Lecture 2: Binary Classification

Kai-Wei Chang
CS @ UCLA
kw@kwchang.net

Couse webpage: https://uclanlp.github.io/CS269-17/

Announcements

- Waiting list: If you're not enrolled, please sign up.

- We will use Piazza as an online discussion platform. Please sign up here:
  piazza.com/ucla/fall2017/cs269

This Lecture

- Supervised Learning
- Linear Classifiers
  - Perceptron Algorithm
  - Support Vector Machine
  - Logistic Regression
- Optimization in machine learning

The Badges game

+ Naoki Abe	- Eric Baum

- Conference attendees to the ICML 1994 were given name badges labeled with + or −.
- What function was used to assign these labels?

Training data

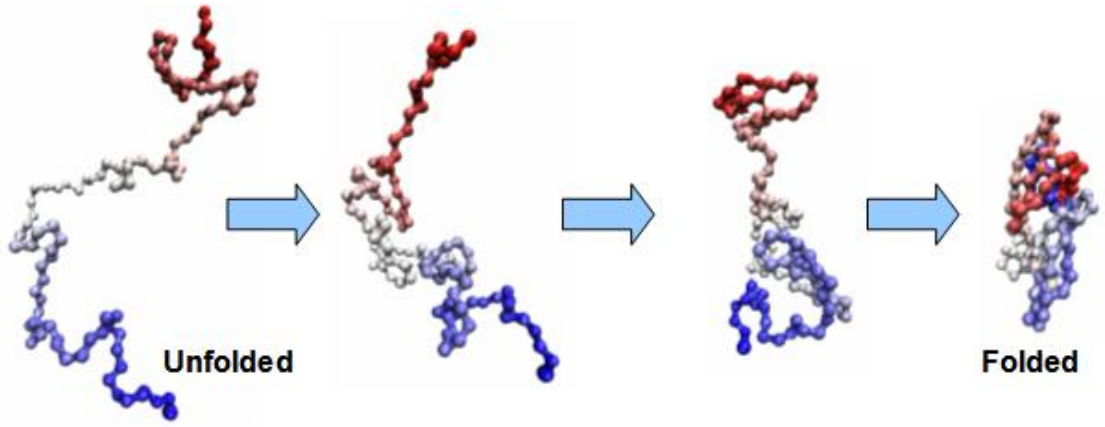
+ Naoki Abe
- Myriam Abramson
+ David W. Aha
+ Kamal M. Ali
- Eric Allender
+ Dana Angluin
- Chidanand Apte
+ Minoru Asada
+ Lars Asker
+ Javed Aslam
+ Jose L. Balcazar
- Cristina Baroglio
+ Peter Bartlett
- Eric Baum
+ Welton Becket
- Shai Ben-David
+ George Berg
+ Neil Berkman
+ Malini Bhandaru
+ Bir Bhanu
+ Reinhard Blasig
- Avrim Blum
- Anselm Blumer
+ Justin Boyan
+ Carla E. Brodley
+ Nader Bshouty
- Wray Buntine
- Andrey Burago
+ Tom Bylander
+ Bill Byrne
- Claire Cardie
+ John Case
+ Jason Catlett
- Philip Chan
- Zhixiang Chen
- Chris Darken

Raw test data

Gerald F. DeJong
Chris Drummond
Yolanda Gil
Attilio Giordana
Jiarong Hong
J. R. Quinlan
Priscilla Rasmussen
Dan Roth
Yoram Singer
Lyle H. Ungar

Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
  - E.g.: Identify DNA binding sites, predicting disease progression, predicting protein folding structure



Why we need machine learning?

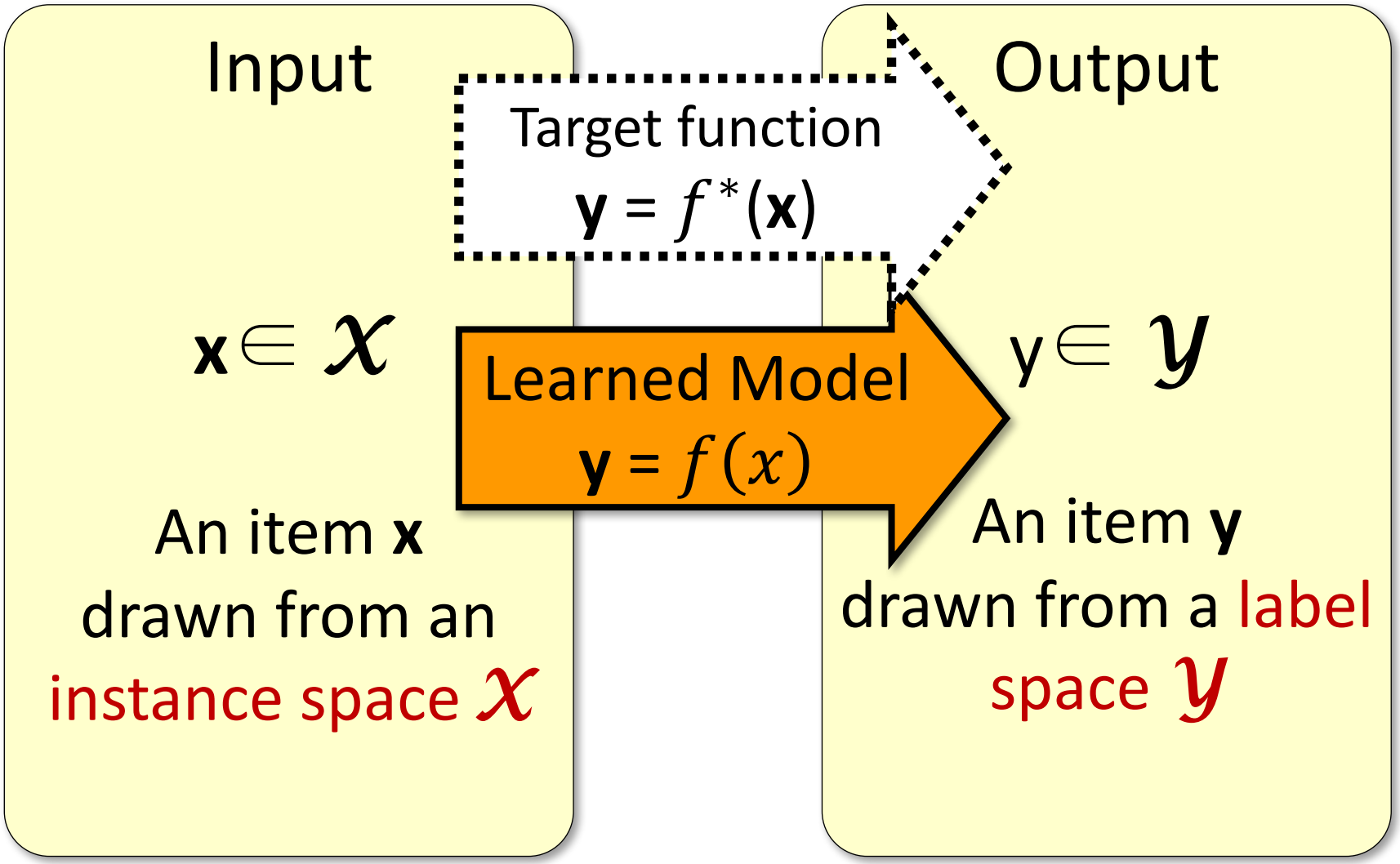
- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can't describe how they do it
  - E.g.: Object recognition

?

Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can't describe how they do it
- The desired function is hard to be written down in a closed form
  - E.g.,: predict stock price

Supervised learning



Supervised learning

Input

$\mathbf{x} \in \mathcal{X}$

An item **x** drawn from an instance space $\mathcal{X}$

x is represented in a feature space
- Typically $x \in \{0,1\}^n$ or R^N
- Usually represented as a vector
- We call it input vector

Supervised learning

y is represented in output space (label space)

Different kinds of output:

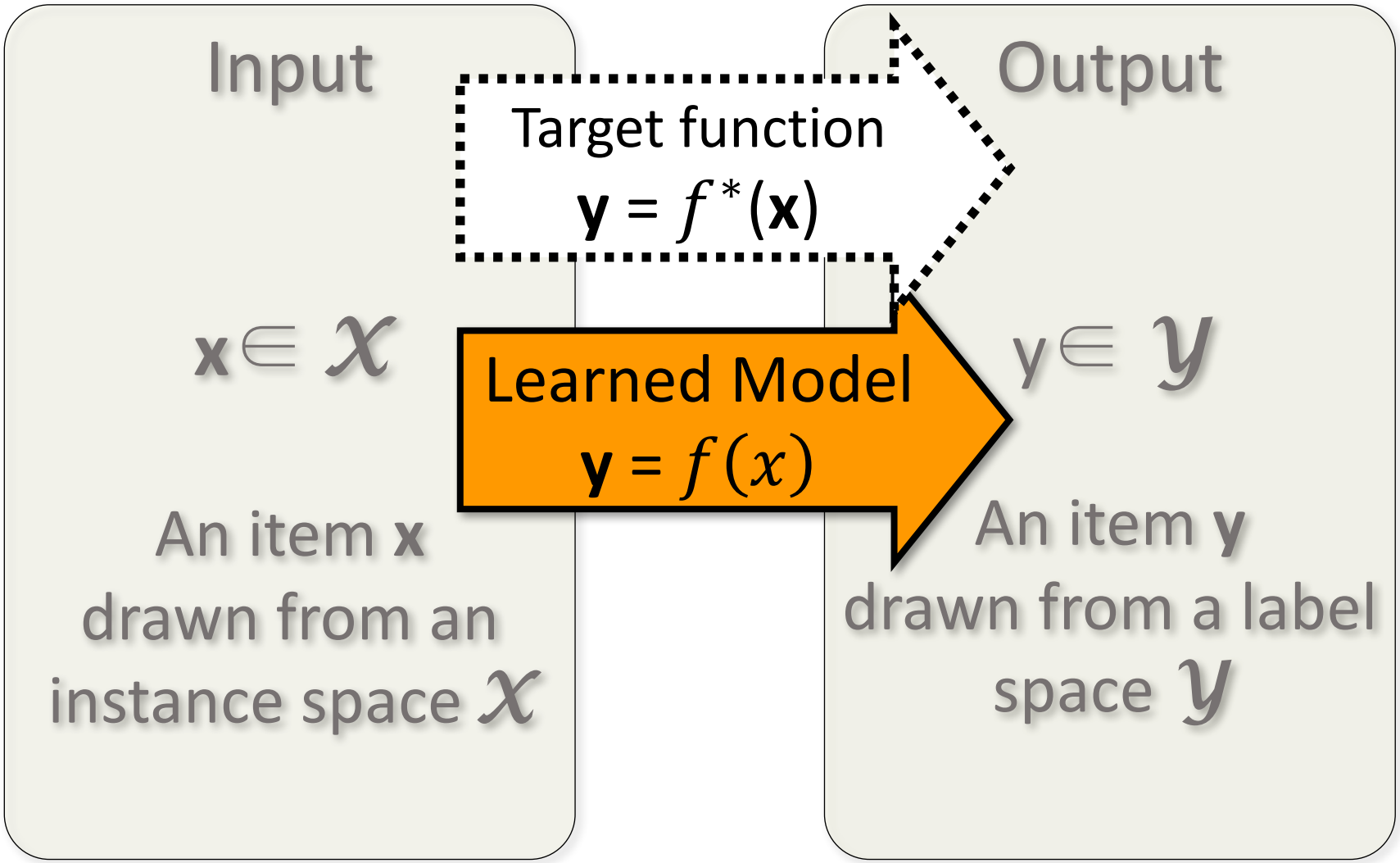
- Binary classification:
  $y \in \{-1,1\}$
- Multiclass classification:
  $y \in \{1,2,3, \ldots K\}$
- Regression:
  $y \in R$
- Structured output
  $y \in \{1,2,3, \ldots K\}^N$

Output

$y \in \boldsymbol{y}$

An item **y** drawn from a label space $\boldsymbol{y}$

Learning the mapping

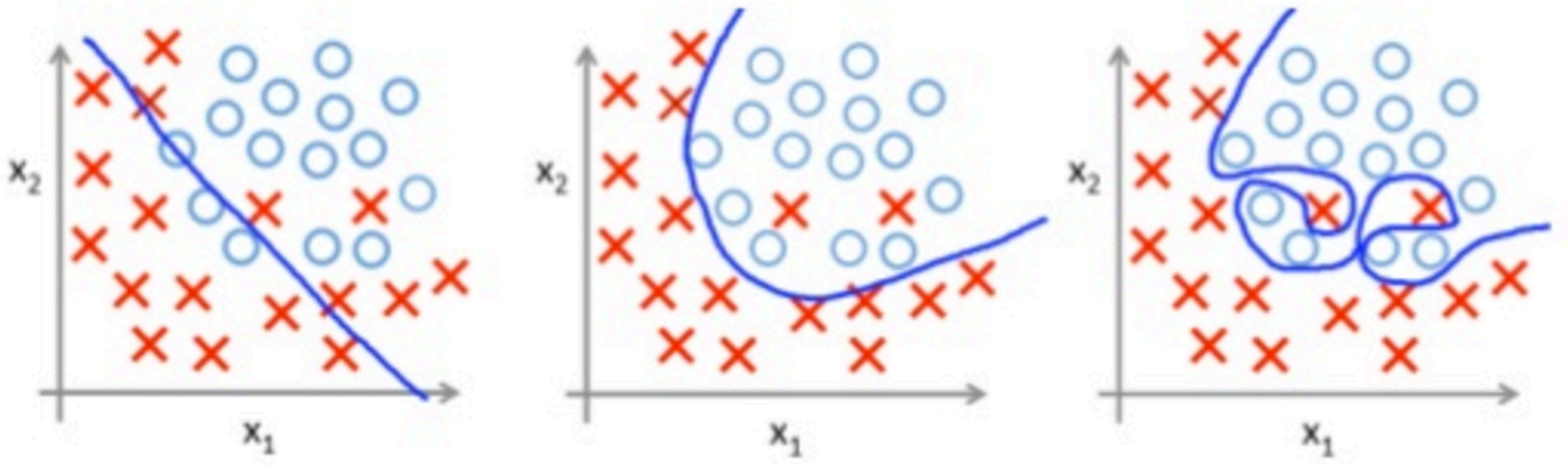


Goal

- Find a good approximation of $f^*(\cdot)$
- Good in what sense?

