# CS 378: Natural Language Processing

# Lecture 2: Text Classification

The University of Texas at Austin

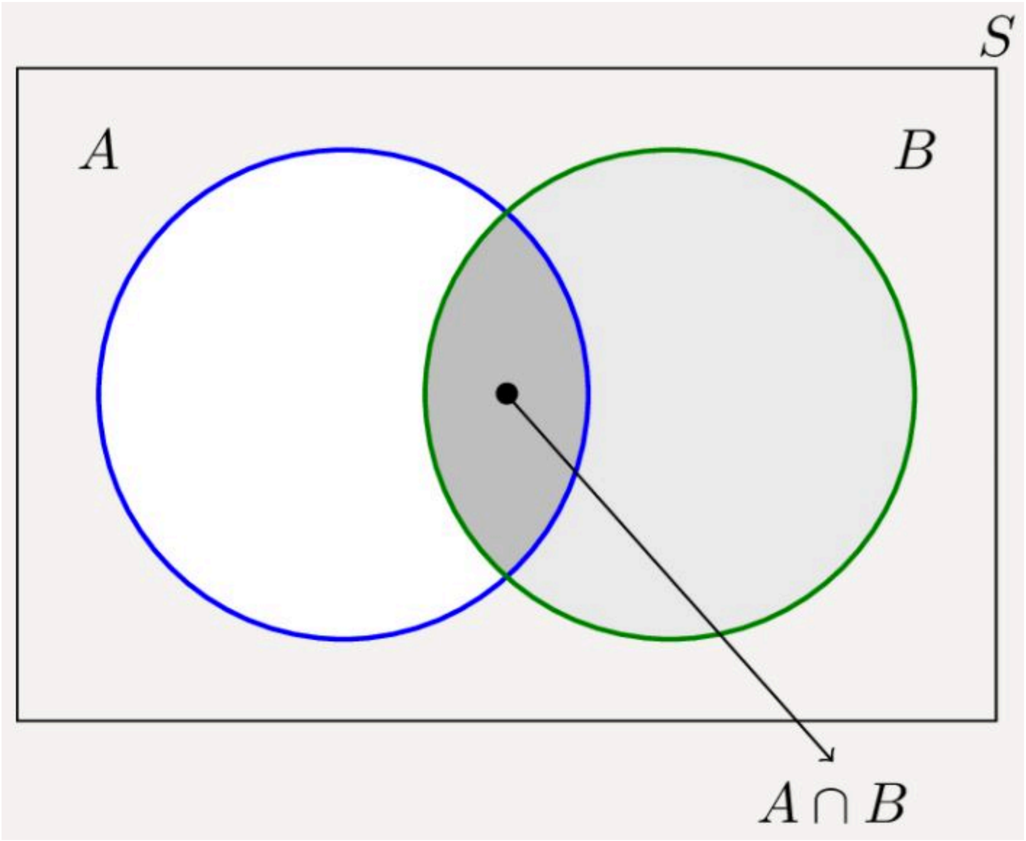
Eunsol Choi

Slides adapted from Yoav Artzi, Greg Durrett

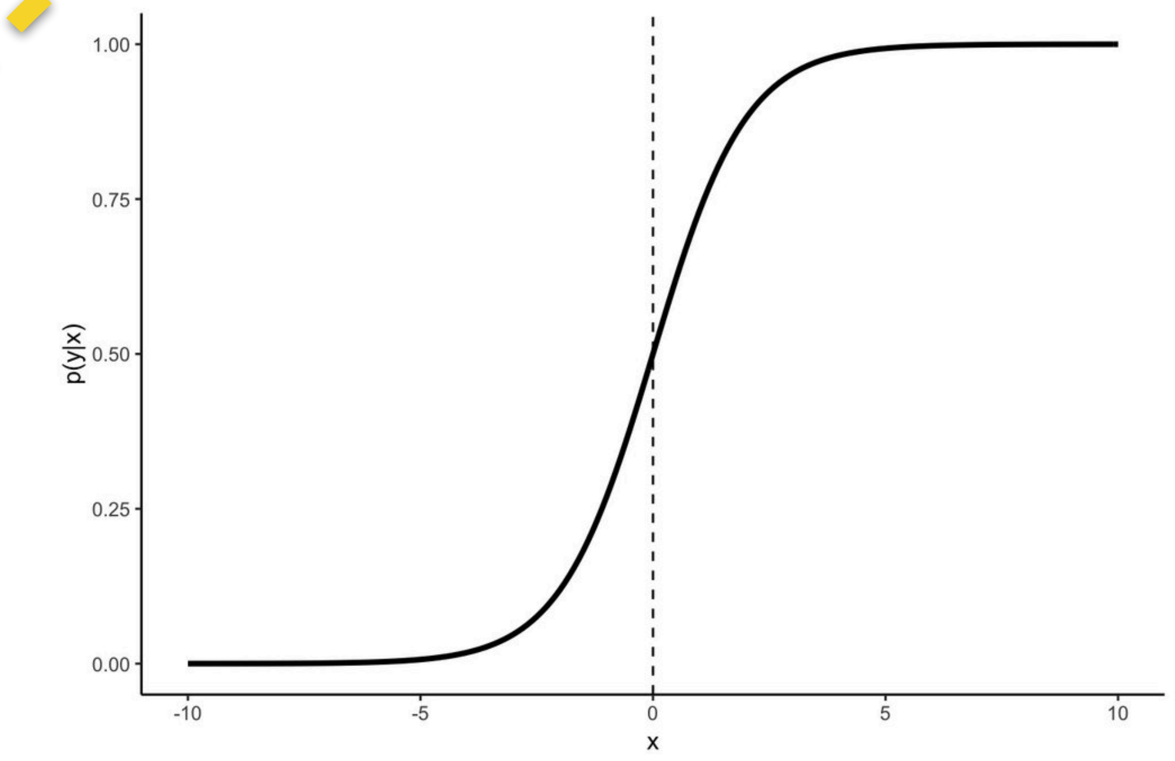
# Overview

- Classification Problem
- Learning a classifier
  - Naive Bayes Classifier
  - **Log-linear classifier (maximum entropy models)**
  - Perceptron
  - Feedforward Neural Network

