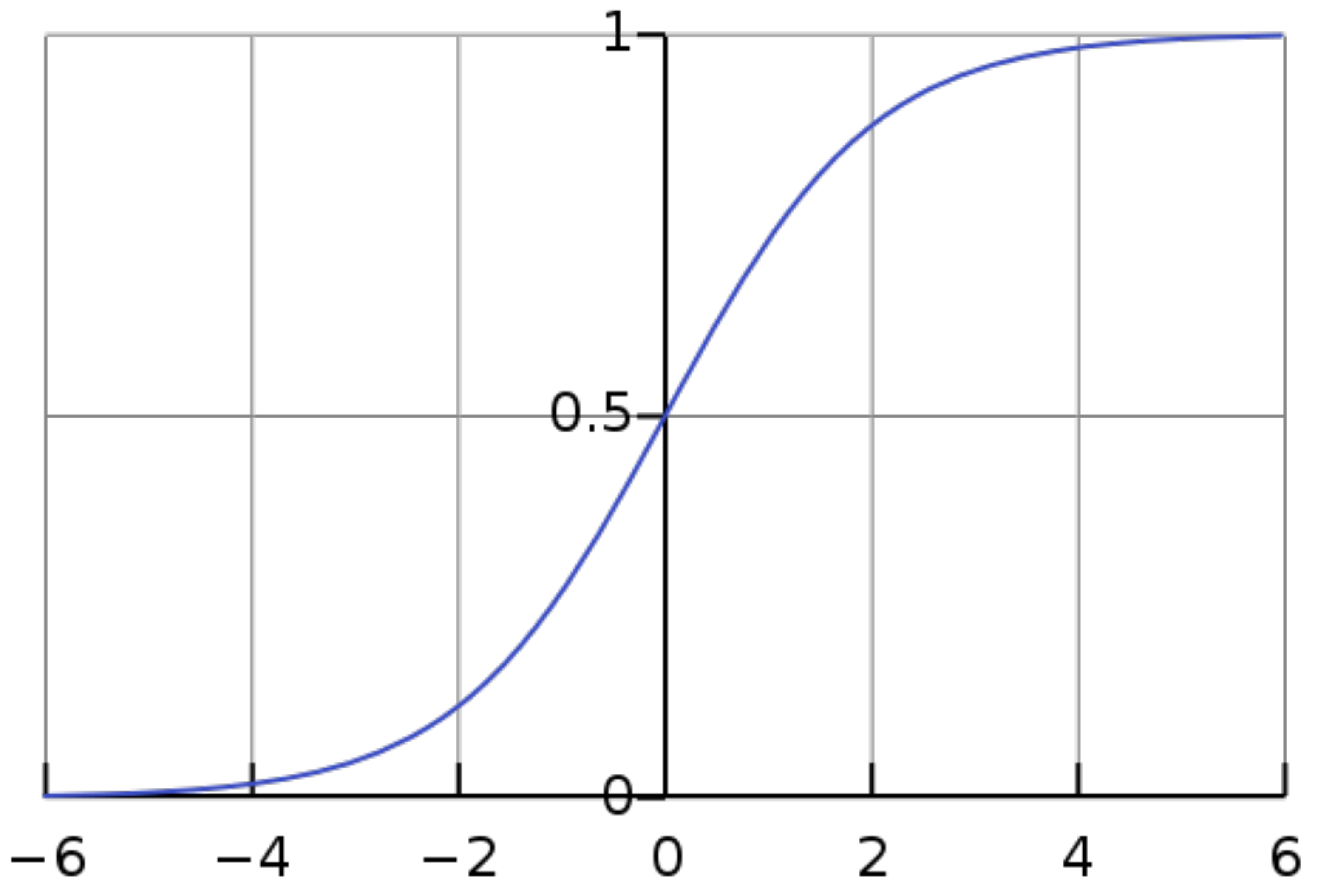
  - Examples: Naive Bayes (see textbook), Hidden Markov Models
- Discriminative models model P(y|x) directly, compute $\text{argmax}_y P(y|x)$
  - Examples: logistic regression
  - Cannot draw samples of *x*, but typically better classifiers

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

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

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

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

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* —

$f(x_1) = [1 \quad 1]$

$f(x_2) = [1 \quad 1]$

$f(x_3) = [1 \quad 0]$

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

position 0 position 1

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

$w = [0.5, 0.5] \longrightarrow P(y = 1 \mid x_2) = \text{logistic}(1) \approx 0.75 \longrightarrow g = [0.25, 0.25]$