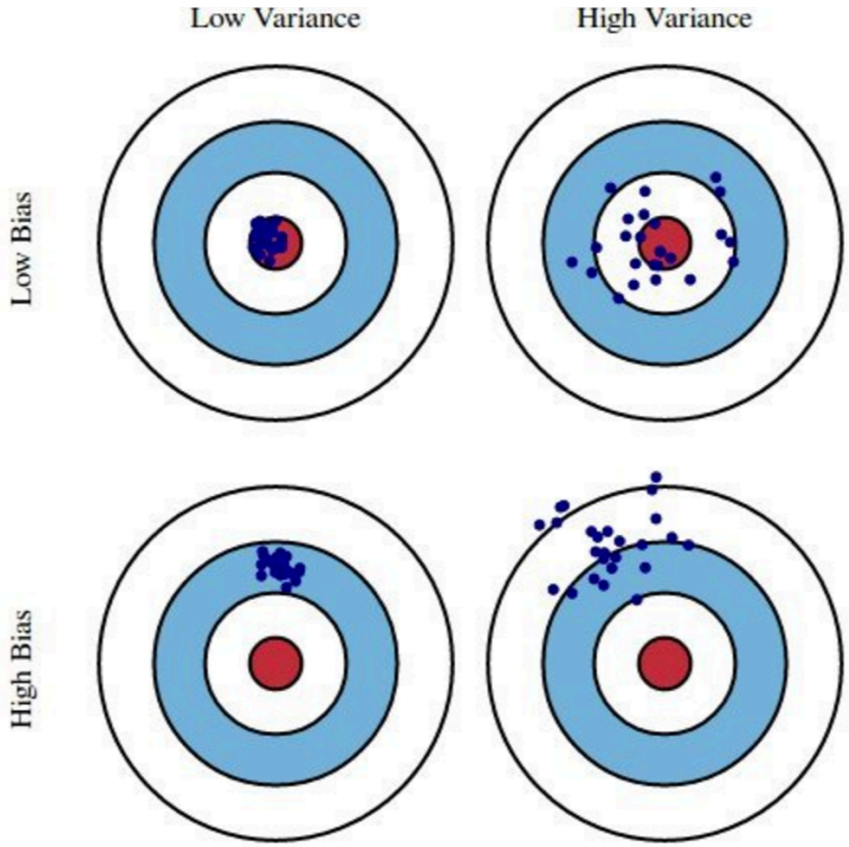
Under-fitting and over-fitting

- Which classifier (blue line) is the best one?



Bias V.S. Variance

- Remember, training data are subsamples drawn from the true distribution
- Exam strategy:
  - Study every chapter well
    - A+: Low var & bias
  - Study only a few chapters
    - A+? B? C? Low bias; High var
  - Study every chapter roughly
    - B+: Low var; high bias
  - Go to sleep
    - B ~D: High var, high bias



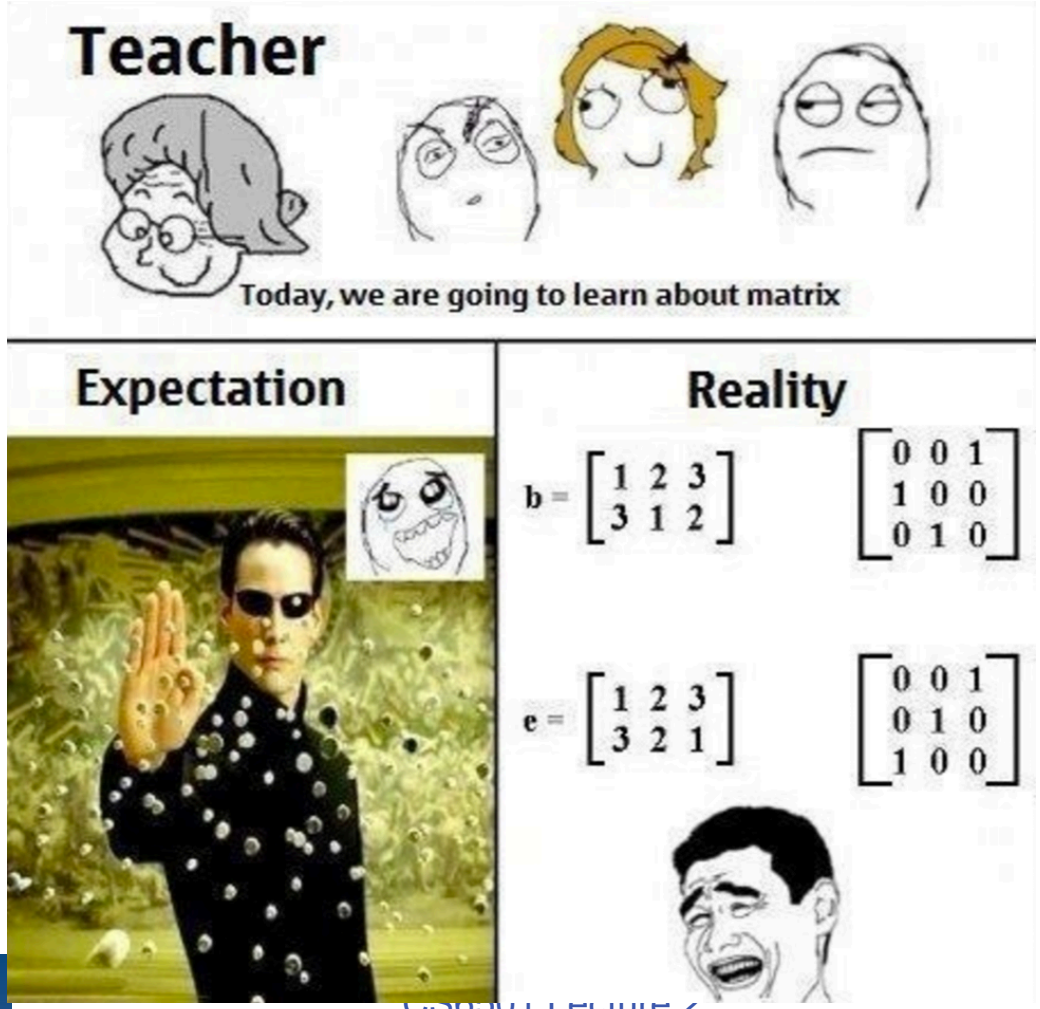
Questions of interest

- Representation
  - How to represent x, y (and latent factors)
- Modeling
  - What assumptions we made
  - i.e., what is the hypothesis set of f ?
- Algorithms
  - (learn) Give data, how to learn f?
  - (inference) Give test instance x and f, how to evaluate f(x)?
- Learning protocols
  - What is the goal of the learning algorithm?

Different learning protocols (more technical terms)

- Supervision signals?
  - Supervised learning, semi-supervised learning, unsupervised learning, bandit feedback
- What to be optimized?
  - Batch learning: minimize the risk (expected average loss)
  - Online learning:
    Receive one sample and make prediction
    Receive the label; then update
    minimize the accumulated loss

Linear classification

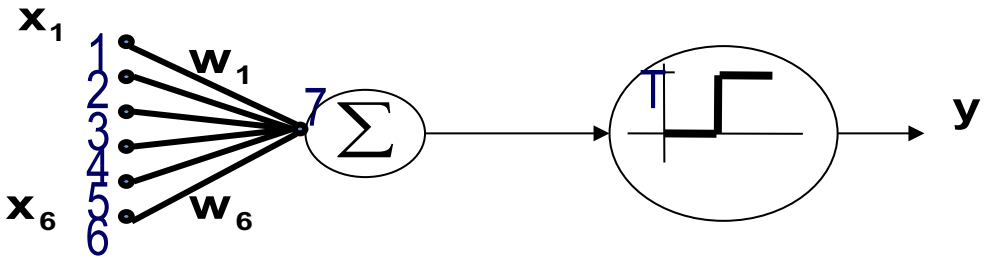


Linear classifiers

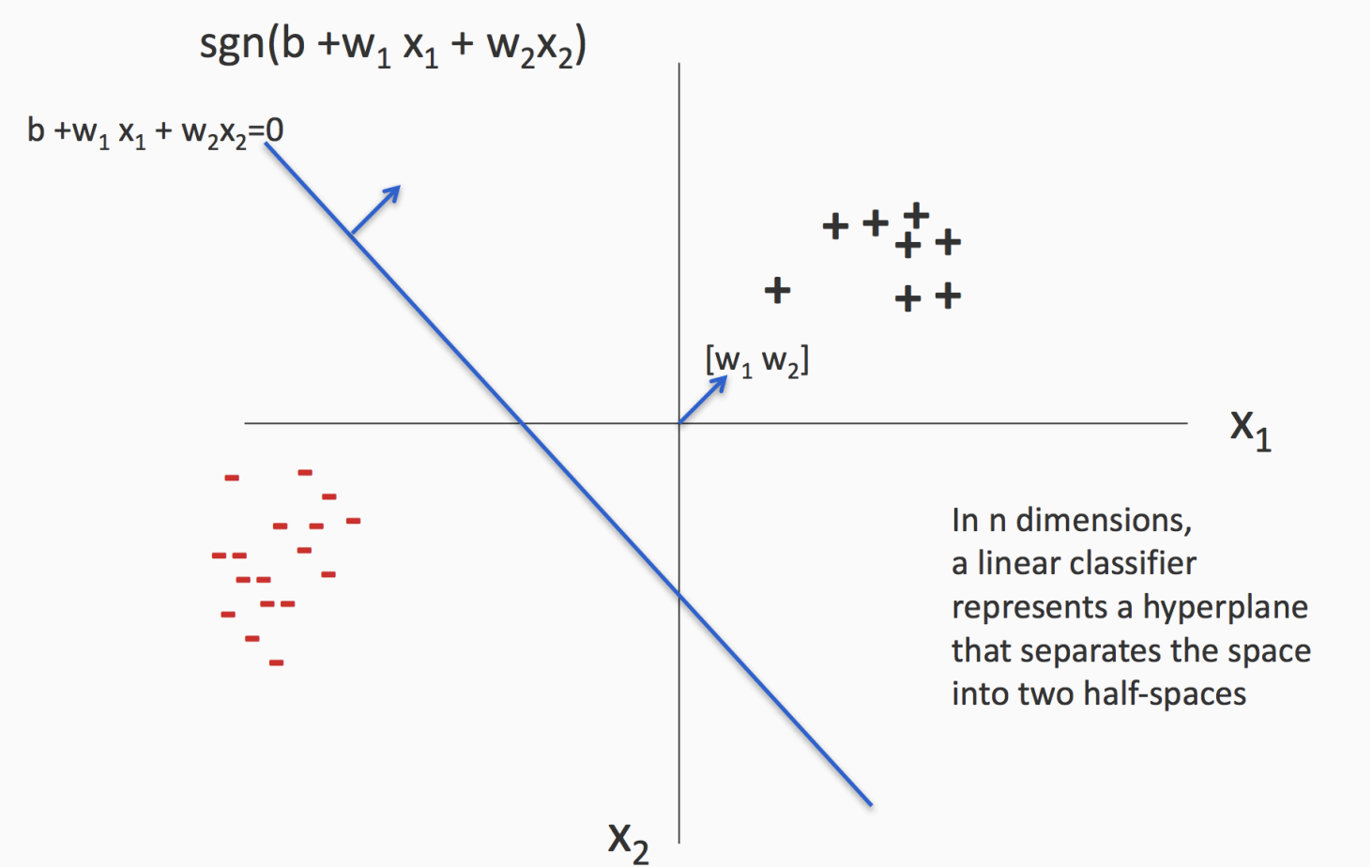
- For now, we consider binary classification
- Given a `training set` $\mathcal{D} = \{(\boldsymbol{x}, y)\}$, find a linear threshold units classify an example x using the classification rule:

$$\text{sgn}(b + \boldsymbol{w}^T \boldsymbol{x}) = \text{sgn}(b + \sum_i w_i x_i)$$