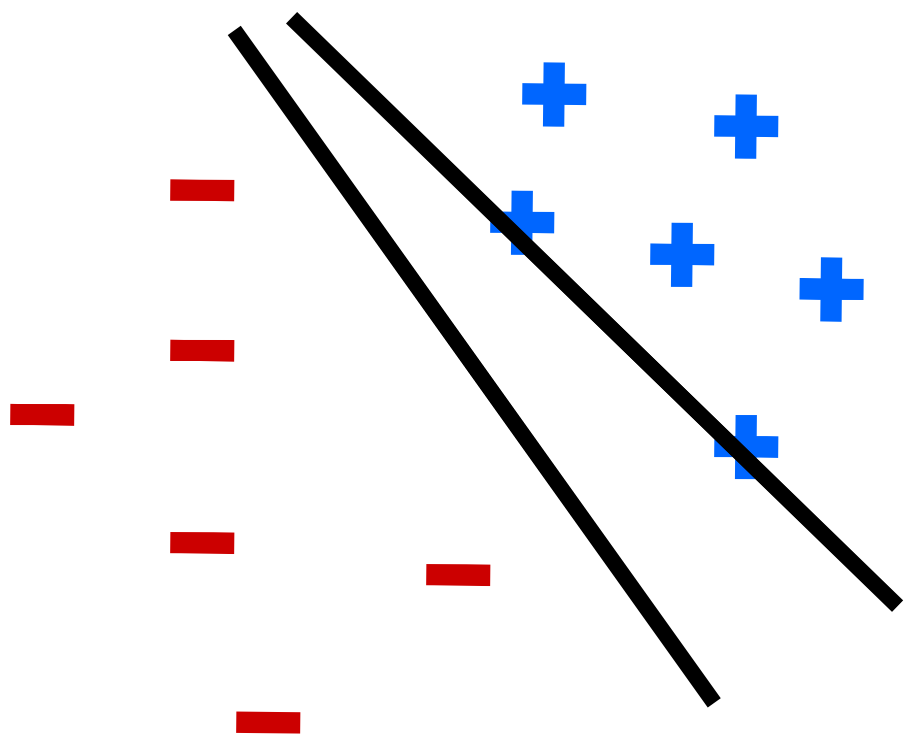
# Types of Supervised Classifiers




Naive Bayes




Logistic regression



Perceptron

Many others....

# Log Linear Model

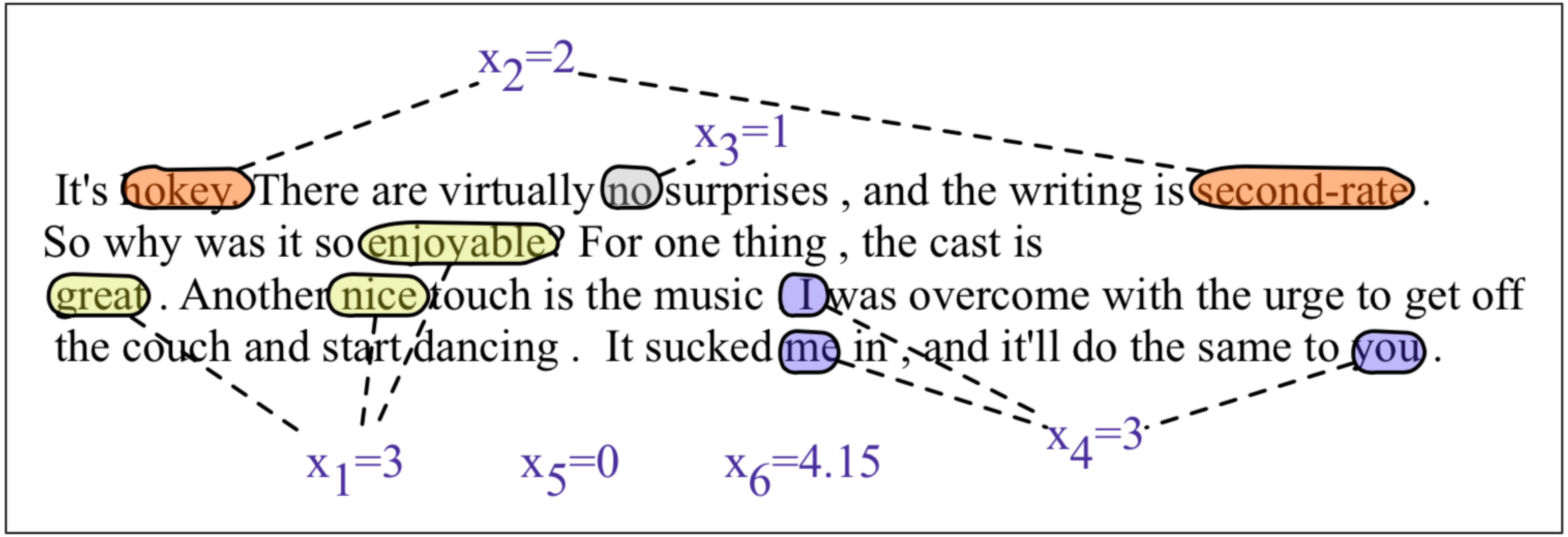
- Inputs:
  - Represent input in a feature representation
  - Classification function to compute $y^*$ using $P(y \mid x)$
  - Loss function (for learning)
  - Optimization algorithm
- Training: Learn the parameters of the model ($w$) to minimize loss function
- Test: Apply parameters to predict class given a new input x

# Features

- Mapping an input to a fixed dimensional vector

- You get inspiration from the data.
- Can incorporate linguistic intuitions / domain expertise
- Can use complex rules

# Features: Sentiment Classification




| Var | Definition | Value in Fig. 5.2 |
|---|---|---|
| $x_1$ | count(positive lexicon) $\in$ doc) | 3 |
| $x_2$ | count(negative lexicon) $\in$ doc) | 2 |
| $x_3$ | $\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 1 |
| $x_4$ | count(1st and 2nd pronouns $\in$ doc) | 3 |
| $x_5$ | $\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 0 |
| $x_6$ | log(word count of doc) | $\ln(64) = 4.15$ |



Fig 5.2 in J&R

# Classification Function

- Given an input feature vector: $\phi(x) \in R^d$
- Output: $p(y = 1 \mid x, w)$

Given input x and weight vector w, what is p(y=1)?

- Learning a scoring function: $f : R^d \rightarrow [0,1]$
- Weight vector: $w \in R^d$