$w = [0.75, 0.75] \rightarrow P(y = 1 \mid x_3) = \text{logistic}(0.75) \approx 0.67 \longrightarrow g = [-0.67, 0]$

$w = [0.08, 0.75]$ …

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

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

Regularization

- 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

Perceptron/SVM

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

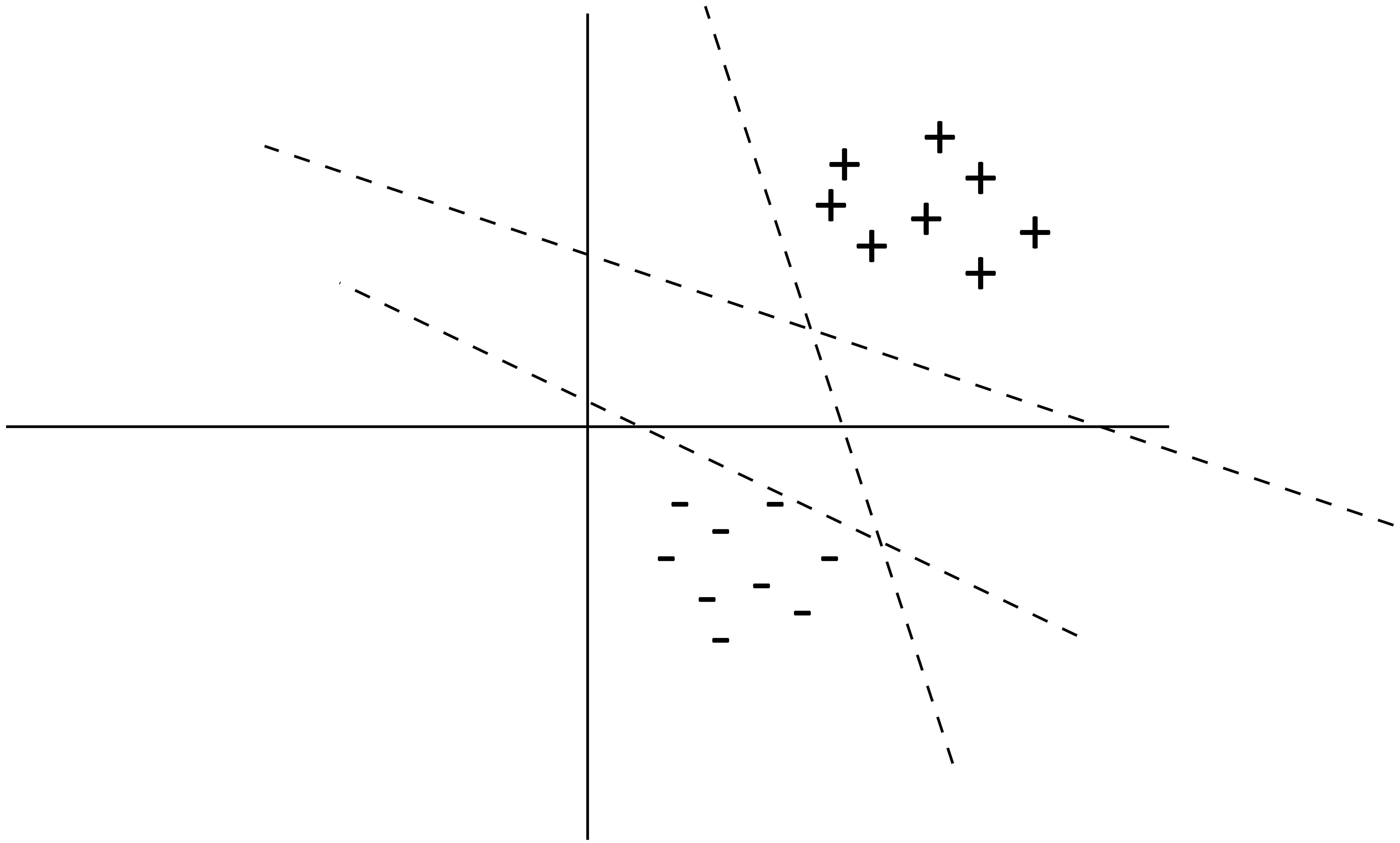