- $b + \boldsymbol{w}^T \boldsymbol{x} \geq 0 \Rightarrow$ Predict y = 1
- $b + \boldsymbol{w}^T \boldsymbol{x} < 0 \Rightarrow$ Predict y = -1



The geometry interpretation



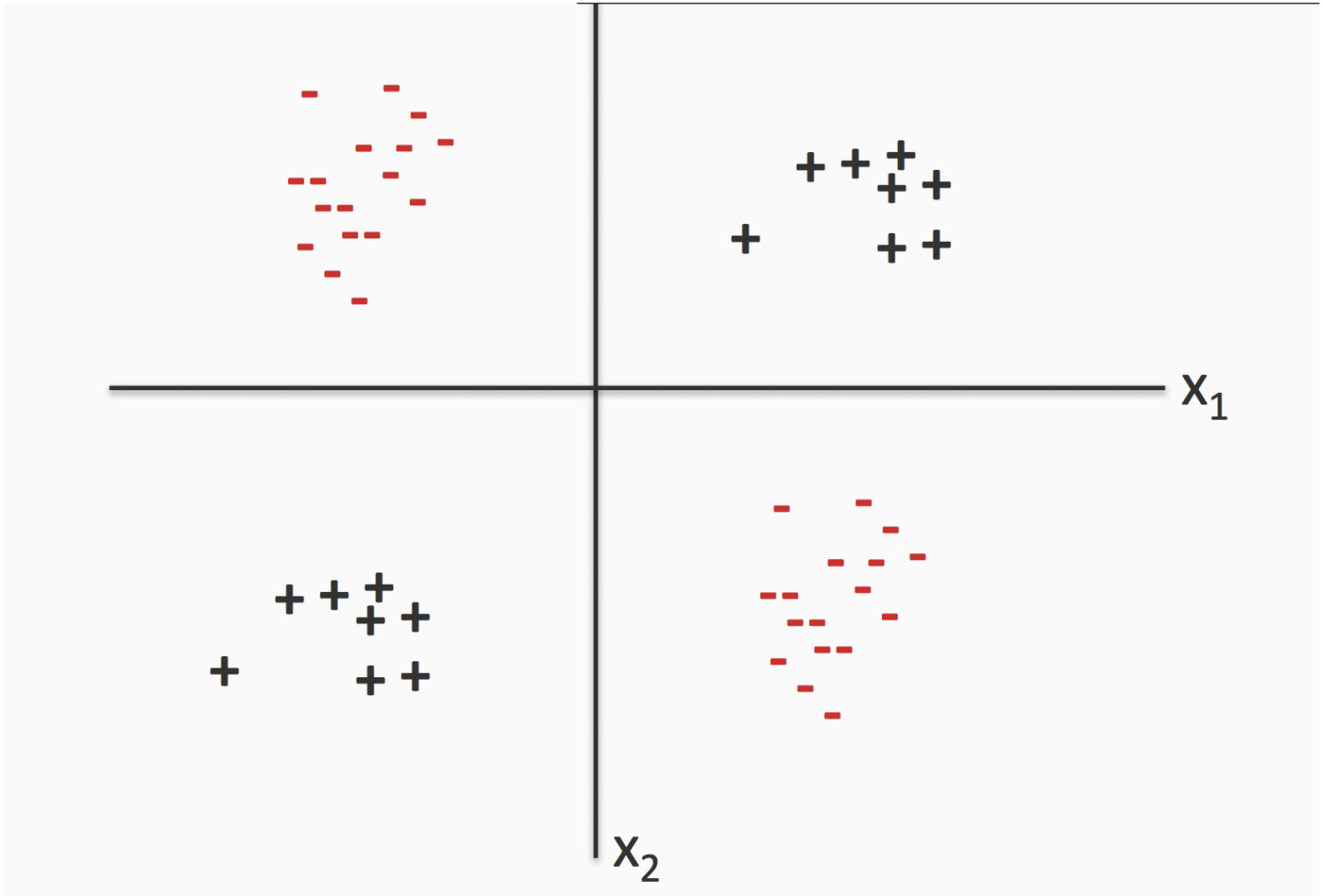
A simple trick to remove the bias term b

$$
\begin{aligned}
& w^T x + b \\
&= [w^T \; b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\
&= \tilde{w} \cdot \tilde{x}
\end{aligned}
$$

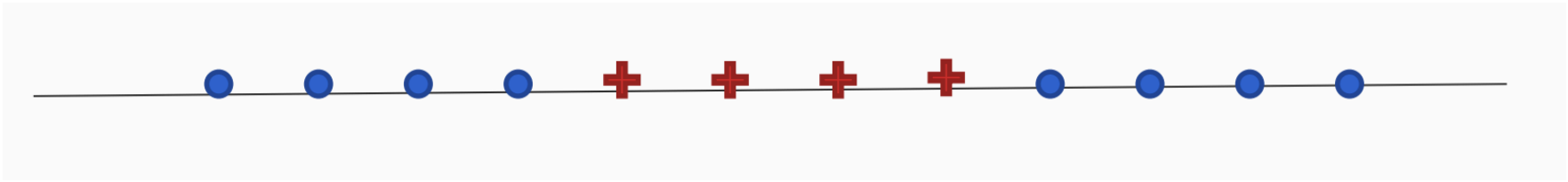
$$
\begin{aligned}
\tilde{w} &= [w^T \; b]^T \\
\tilde{x} &= [x^T \; 1]^T
\end{aligned}
$$

For simplicity, I'll write $\tilde{w}$ and $\tilde{x}$ as w and x when there is no confusion