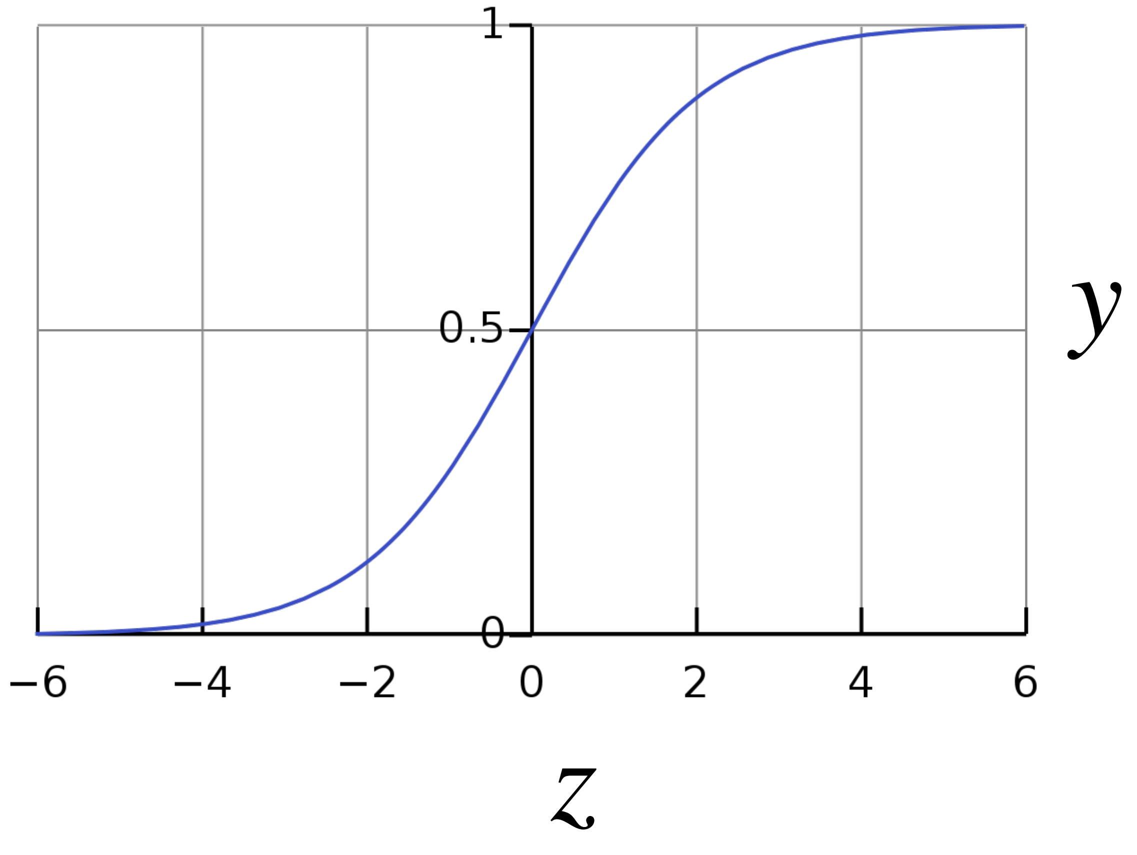
$$z = w \cdot \phi(x)$$

$$z = w \cdot \phi(x) + b$$

- In addition to weights, we often introduce bias (scalar intercept) to compute score.
- For most of this lecture, we omit this scalar bias for simplicity.

- Sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$




$$p(y = 1 \mid x, w) = \sigma(w \cdot \phi(x))$$

# Binary Classification

- Probabilities:

$$P(y = 1 \mid x, w) = \frac{1}{1 + e^{-(w \cdot \phi(x))}}$$

- Decision Boundary:

$$y^* = \text{argmax}_{y \in 0,1} p(y \mid x, w)$$

# Example: Sentiment Classification

| Var | Definition | Value in Fig. 5.2 |
|---|---|---|
| $x_1$ | $\text{count}(\text{positive lexicon}) \in \text{doc})$ | 3 |
| $x_2$ | $\text{count}(\text{negative lexicon}) \in \text{doc})$ | 2 |
| $x_3$ | $\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 1 |
| $x_4$ | $\text{count}(1\text{st and 2nd pronouns} \in \text{doc})$ | 3 |
| $x_5$ | $\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 0 |
| $x_6$ | $\log(\text{word count of doc})$ | $\ln(64) = 4.15$ |

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

$$
\begin{aligned}
p(+|x) = P(Y = 1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
&= \sigma(.805) \\
&= 0.69 \qquad (5.6) \\
p(-|x) = P(Y = 0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.31
\end{aligned}
$$

# Classification Function

- ***binary*** classifier (sigmoid):

Feature vector: $\phi(x)$

$$P(y=1 \mid x, w) = \frac{1}{1+e^{-(w \cdot \phi(x))}}$$

$$y^* = \text{argmax}_{y \in 0,1} p(y \mid x, w)$$

- ***multi class*** classifier (softmax):

$\phi(x, y)$

$$p(y \mid x, w) = \frac{e^{(w \cdot \phi(x,y))}}{\sum_{y'} e^{(w \cdot \phi(x,y'))}}$$

$$y^* = \text{argmax}_{y \in C} w \cdot \phi(x, y)$$

# Log Linear Model

- Inputs:
  - Represent input in a feature representation
  - Classification function to compute $y^*$ using $P(y|x)$
  - Loss function (for learning)
  - Optimization algorithm
- Training: Learn the parameters of the model to minimize loss function
- Test: Apply parameters to predict class given a new input x

# Log linear model: Classification function

- You compute the score for each label, and predict the label with the highest score

$$P(y=1 \mid x, w) = \frac{1}{1+e^{-(w \cdot \phi(x))}}$$

$$y^* = \textbf{argmax}_{y \in 0,1} p(y \mid x, w)$$

How do we learn the weight vector w?

$$\theta = [w]$$

# What is the best weights?

- Goal: Choose the weights that will give highest ***test*** set accuracy / F1
  - But we don't have an access to the test set

- Maybe choose the weights that gives highest ***training*** set accuracy / F1?
  - Optimization is not easy
  - May not generalize to test set

# Learning Objective

- Model: we use the scores to compute probabilities

$$p(y = 1 | x, w) = \frac{1}{(1 + e^{(-w \cdot \phi(x))})}$$

- Learning: maximize the **log conditional likelihood** of training data
  - How much predicted label distribution differs from the true label distribution?

$$L(w) = \log \prod_{i=1}^{N} p(y^i | x^i, w) = \sum_{i=1}^{N} \log(p(y^i | x^i, w))$$

$$w^* = \text{argmax}_w L(w)$$

# Cross Entropy Loss

- Minimize the discrepancy between the true distribution $P(y|x)$ and predicted distribution $P(y^*|x)$

$$H(p,q) = -E_p(\log q)$$

$$H(p(y|x), p(y^*|x)) = -\sum_y p(y|x)\log p(y^*|x)$$

$$L(w) = \log \prod_{i=1}^{N} p(y^i|x^i, w) = \sum_{i=1}^{N} \log(p(y^i|x^i, w))$$

$$w^* = \mathrm{argmax}_w L(w)$$

# Example: Loss

| Var | Definition | Value in Fig. 5.2 |
|---|---|---|
| $x_1$ | count(positive lexicon) $\in$ doc) | 3 |
| $x_2$ | count(negative lexicon) $\in$ doc) | 2 |
| $x_3$ | $\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 1 |
| $x_4$ | count(1st and 2nd pronouns $\in$ doc) | 3 |
| $x_5$ | $\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 0 |
| $x_6$ | log(word count of doc) | $\ln(64) = 4.15$ |