$$w \leftarrow w + x(1 - P(y = 1|x))$$

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

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

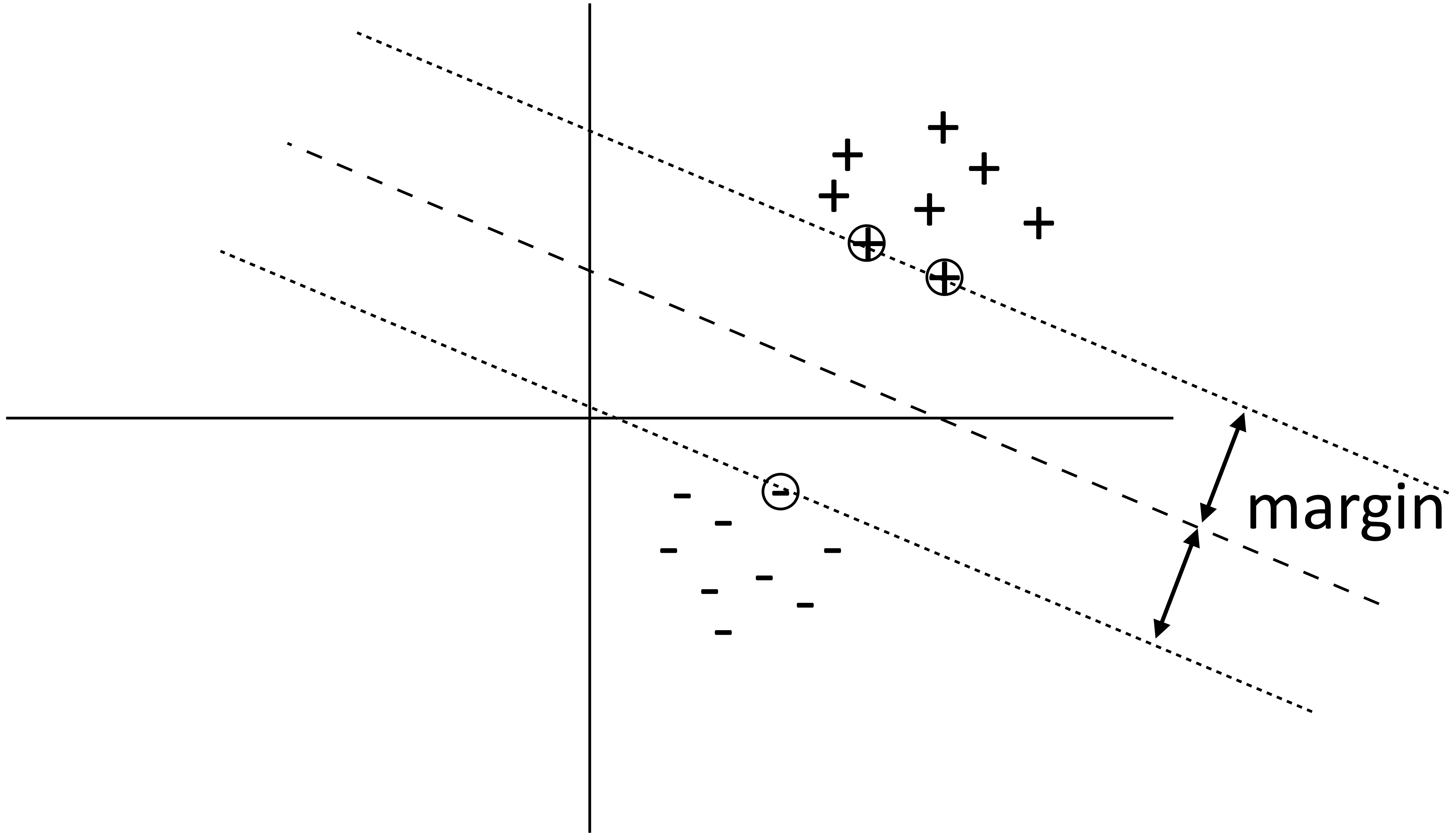
Support Vector Machines

- 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:

$$\boxed{\begin{aligned} &\text{Minimize } \|w\|_2^2 \\ &\text{s.t. } \forall j \quad w^\top x_j \geq 1 \text{ if } y_j = 1 \\ &\qquad\qquad w^\top x_j \leq -1 \text{ if } y_j = 0 \end{aligned}}$$