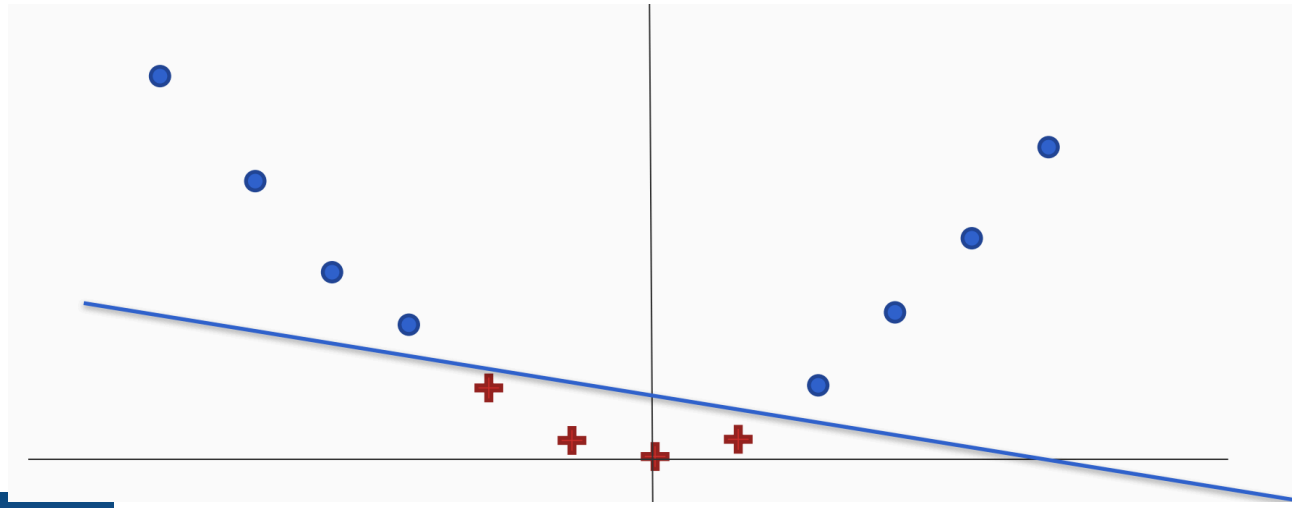
Some data are not linearly separable



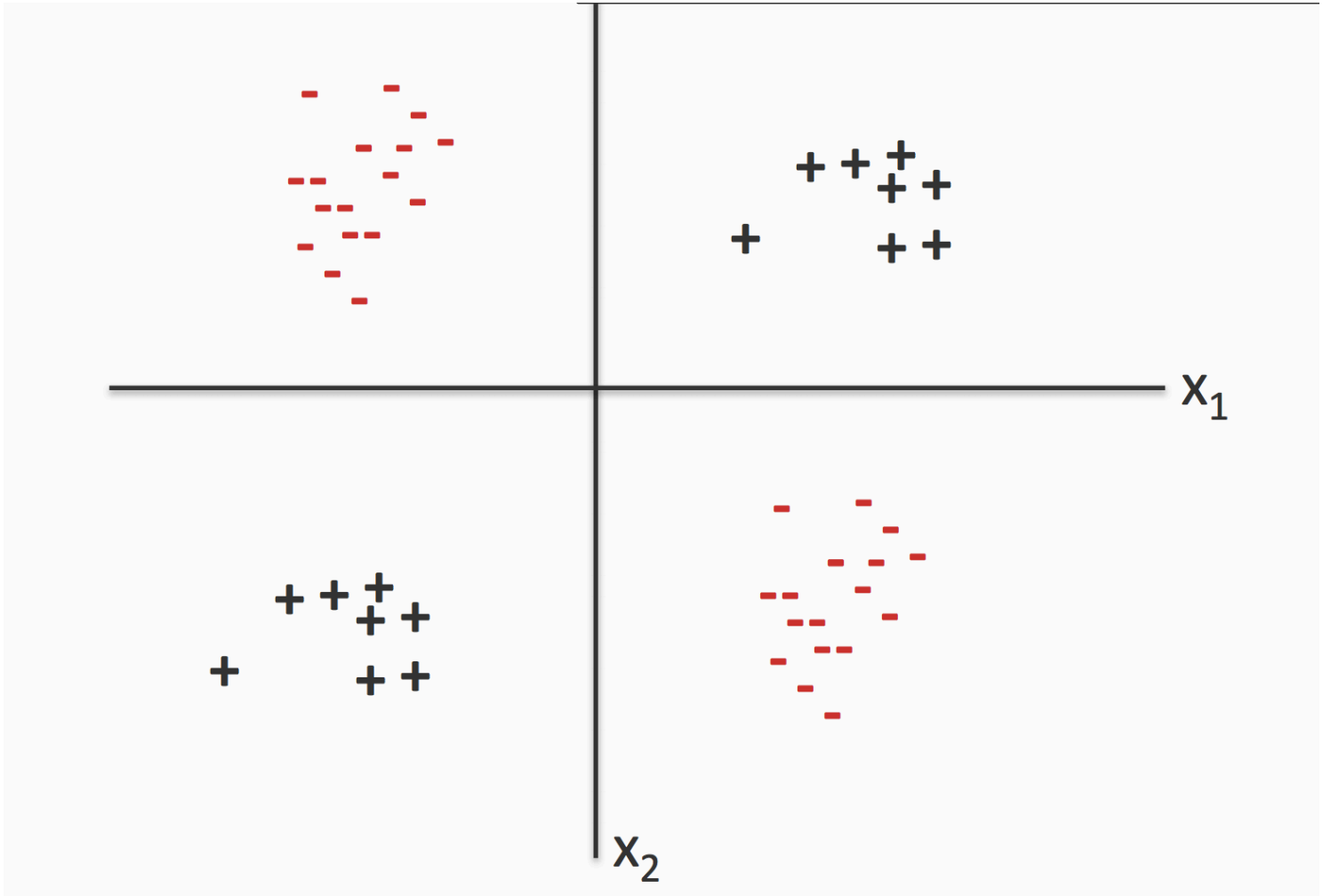
But they can be **made** liner



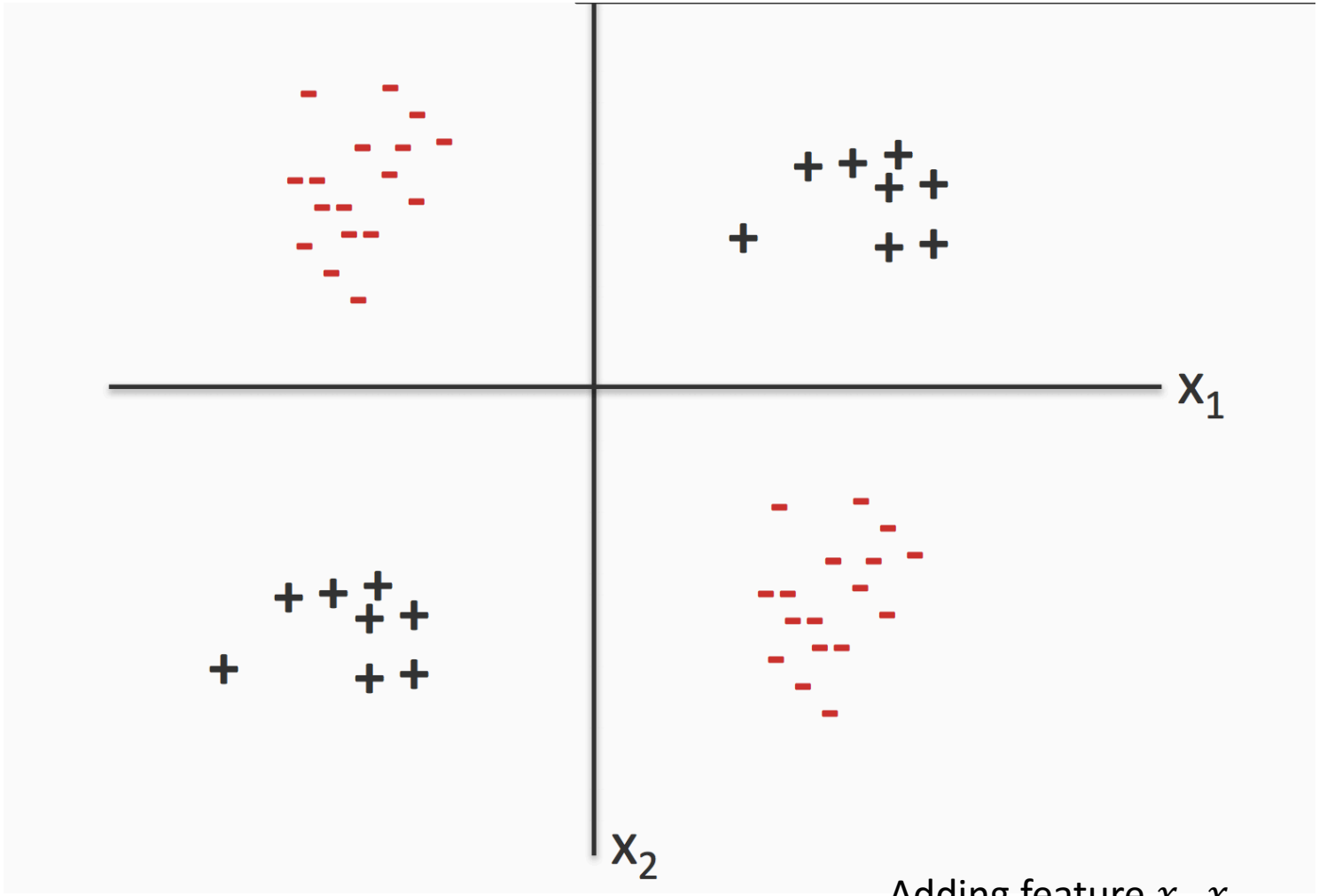
Using a different representation
e.g., feature conjunctions,
non-linear mapping



Exercise: can you make these data points linearly separable?



Exercise: can you make these data points linearly separable?



Adding feature x_1 x_2

Linear classifiers

- Let's take a look at a few linear classifiers
- We will show later, they can be written in the same framework!

- Perceptron
- (Linear) Support Vector Machines
- Logistic Regression

The Perceptron Algorithm [Rosenblatt 1958]

- Goal: find a separating hyperplane
- Can be used in an online setting: considers one example at a time
- Converges if data is separable -- mistake bound

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. $\hat{y} = \mathrm{sgn}(\boldsymbol{w}^\top \boldsymbol{x})$ (predict)
5. if $\hat{y} \neq y$, $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$ (update)
6. Return $\boldsymbol{w}$

Learning rate

Prediction: $y^{\text{test}} \leftarrow \mathrm{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

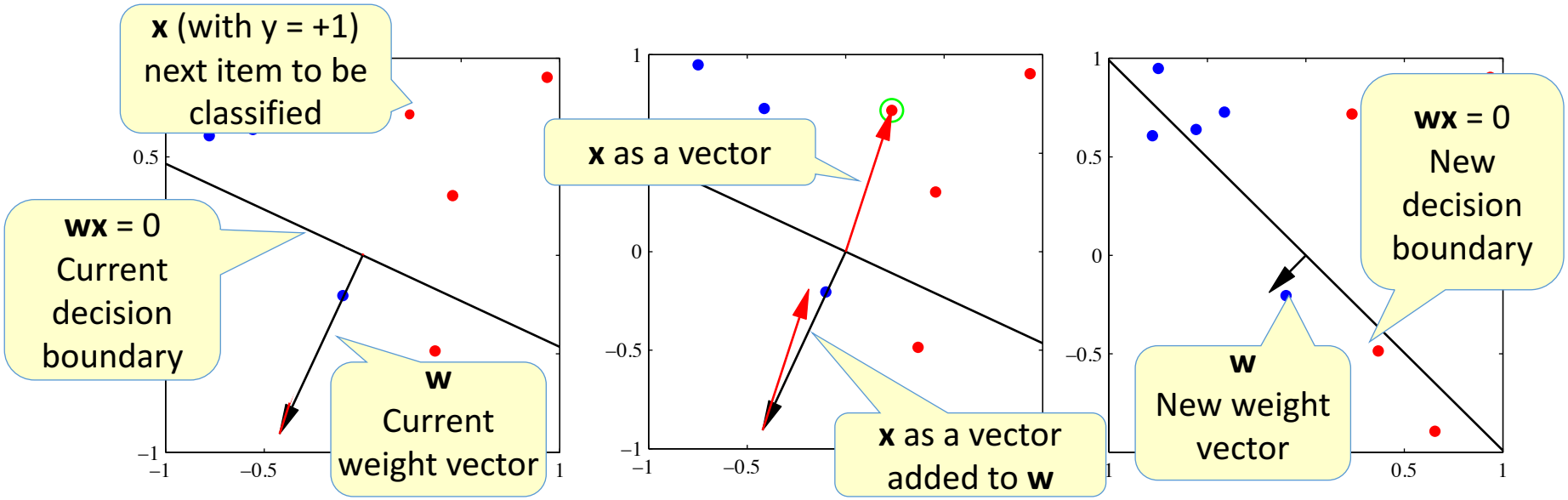
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

Geometry Interpretation

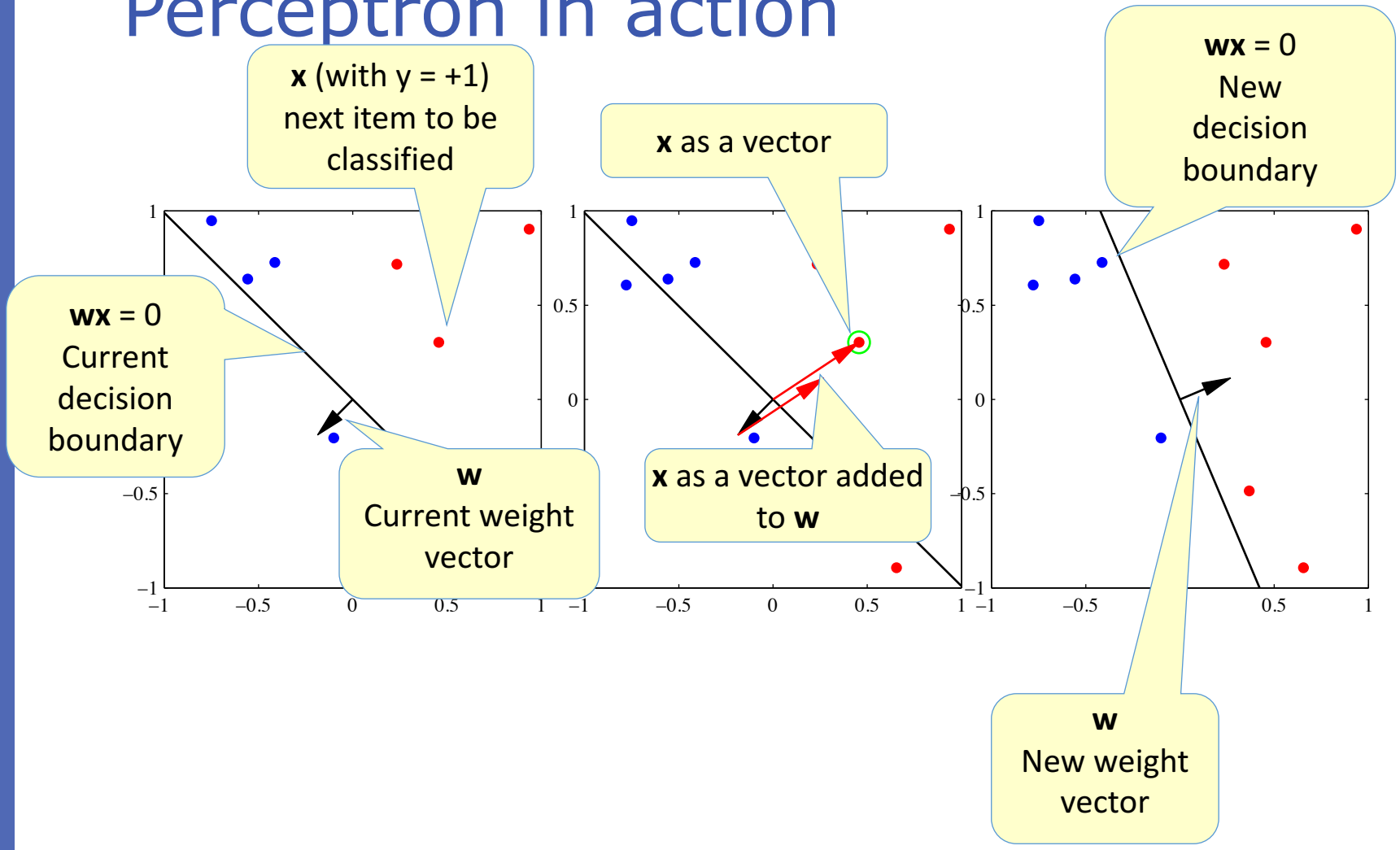


Weight vector points to the positive side

(Figures from Bishop 2006)

• Positive
• Negative

Perceptron in action



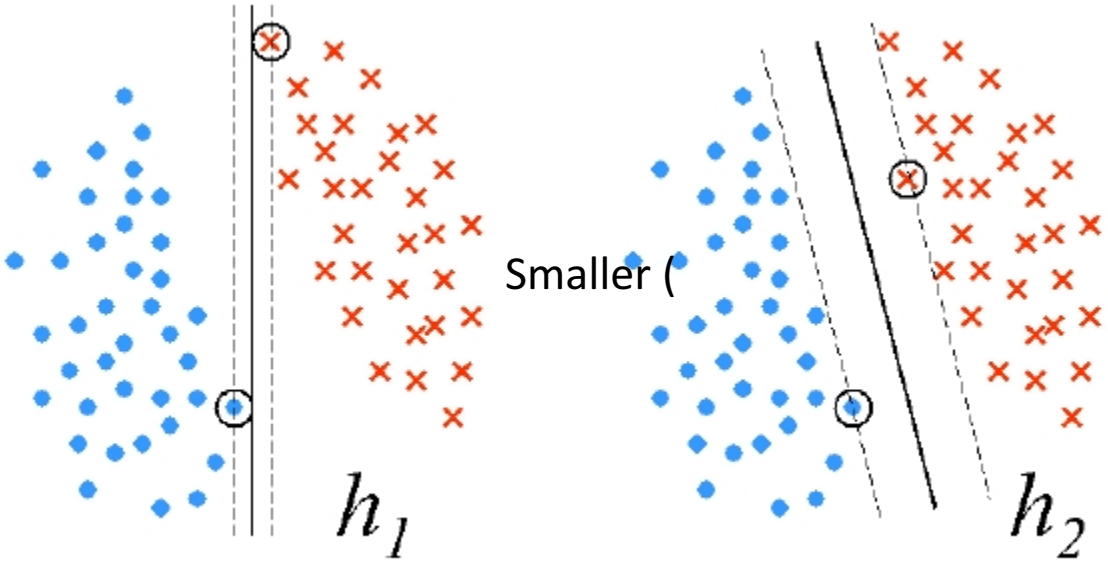
(Figures from Bishop 2006)

Convergence of Perceptron

- Mistake bound
  - If data is linearly separable (i.e., a good linear model exists), the perceptron will converge after a fixed number of mistakes [Novikoff 1962]

Marginal Perceptron -- Motivation

- Which separating hyper-plane is better?