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

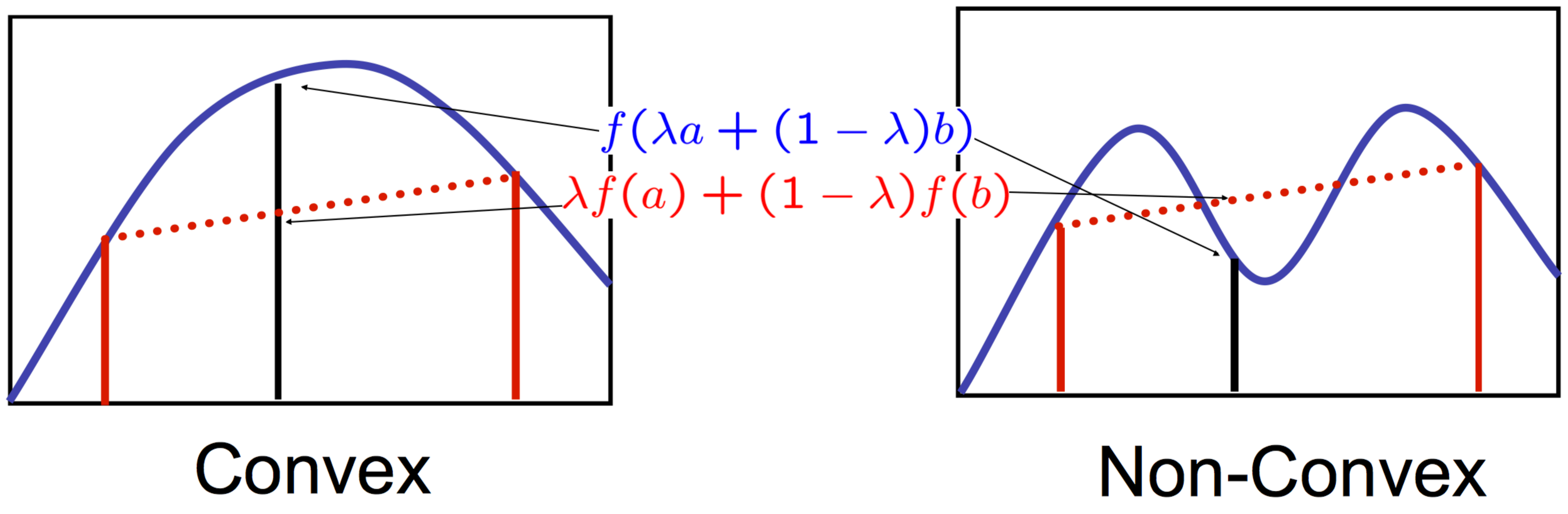
$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
&= \sigma(.805) \\
&= 0.69 \qquad (5.6) \\
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.31
\end{aligned}
$$

- If y = 1 (positive sentiment), $L(w) = \log(0.69) = -0.37$
- If y = 0 (negative sentiment), $L(w) = \log(0.31) = -1.17$

# Cross Entropy Loss

- Unfortunately, this does not have a closed form solution
- Differentiable
- Convex (local optima = global optima)
- Easy to optimize

$$f(\lambda a + (1-\lambda)b) \geq \lambda f(a) + (1-\lambda) f(b)$$



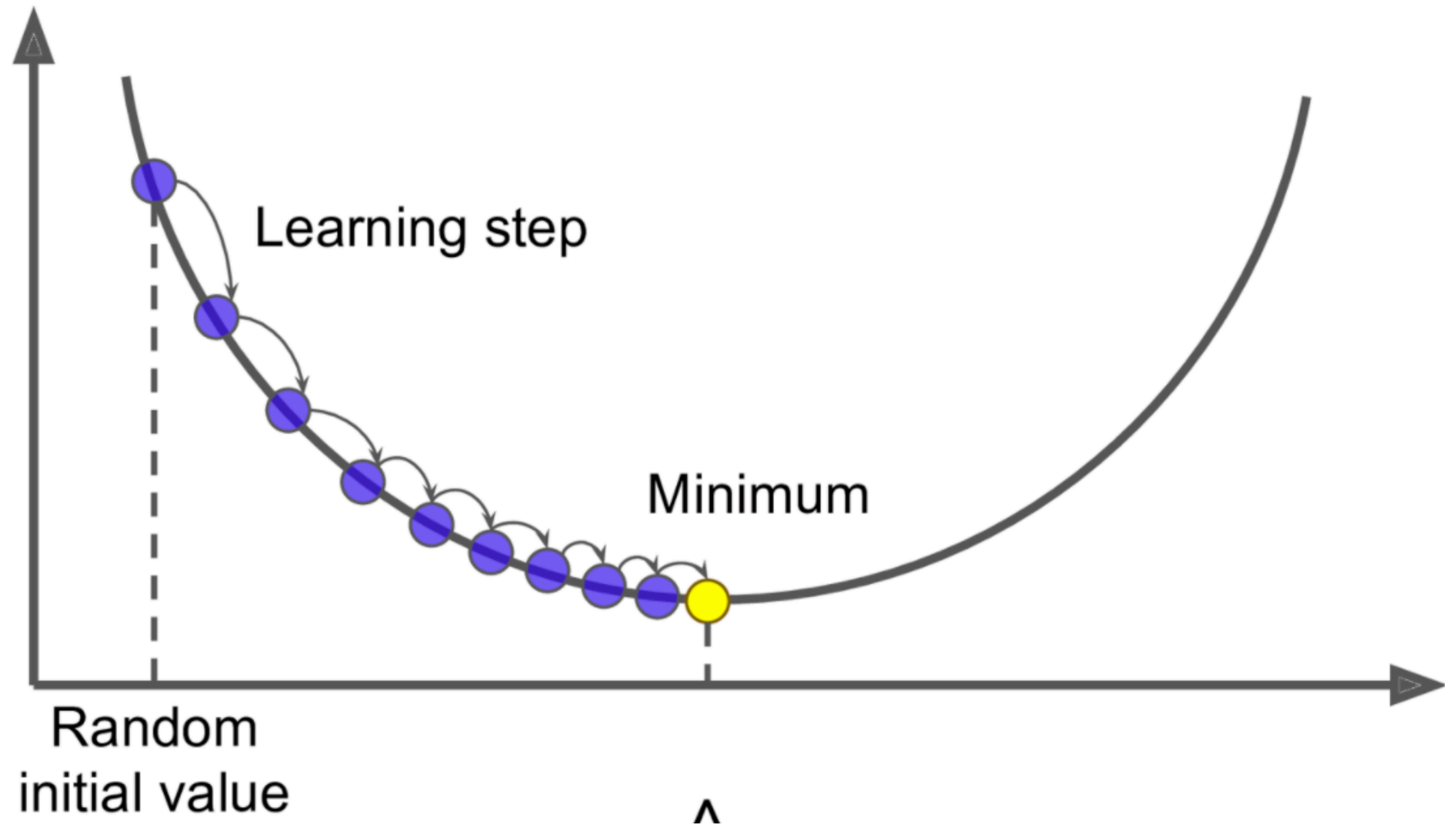
# Log Linear Model

- Inputs:
  - Represent input in a feature representation
  - Classification function to compute $y*$ using $P(y|x)$
  - Loss function (for learning)
  - Optimization algorithm
- Training: Learn the parameters of the model to minimize loss function
- Test: Apply parameters to predict class given a new input x

# Optimization

$$L(w) = \log \prod_{i=1}^{N} p(y^i \mid x^i; w) = \sum_{i=1}^{N} \log(p(y^i \mid x^i; w))$$

$$w^* = \text{argmax}_w L(w)$$




- Basic Idea:
  - Compute the derivate, move following the gradient incrementally
  - At local optimum, derivative will be zero
- Online learning algorithm: stochastic gradient descent/ascent
  - Compute loss and minimize after each training example

# Stochastic Gradient Descent

**function** STOCHASTIC GRADIENT DESCENT($L()$, $f()$, $x$, $y$) **returns** $\theta$

  # where: L is the loss function

  #    f is a function parameterized by $\theta$

  #    x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

  #    y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$

**repeat** til done  # see caption

 For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

  1. Optional (for reporting):  # How are we doing on this tuple?

   Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$  # What is our estimated output $\hat{y}$?

   Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$  # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?

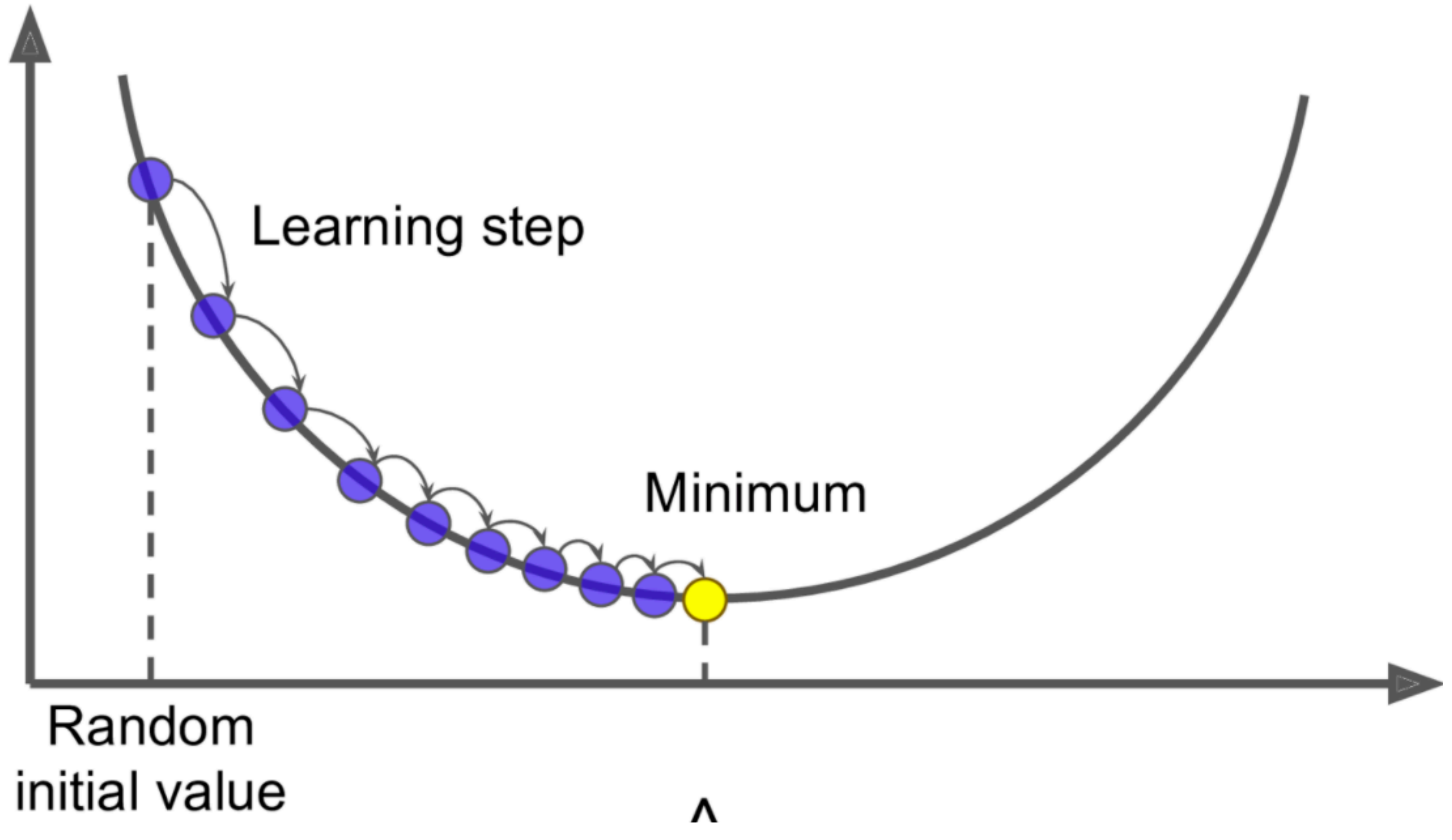
  2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$  # How should we move $\theta$ to maximize loss?