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

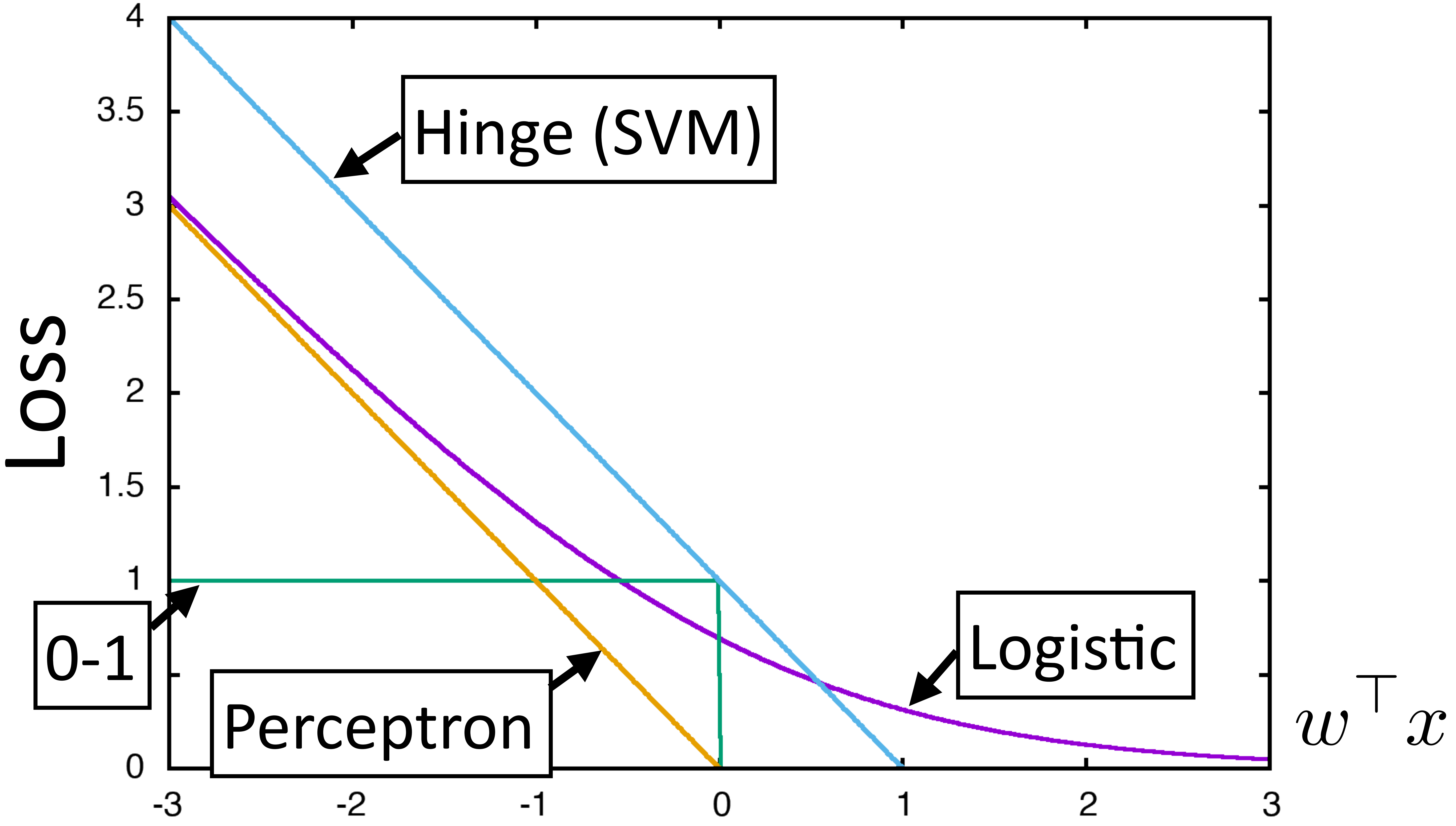
$$x(1 - \text{logistic}(w^\top x))$$

Perceptron

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

SVM (ignoring regularizer)

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



*gradients are for maximizing things, which is why they are flipped

Comparing Gradient Updates (Reference)

Logistic regression (unregularized)