The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

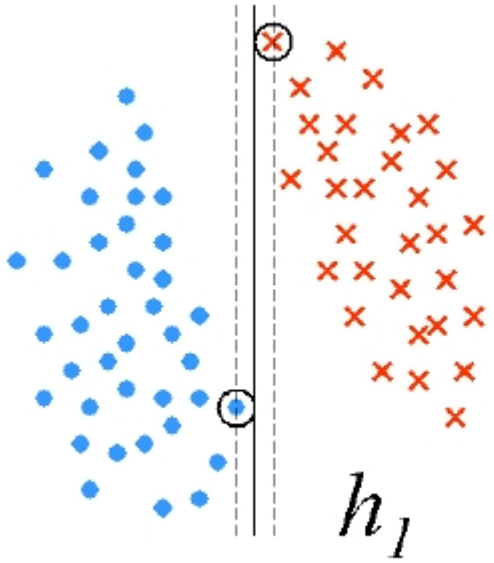
1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \delta$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$



Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

How about batch setting?

- Learning as loss minimization

1. Collect training data $\widehat{D} = \{(\boldsymbol{x}, y)\}$
2. Pick a hypothesis class
   - E.g., linear classifiers, deep neural networks
3. Choose a loss function
   - Hinge loss, negative log-likelihood
   - We can impose a preference (i.e., prior) over hypotheses, e.g., simpler is better
4. Minimize the expected loss
   - SGD, coordinate descent, Newton methods, LBFGS

Batch learning setup

- $\widehat{D} = \{(\boldsymbol{x}, y)\}$ drawn from a fixed, unknown distribution $\mathcal{D}$
- A hidden oracle classifier f^*, $y = f^*(x)$
- We wish to find a hypothesis $f \in H$ that mimics f^*
- We define a loss function $L(f(x), f^*(x))$ that penalizes mistakes
- What is the ideal f?

$$\arg\min_{f \in H} E_{x \sim D} \left[L\left(f(x), f^*(x)\right) \right]$$

expected loss

Batch learning setup

- $\widehat{D} = \{(\boldsymbol{x}, y)\}$ drawn from a fixed, unknown distribution $\mathcal{D}$
- A hidden oracle classifier f^*, $y = f^*(x)$
- We wish to find a hypothesis $f \in H$ that mimics f^*
- We define a loss function $L(f(x), f^*(x))$ that penalizes mistakes
- What is the ideal f?

Let's define

$$L_{o-1}(y, y') = 1 \ \ if \ y \neq y'$$
$$0 \ \ if \ y = y'$$

$$\min_{f \in H} E_{x \sim D} \left[L_{0-1}\left(f(x), f^*(x)\right) \right] = \min_{f \in H} E_{x \sim D}[\ \#mistakes]$$

How can we learn f from $\widehat{D}$

- We don't know D, we only see samples in $\widehat{D}$
- Instead, we minimize empirical loss

$$\min_{f \in H} \frac{1}{|\widehat{D}|} \sum_{(x,y) \in \widehat{D}} [L(f(x), y)]$$

How can we prevent over-fitting?

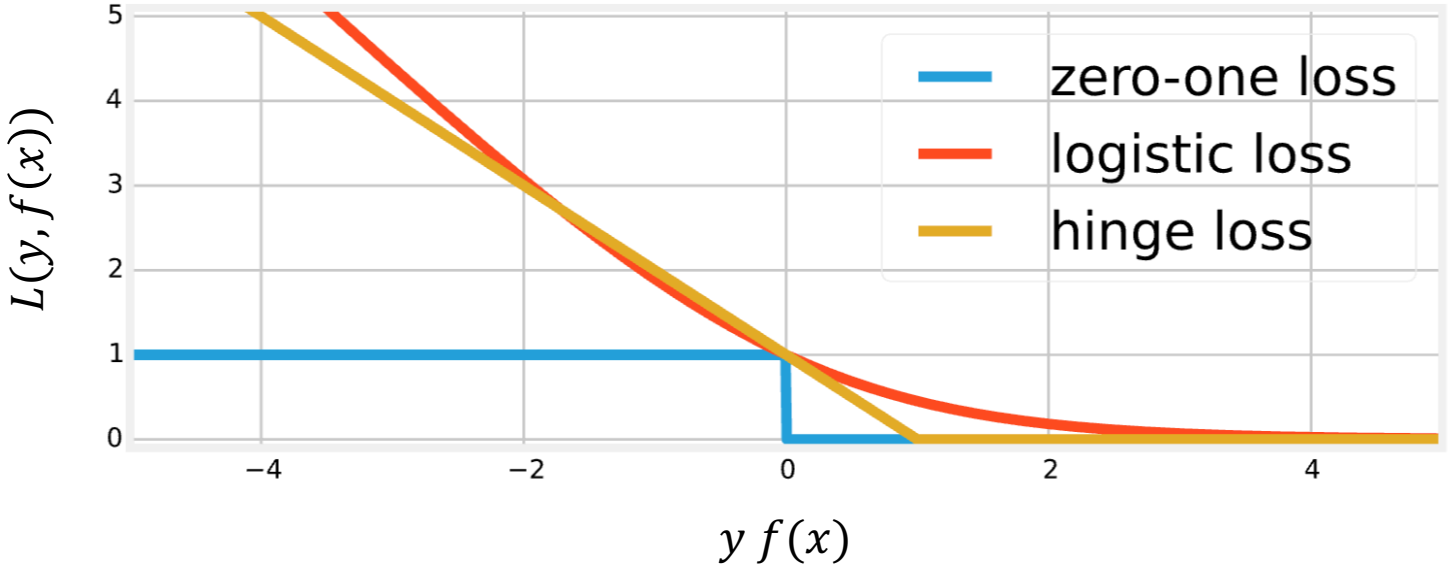
- With sufficient data, $\widehat{D} \approx$ D
- However, if data is insufficient ⇒ overfitting
- We can impose a preference over models

$$\min_{f \in H} R(f) + \frac{1}{\left|\widehat{D}\right|} \sum_{(x,y) \in \widehat{D}} [L(f(x), y)]$$

- We will discuss the choices of $R(f)$ later

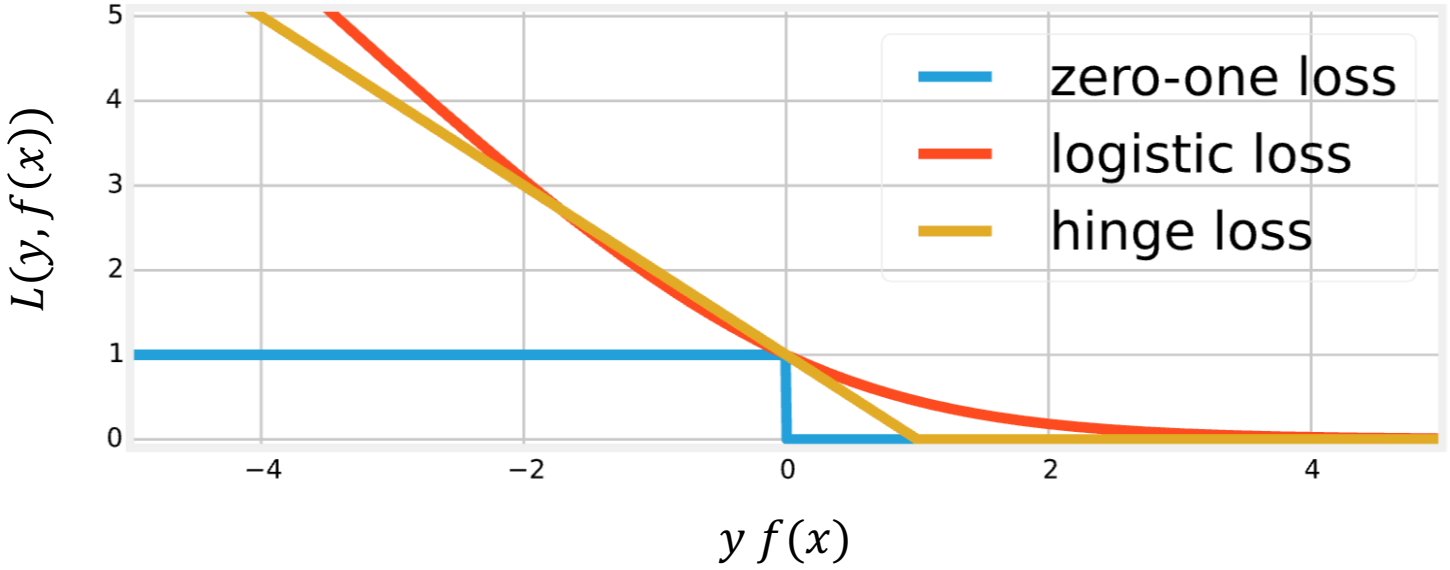
How about the loss function?

- Usually, we cannot minimize 0-1 loss
  - It is a combinatorial optimization problem: NP-hard
- Idea: minimizing its upper-bound



How about the loss fu

- Usually, we cannot minim
  - It is a combinatorial optimiz
- Idea: minimizing its upper-bound



Many choices

- We are minimizing with R, L, H with your choice

$$\min_{f \in H} R(f) + \frac{1}{|\widehat{D}|} \Sigma_{(x,y) \in \widehat{D}} [L(f(x), y)]$$

- Let consider H is a set of d-dimensional linear function
- H can be parameterized as $\{f(x): w^T x \geq 0\}, w \in R^d$

Back to Linear model

- We are minimizing with R, L, H with your choice

$$\min_{f \in H} R(f) + \frac{1}{|\widehat{D}|} \Sigma_{(x,y) \in \widehat{D}} [L(f(x), y)]$$

- Let decide H to be a set of d-dimensional linear function
- H can be parameterized as $\{f(x): w^T x \geq 0\}, w \in R^d$

- We are going to fine the best one based on $\widehat{D}$
  - i.e., find the best setting of w and b

Rewrite our optimization problem

- Minimizing the empirical loss:

$$\min_{f \in H} R(f) + \frac{1}{|\widehat{D}|} \sum_{(x,y) \in \widehat{D}} [L(f(x), y)]$$

- Minimizing the empirical loss with linear function

$$\min_{w \in R^d} R(w) + \frac{1}{|\widehat{D}|} \sum_{(x,y) \in \widehat{D}} [L(x, w, y)]$$

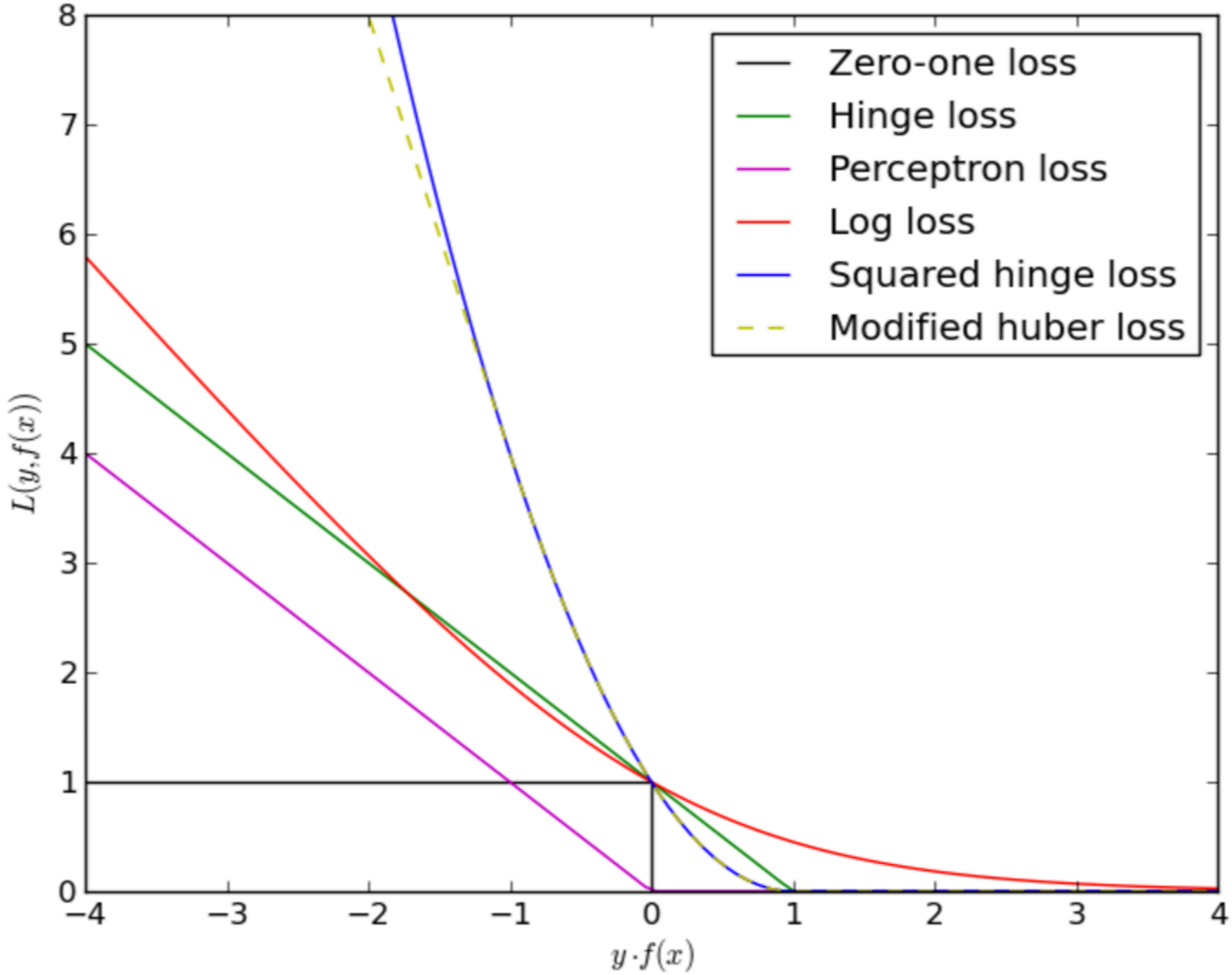
- What choices of R and L we have?