  3. $\theta \leftarrow \theta - \eta\, g$  # Go the other way instead

return $\theta$

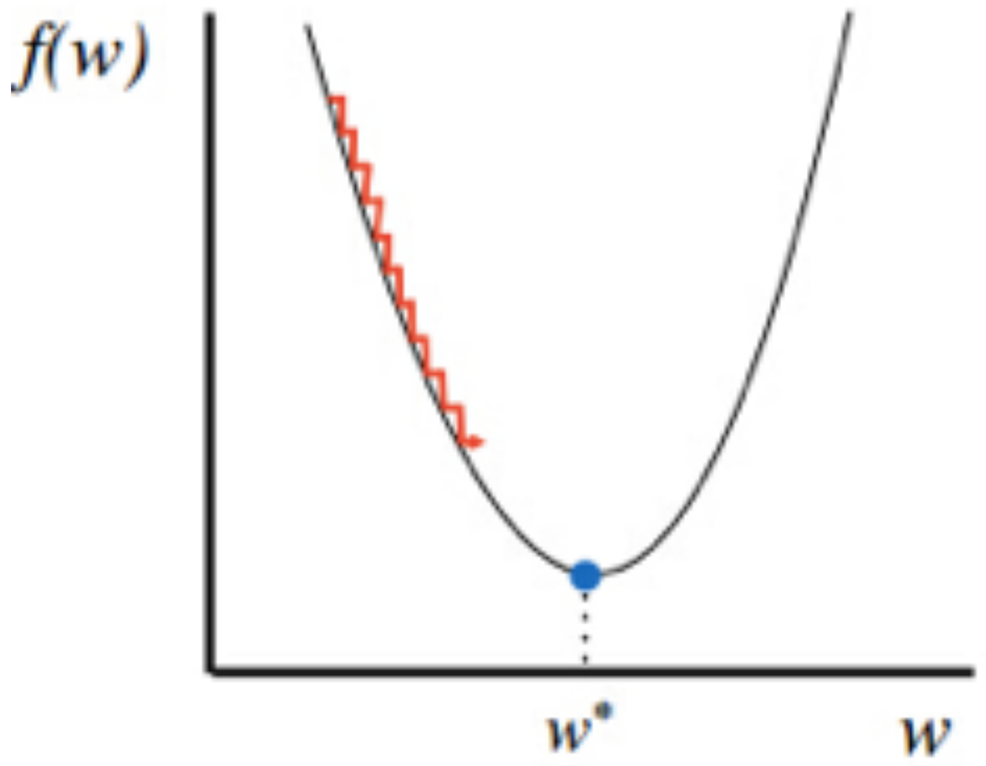
# Learning Rate

Updates: $w^{t+1} = w^t + \eta \frac{d}{dw} L(w)$

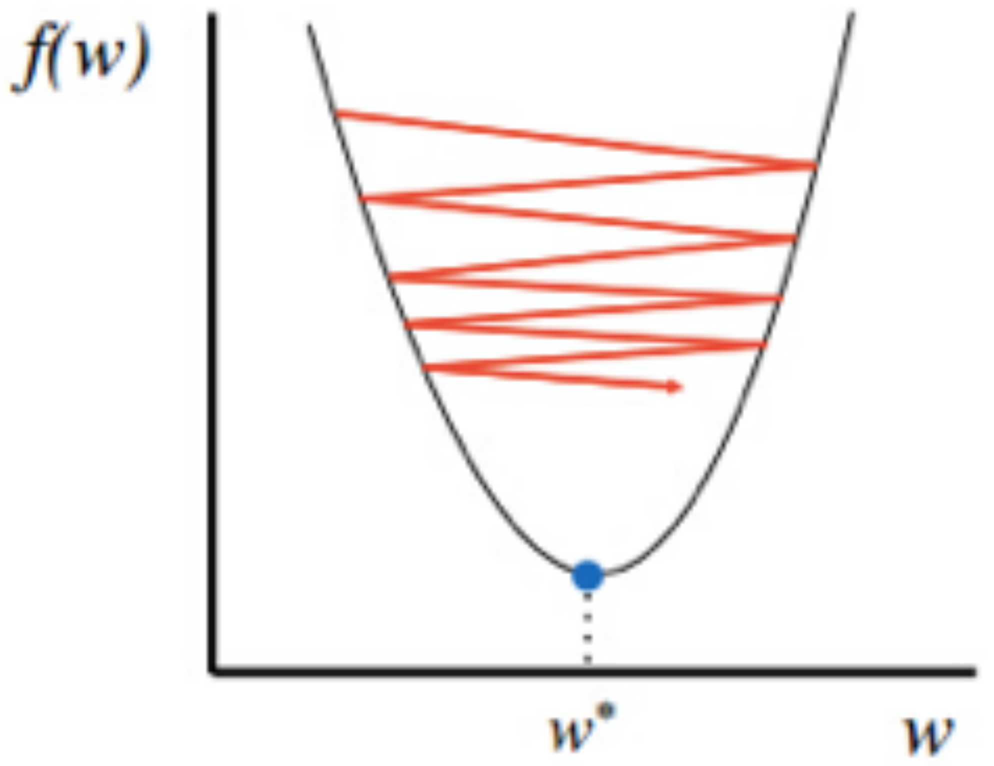



Higher/faster learning rate ($\eta$):

larger updates to parameters



Too small: converge very slowly



Too big: overshoot and even diverge

# Let's compute the gradient

$$L(w) = \log \prod_{i=1}^{N} p(y^i \mid x^i, w)$$

$$\hat{y} = \sigma(w \cdot \phi(x))$$

$$= \log \Pi_{i=1}^{n} P(y \mid x) = \log \Pi_{i=1}^{n} \hat{y}^{y} (1 - \hat{y})^{1-y}$$

$$= \sum_{i=1}^{n} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})]$$

$$= \sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot \phi(x^{(i)})) + (1 - y^{(i)}) \log (1 - \sigma(w \cdot \phi(x^{(i)})))]$$

Calculus!
(See J&M 5.10)

Correct label

$$\frac{dL(w)}{dw_j} = \sum_{i=1}^{n} [y^{(i)} - \sigma(w \cdot \phi(x^{(i)}))] \phi(x^i)_j$$

J-th input feature value

Predicted label

# Implementation

$$\frac{dL(w)}{dw_j} = \sum_{i=1}^{n} [y^{(i)} - \sigma(w \cdot \phi(x^{(i)}))]\phi(x^i)_j$$

- Supposing *k* active features on an instance, gradient is only nonzero on *k* dimensions
- *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 (assumes dense feature vectors)

# Overfitting

- Regularization:
  - Controlling large feature weights
  - Add a L2 regularization term to the likelihood, to push weights towards zero

$$L(w) = \log \prod_{i=1}^{N} p(y^i | x^i, w) - \frac{\lambda}{2} ||w||^2$$

$$\frac{dL(w)}{dw_j} = \sum_{i=1}^{n} [y^{(i)} - \sigma(w \cdot \phi(x^{(i)}))]\phi(x^i)_j \quad - \lambda w_j$$

Big weights are bad

# Logistic Regression: Multinomial

- Model: $$p(y|x;w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y'} \exp(w \cdot \phi(x,y'))}$$
- Inference: $$y^* = \text{argmax}_{y \in C} w \cdot \phi(x,y)$$
- Learning: gradient ascent on the discriminative log-likelihood

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{N} (\phi_j(x_i, y_i) - \sum_{y'} P(y'|x_i, w)\phi_j(x_i, y'))$$

Total count of feature j in correct candidate

Expected count of feature j in predicted candidate



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

# How well does this classifier work?

| | 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** |

Pretty comparable performances with Naive Bayes!



Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (2002)