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

y = 1 for pos,

0 for neg

Perceptron

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

0 else

SVM

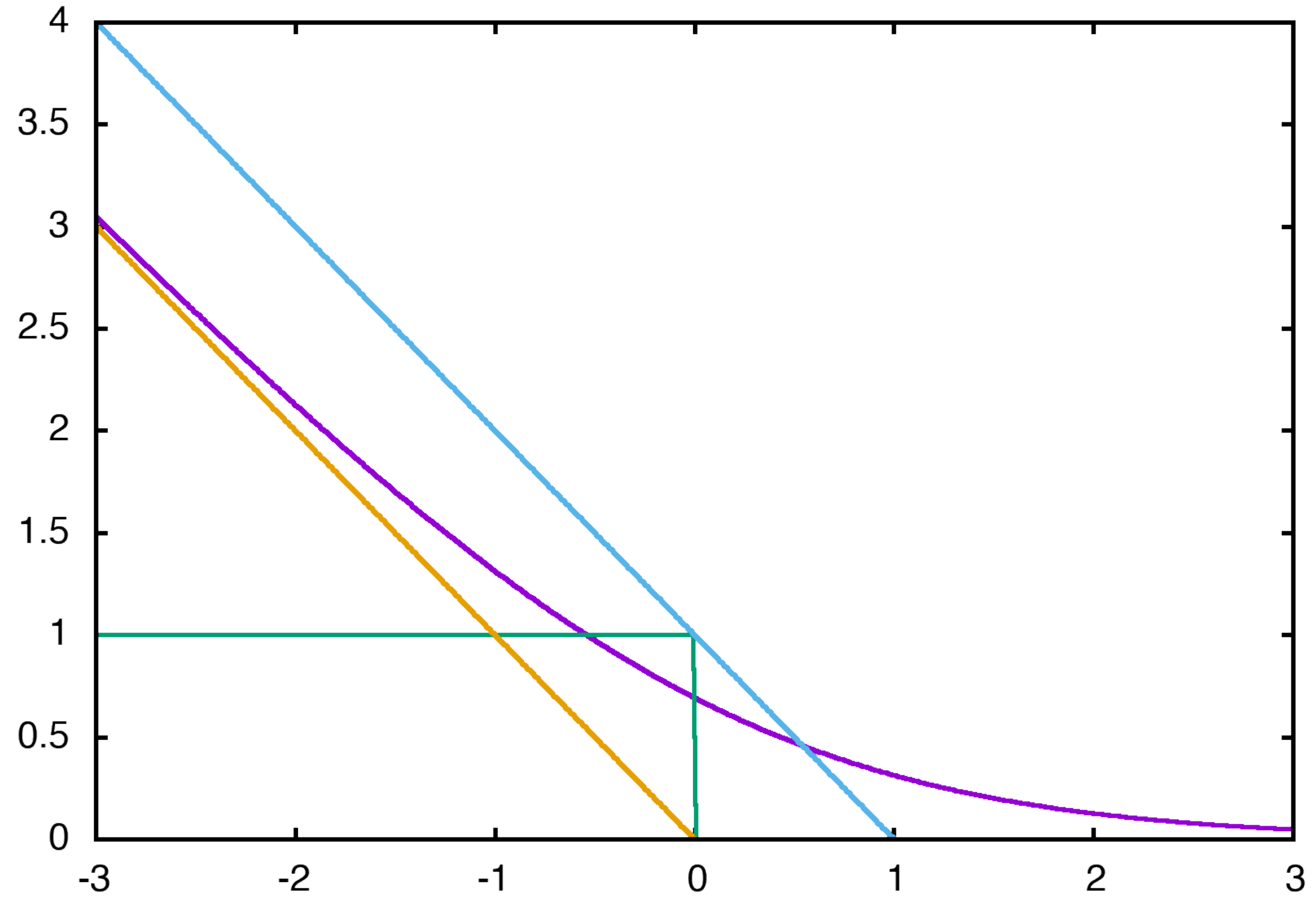
$(2y-1)x$ if not classified correctly with margin of 1

0 else

Optimization

Structured Prediction

- Four elements of a structured machine learning method:
  - Model: probabilistic, max-margin, deep neural network
  - Objective



  - Inference: just maxes and simple expectations so far, but will get harder
  - Training: gradient descent?

Optimization

- Stochastic gradient *ascent*

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

  - 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?)
- Newton's method

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

  - Second-order technique
  - Optimizes quadratic instantly

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

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

AdaGrad

- Optimized for problems with sparse features
- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

$$w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t_i}$$

(smoothed) sum of squared gradients from all updates

- Generally more robust than SGD, requires less tuning of learning rate
- Other techniques for optimizing deep models — more later!

Duchi et al. (2011)

Implementation

- Supposing *k* active features on an instance, gradient is only nonzero on *k* dimensions

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

- *k* < 100, total num features = 1M+ on many problems
- Be smart about applying updates!
- In PyTorch: applying sparse gradients only works for certain optimizers and sparse updates are very slow. The code we give you is much faster

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

  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 time: multiclass classification