Many choices of loss function (L)

Many loss functions exist

- Perceptron loss $L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$
- Hinge loss (SVM) $L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$
- Exponential loss (AdaBoost) $L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$
- Logistic loss (logistic regression)

$$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$


Zero-one loss
Hinge loss
Perceptron loss
Log loss
Squared hinge loss
Modified huber loss
$L(y, f(x))$
$y \cdot f(x)$
8
7
6
5
4
3
2
1
0
−4
−3
−2
−1
0
1
2
3
4

Many choices of R(w)

- Minimizing the empirical loss with linear function

$$\min_{w \in R^d} \; R(w) \; + \; \frac{1}{|\widehat{D}|} \sum_{(x,y) \in \widehat{D}} [L(x, w, y)]$$

- Prefer simpler model: (how?)
  - Sparse:
    $R(w) =$ #non-zero elements in w (L0 regularizer)
    $R(w) = \sum_i |w_i|$ (L1 regularizer)
  - Gaussian prior (large margin w/ hinge loss):
    $R(w) = \sum_i w_i^2 = w^T w$ (L2 regularizer)

Support Vector Machines

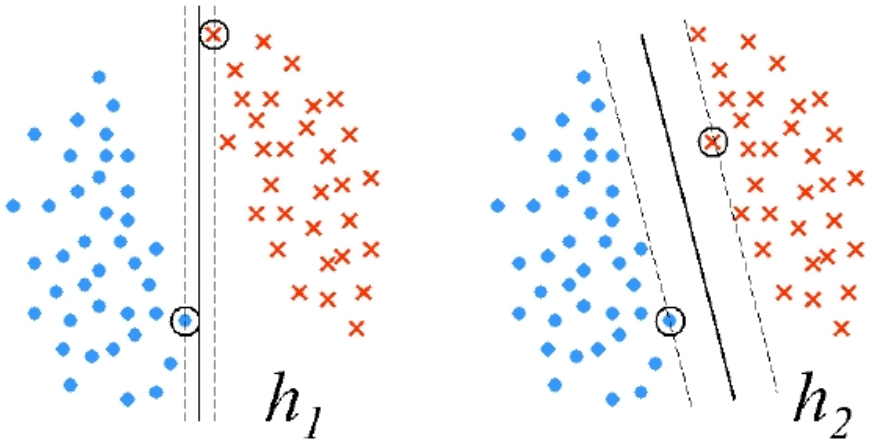
CMU ML protest

Support Vector Machines (SVMs)

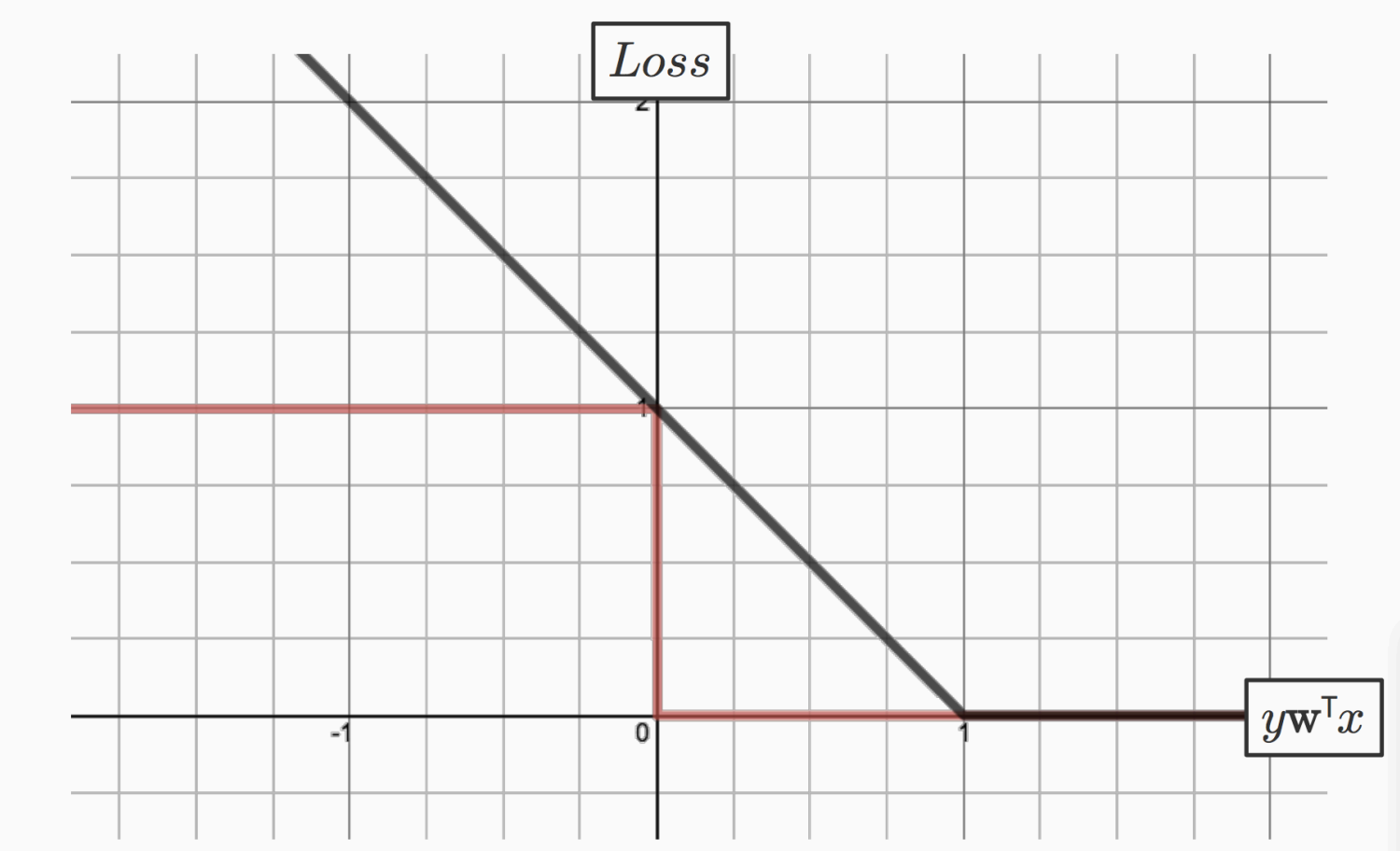
- R(w): l2-loss, $L(w, x, y)$: hinge loss

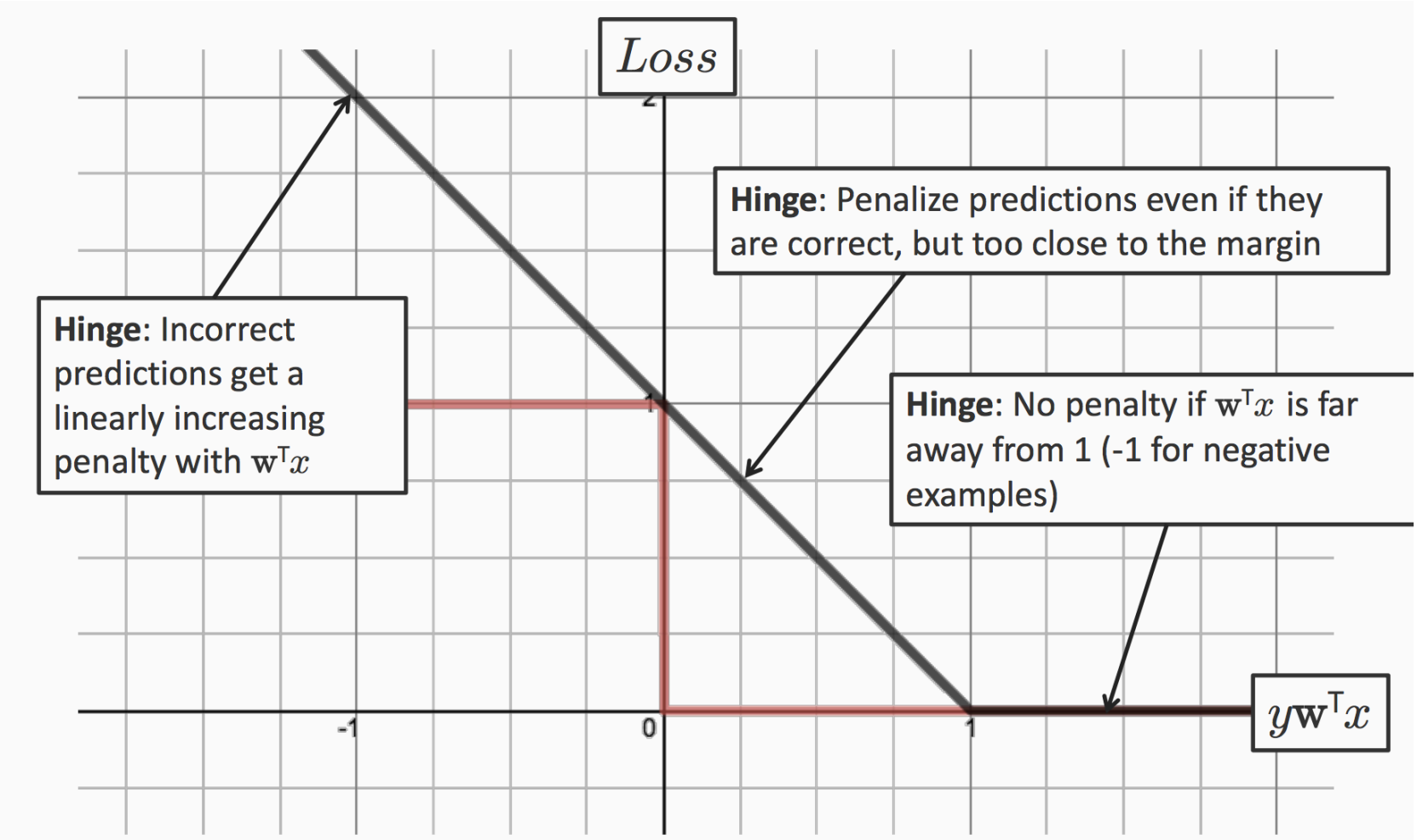
$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \ \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}))$$

- Maximizing margin (why?!!)



The hinge loss



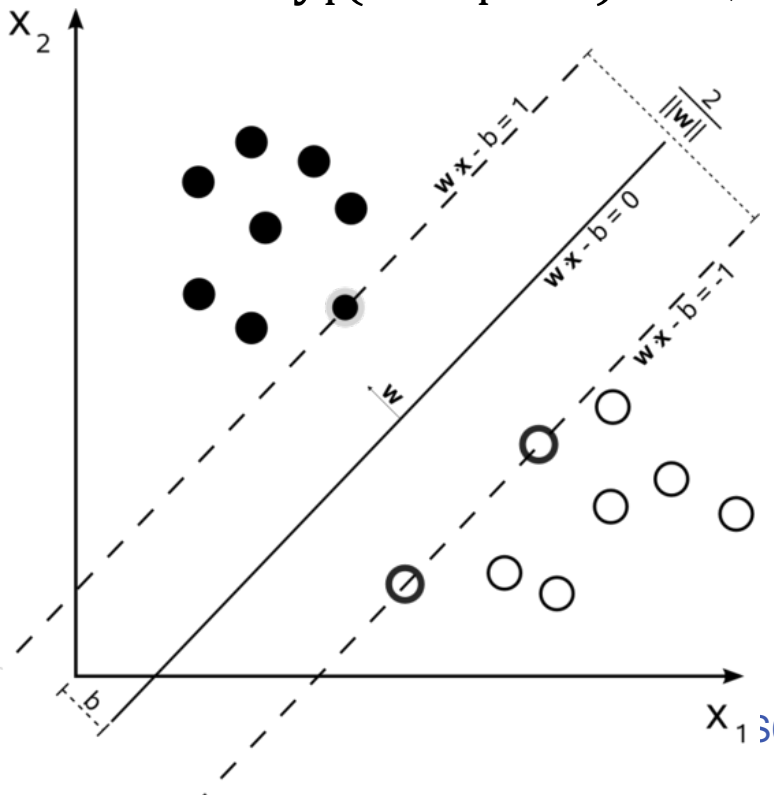

Loss
Hinge: Penalize predictions even if they are correct, but too close to the margin
Hinge: Incorrect predictions get a linearly increasing penalty with $\mathbf{w}^T x$
Hinge: No penalty if $\mathbf{w}^T x$ is far away from 1 (-1 for negative examples)
$y\mathbf{w}^T x$
2
1
-1
0
1