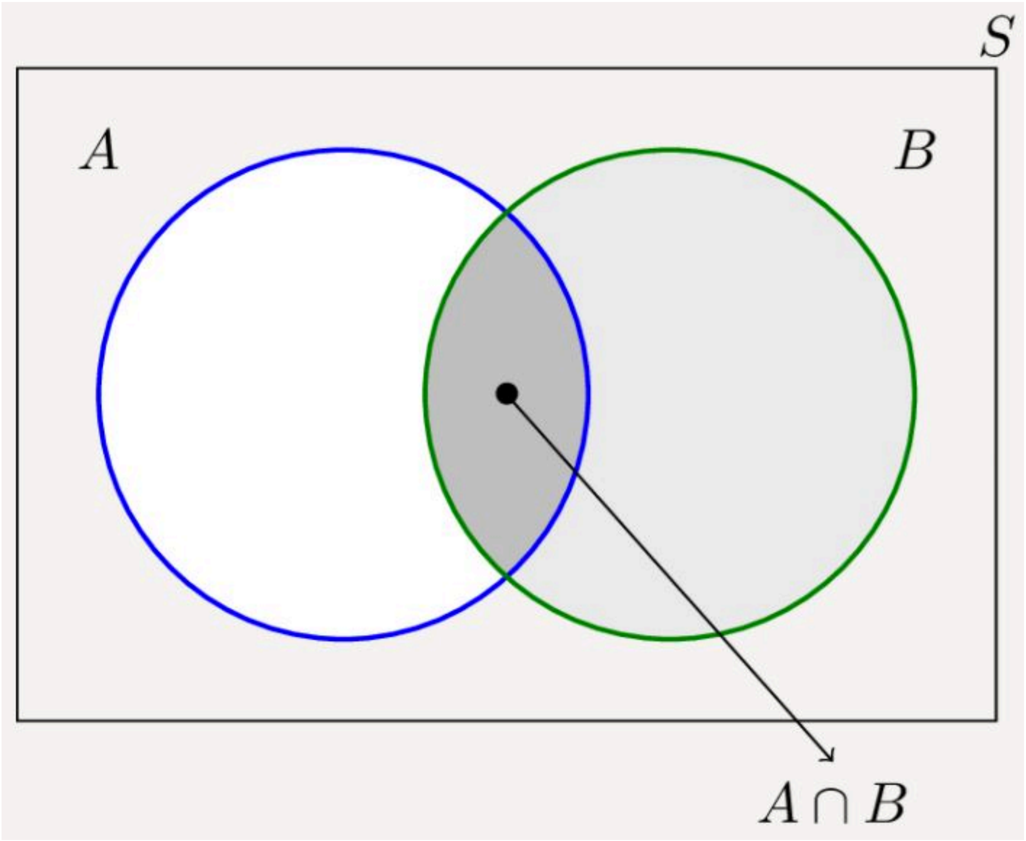
# Recap: Two Types of Classifiers

- Generative model (e.g., Naive Bayes)
  - Work with a joint probabilistic model of the data P(x,y)
  - Estimating **probabilities** from the data
  - Advantage: learning weights is easy and well understood
- Discriminative Models (e.g., logistic regression)
  - Work with conditional probability p(y|x)
  - Estimate **parameters** from data
  - Advantage: you do not model p(x). Can develop rich features for p(y|x)!

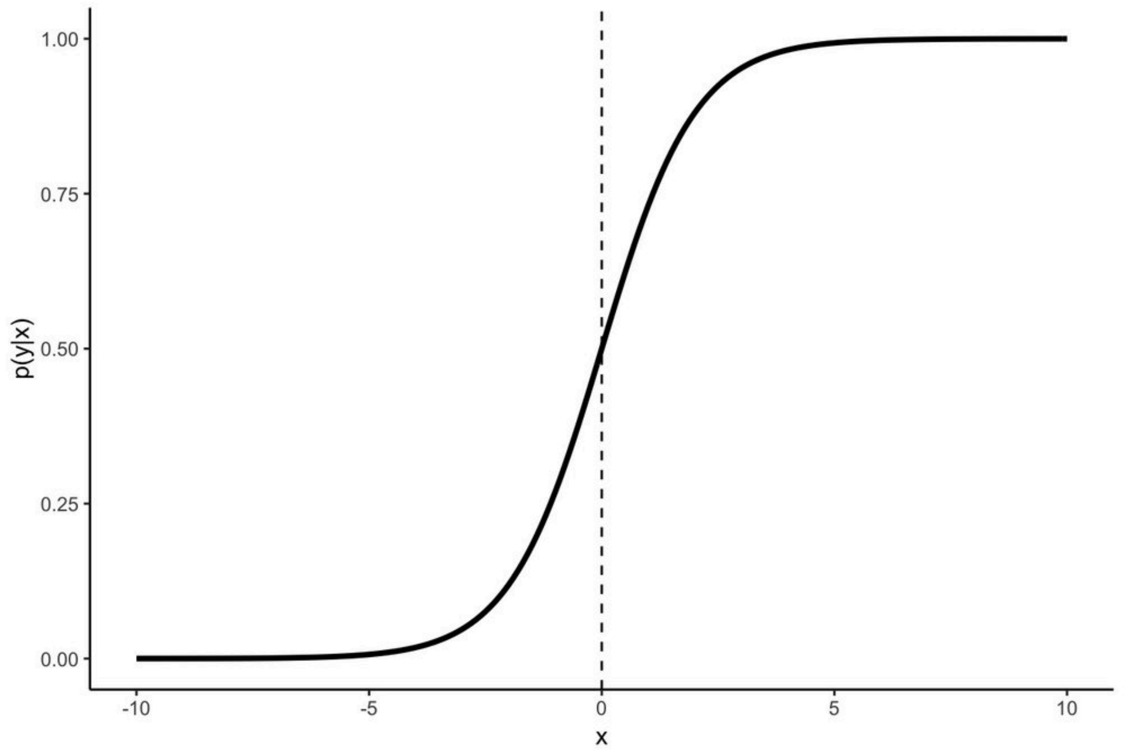
# Types of Supervised Classifiers

- Two probabilistic approaches for predicting classes
  - Joint: probabilistic model of the data, $y* = \text{argmax}_y(p(x, y))$
  - Conditional: can develop rich features $y* = \text{argmax}_y p(y \mid x)$
- Q: Do we have to estimate a probability distribution at all?
  - Linear predictor: $y* = \text{argmax}_y w \cdot \phi(x, y)$
  - Perceptron algorithm
    - Error driven, simple, additive updates