Let's view it from another direction

- SVM learns a model $\boldsymbol{w}$ on $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}$ by solving:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\text{s.t} \quad \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) \geq 1, \forall(\boldsymbol{x}_i, y_i) \in \mathcal{D}$$

(Hard SVM)



Why the margin is $\frac{2}{||w||}$?

$$\arg\max \frac{2}{||w||}$$
$$= \arg\min ||w||$$
$$= \arg\min ||w||^2$$
$$= \arg\min w^T w$$

Soft SVMs

- Data is not separable ⇒ hard SVM fails
  Why?
- Introduce a set of slack variable $\{\xi_i\}$
  ⇒ relax the constraints
- Given $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}$, soft SVM solves:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i \qquad \text{(Soft SVM)}$$

$$\text{s.t} \quad \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) \geq 1 - \xi_i;\ \xi_i \geq 0 \qquad \forall i$$

penalty parameter

An alternative formulation

$$\min_{w,b,\xi} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

$$\text{s.t} \quad \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) \geq 1 - \xi_i;\ \xi_i \geq 0 \qquad \forall i$$

- Rewrite the constraints:

$$\xi_i \geq 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b);\ \xi_i \geq 0 \qquad \forall i$$

- In the optimum, $\xi_i = \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b))$
- Soft SVM can be rewritten as:

$$\min_{w,b} \quad \frac{1}{2}\boxed{\boldsymbol{w}^T\boldsymbol{w}} + C\boxed{\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b))}$$

Regularization term

Empirical loss

An alternative formulation

$$\min \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

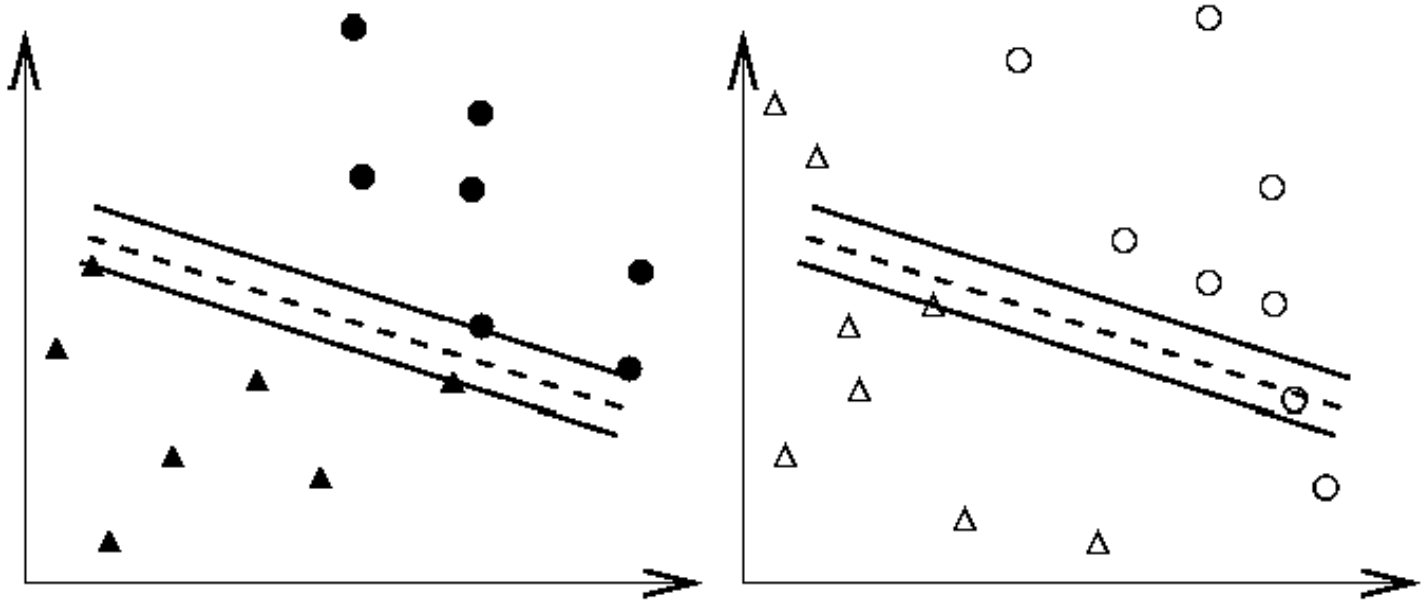
- We can simply "b" using the trick
- However, we will add b into the regularization term
- It is often okay, if we have many features

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b))$$

Regularization term

Empirical loss

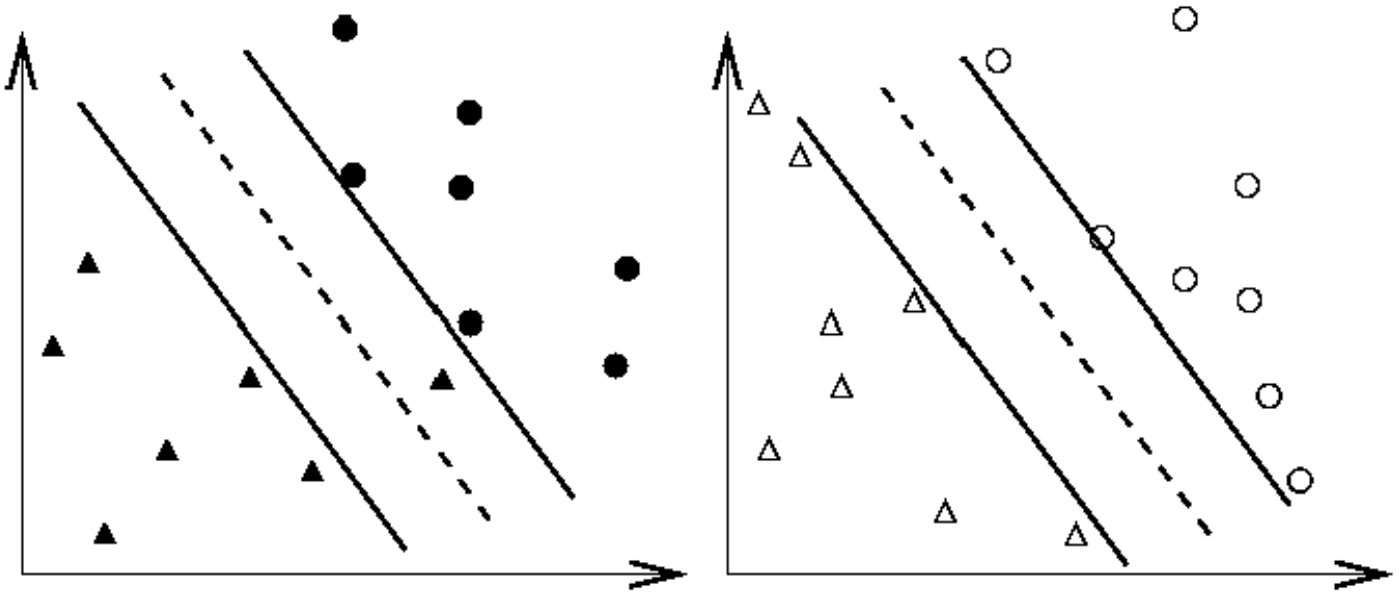
Balance between regularization and empirical loss



(a) Training data and an over-fitting classifier

(b) Testing data and an over-fitting classifier

Balance between regularization and empirical loss



(c) Training data and a better classifier

(d) Testing data and a better classifier

(DEMO)

Regularized loss minimization

- L1-Loss SVM

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$

- L2-Loss SVM

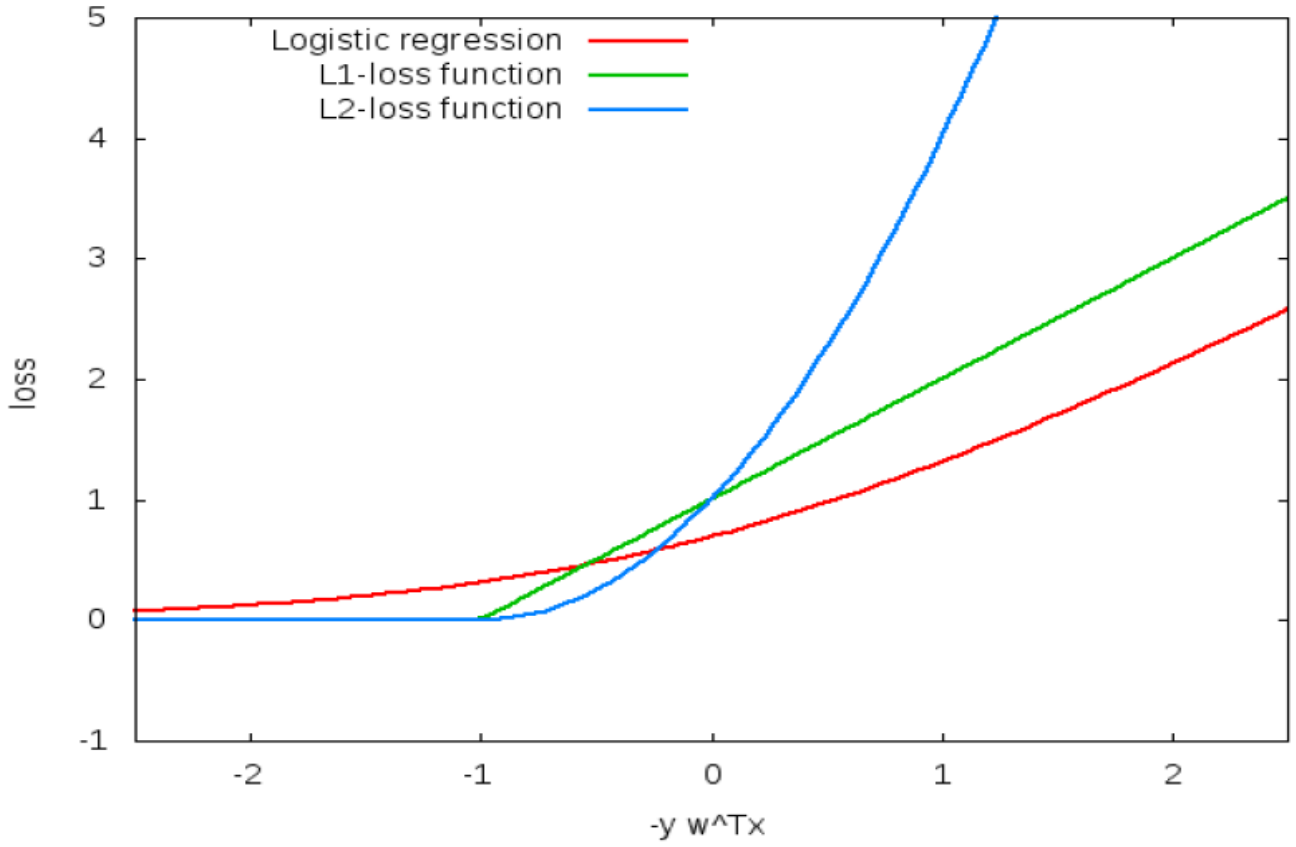
$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))^2$$

- Logistic Regression (regularized)

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \log(\,1 + e^{-\mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i})})$$