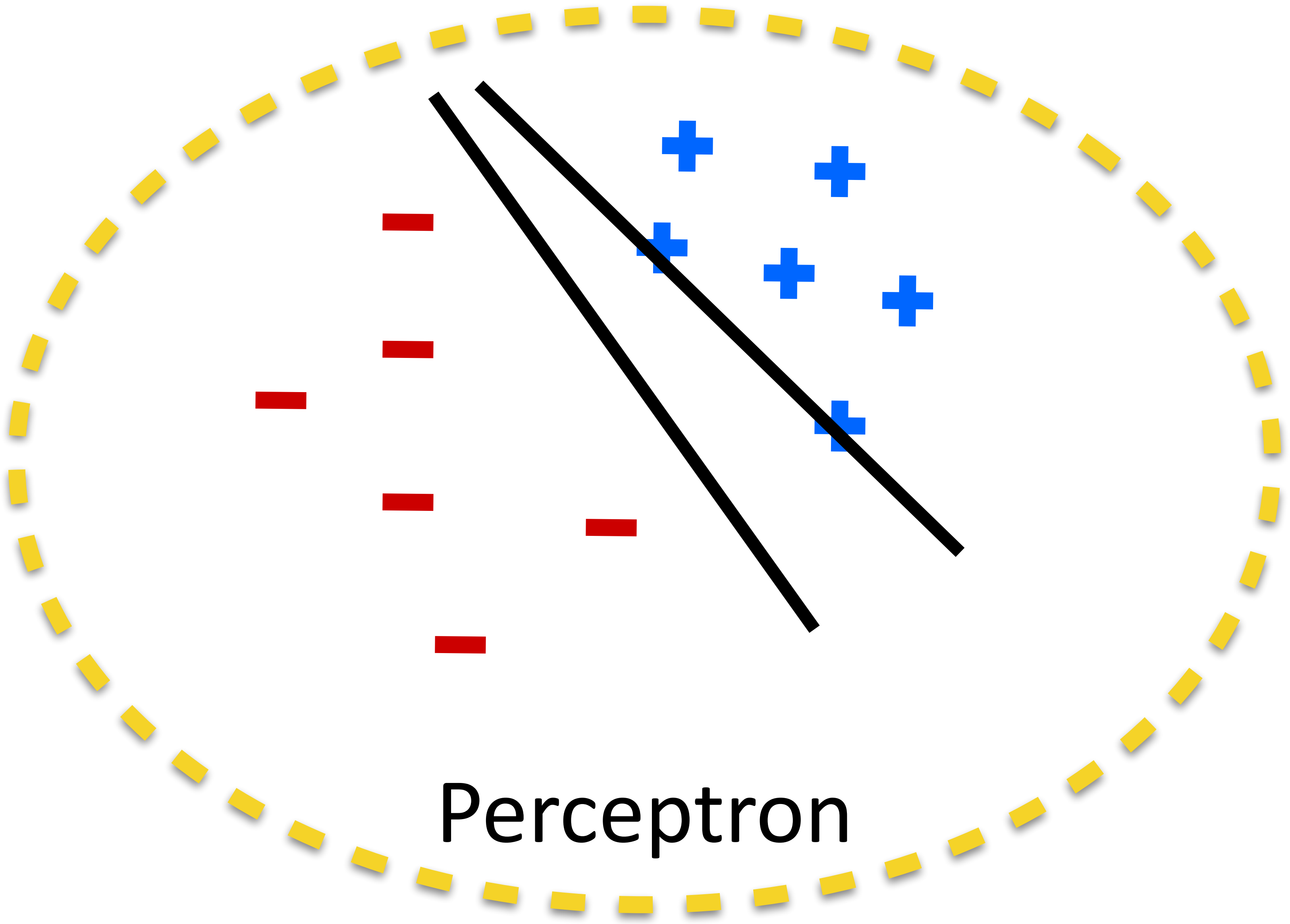
# Types of Supervised Classifiers




Naive Bayes




Logistic regression



Perceptron

Many others….

# Perceptron

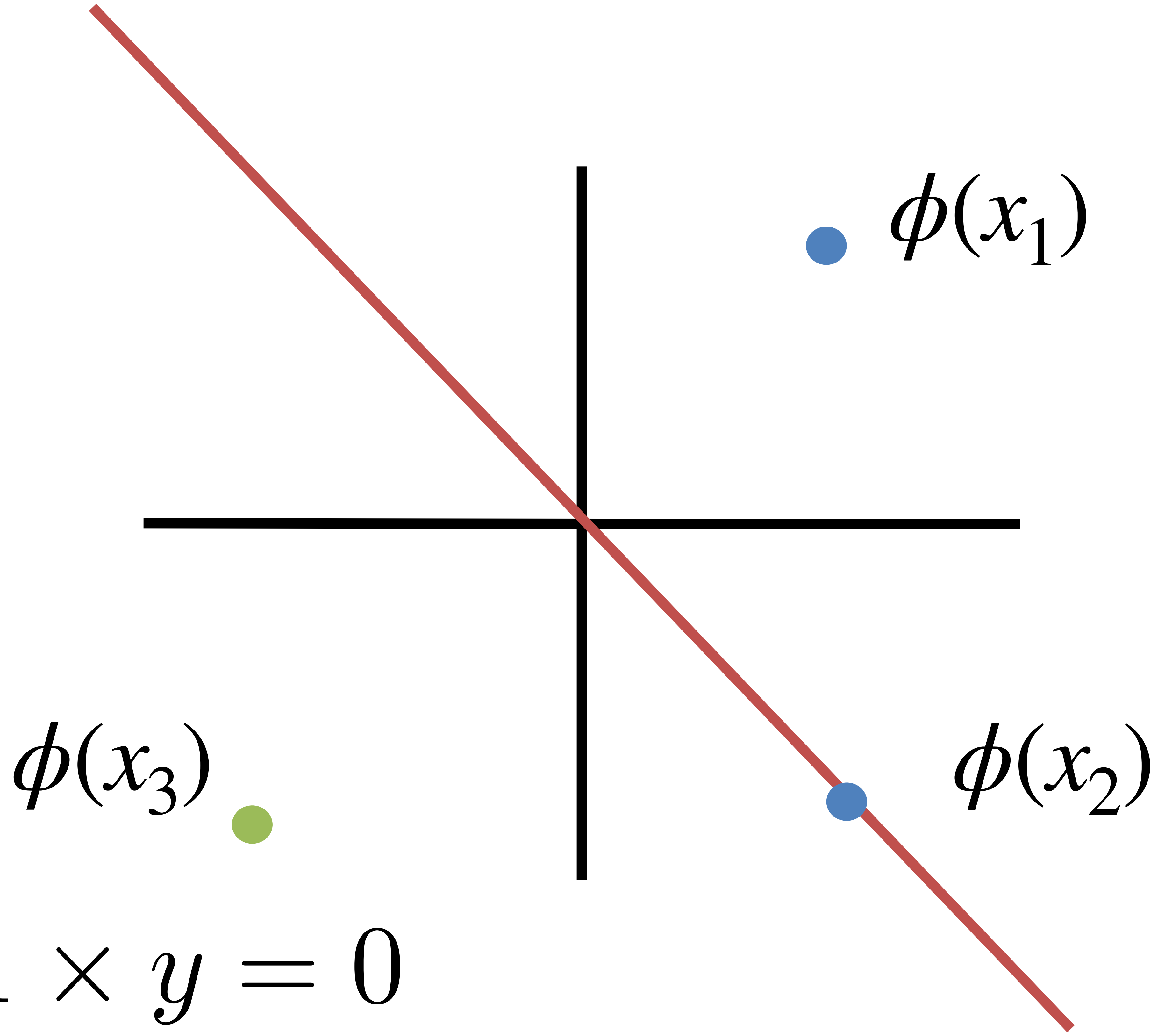
- Start with zero weight vector.
- Visit training examples one by one.
- Decision rule: $w \cdot \phi(x) > 0$
  - If correct: do nothing!
  - If incorrect: if label is positive, $w \leftarrow w + \phi(x)$

    negative, $w \leftarrow w - \phi(x)$

# Geometric Interpretation: Separating Hyperplane

$$\phi(x_1) = [1, 1], \quad y^{(1)} = 1$$
$$\phi(x_2) = [1, -1], \quad y^{(2)} = 1$$
$$\phi(x_3) = [-1, -1], \quad y^{(3)} = -1$$

$$w = [1, 1]$$




$$w \cdot [x, y] = 1 \times x + 1 \times y = 0$$

# Geometric Interpretation: Separating Hyperplane