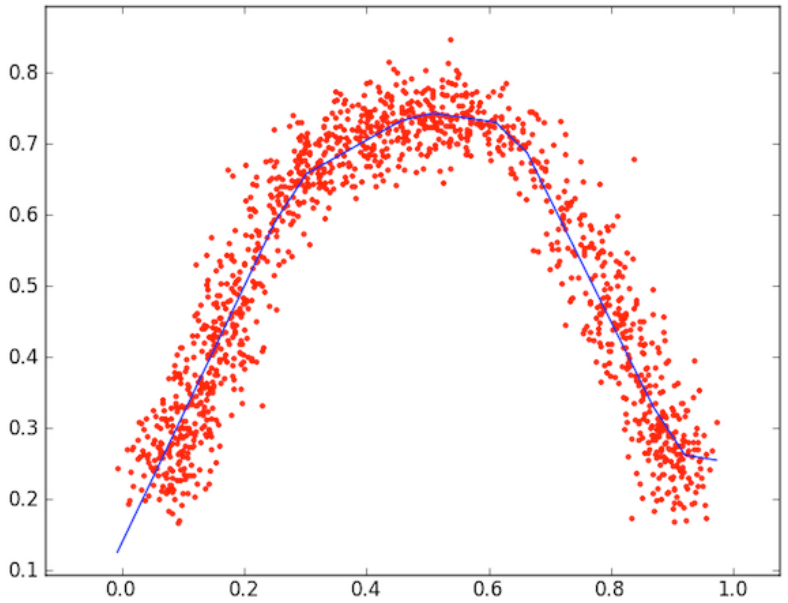
- Loss over training data + regularizer

Loss Functions

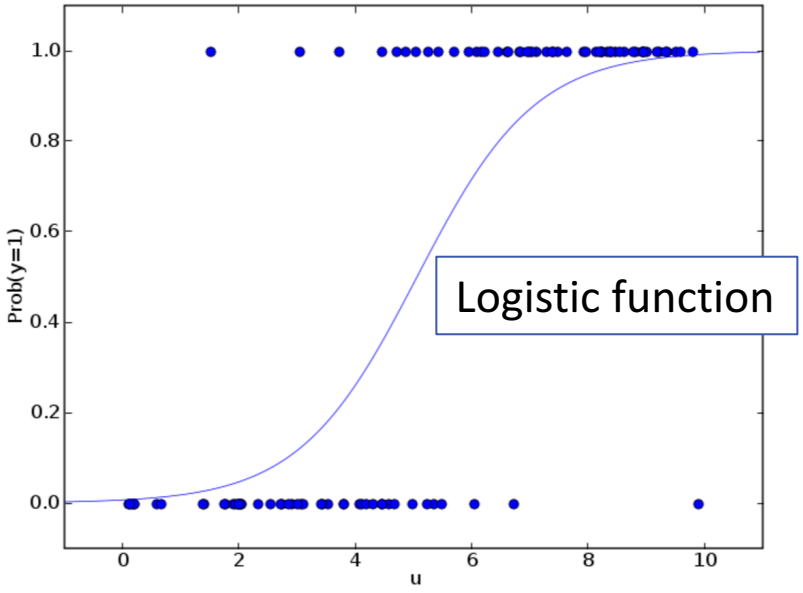


Logistic Regression

Regression



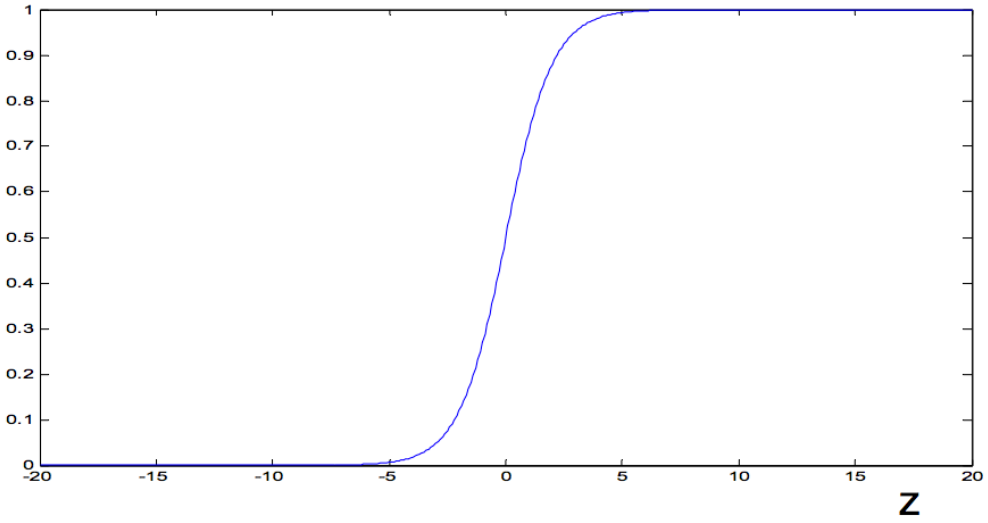
Logistic regression



logistic function / sigmoid function

- When $z \to \infty$ what is $\sigma(z)$?
- When $z \to -\infty$ what is $\sigma(z)$?
- When $z = 0$ what is $\sigma(z)$?

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

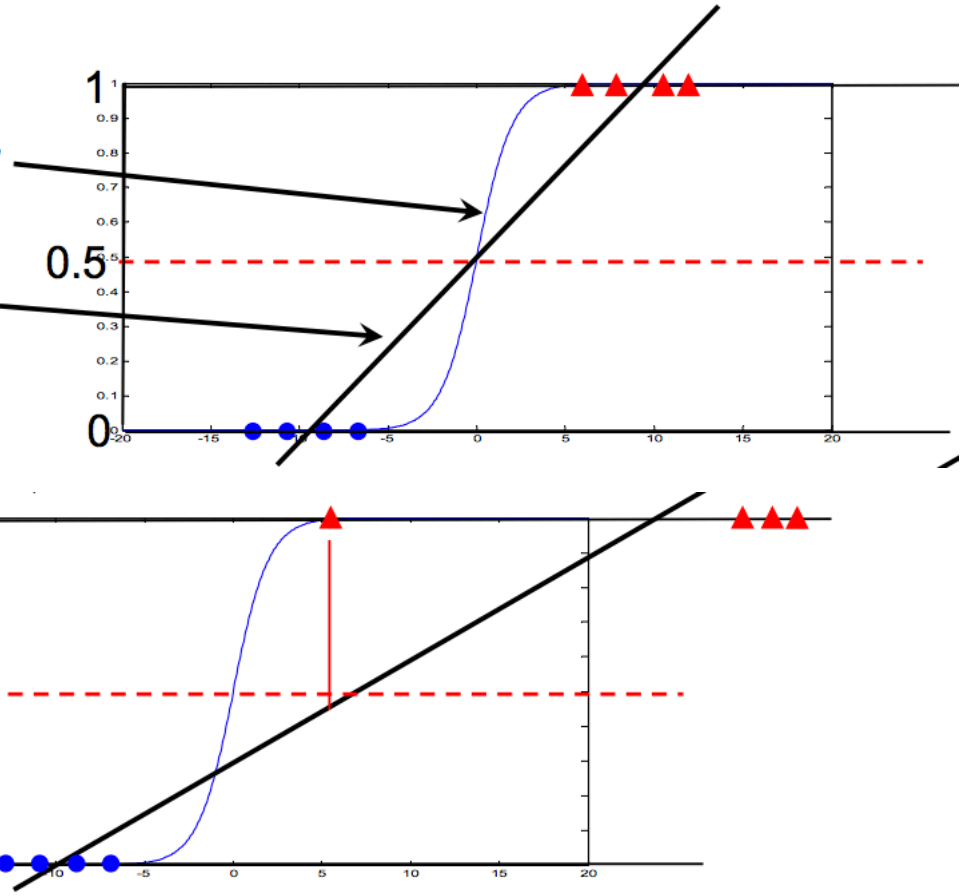


Why sigmoid?

Least squares fit

$\sigma(wx + b)$ fit to y

$wx + b$ fit to y



Probabilistic Interpretation

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \log(1 + e^{-\mathrm{y_i}(\mathbf{w}^\mathrm{T}\mathbf{x_i})})$$

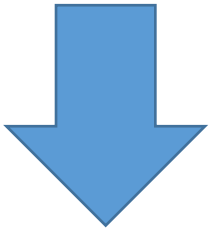
Assume labels are generated using the following probability distribution:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

$$P(y = -1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}}$$

$$P(y=1|\mathbf{x},\mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

$$P(y=-1|\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}}$$



$$P(y|\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp\left(-y\mathbf{w}^T\mathbf{x}\right)}$$

How to make prediction?

Predict y=1 if P(y=1|x,w) > p (y= -1|x, w)

Decision boundary?

$$P(y=1|\mathbf{x},\mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

$$P(y=-1|\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}}$$

$$\log \frac{P(Y=1 \mid x)}{P(Y=-1 \mid x)} = w^T x + b.$$

- The decision boundary?

$$w^T x + b = 0.$$

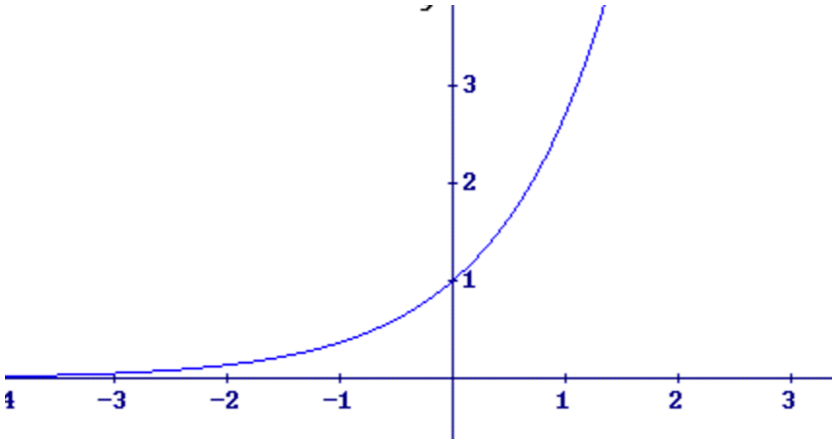
Alternative view

- Predict y=1 if P(y=1|x,w) > P(y= -1|x, w)

- Predict y=1 if P(y=1|x,w) > P(y= -1|x, w)

- When does this happen?
  - $\frac{1}{1+\exp(-w^T x)} > 0.5$
    $\Rightarrow 1 + \exp(-w^T x) < 2$
    $\Rightarrow \exp(-w^T x) < 1$
    $\Rightarrow w^T x > 0$

- When does this happen?
  - $\frac{1}{1+\exp(-w^T x)} > 0.5$
    $\Rightarrow 1 + \exp(-w^T x) < 2$
    $\Rightarrow \exp(-w^T x) < 1$
    $\Rightarrow w^T x > 0$



Maximum likelihood estimation

- Probabilistic model assumption:

$$P(y|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp\left(-y\mathbf{w}^T\mathbf{x}\right)}$$

- The log-likelihood of seeing a dataset D = {(x , y)} if the true weight vector was w:

$$\log P(D|\mathbf{w}) = -\sum \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x})\right)$$

$$P(D|w) = \Pi_i\, P(y_i|x_i, w)$$
$$\Rightarrow \log P(D|w) = \sum_i \log P(y_i|x_i, w)$$

Minimizing negative log-likelihood

- Log likelihood

$$\log P(D|\mathbf{w}) = -\sum \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x})\right)$$

- Logistic regression

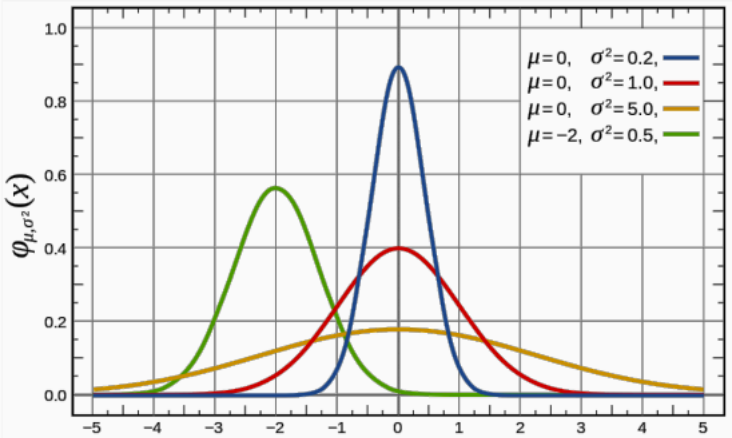
$$\min_{\boldsymbol{w},b} \quad \sum_i \log(\,1 + e^{-\mathrm{y_i}(\mathbf{w}^\mathrm{T}\mathbf{x_i})})$$

- Let's add some prior
  - Simpler is better ⇒ add Gaussian Prior

Add Gaussian Prior

- Simpler is better ⇒ add Gaussian Prior
- Suppose each element in w is drawn independently from the normal distribution centered at zero with variance σ^2
- Bias towards smaller weights

$$P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)$$



Source: Wikipedia

Regularized Logistic regression

$$P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)$$

- Remember we are in the log space

$$\log P(\mathbf{w}) = -\frac{1}{2\sigma^2}\mathbf{w}^T\mathbf{w} + \text{constant terms}$$

- Put them together
  - $P(w|D) \propto P(w, D) = P(D\,|w)P(w)$
  - Learning:
    Find weight vector by maximizing the posterior distribution P(w | D)

Maximum a posteriori estimation

- Put them together
  - $P(w|D) \propto P(w, D) = P(D\,|w)P(w)$
- Learning: Find weight vector by maximizing the posterior distribution P(w | D)

$$\log P(D, \mathbf{w}) = -\frac{1}{2\sigma^2}\mathbf{w}^T\mathbf{w} - \sum_i \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x})\right)$$

prior

Log-likelihood

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \log(\,1 + e^{-\mathrm{y_i}(\mathbf{w}^\mathrm{T}\mathbf{x}_\mathrm{i})})$$

Regularized loss minimization

- L1-Loss SVM

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$

- L2-Loss SVM

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))^2$$

- Logistic Regression (regularized)

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \log(\,1 + e^{-\mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i})})$$

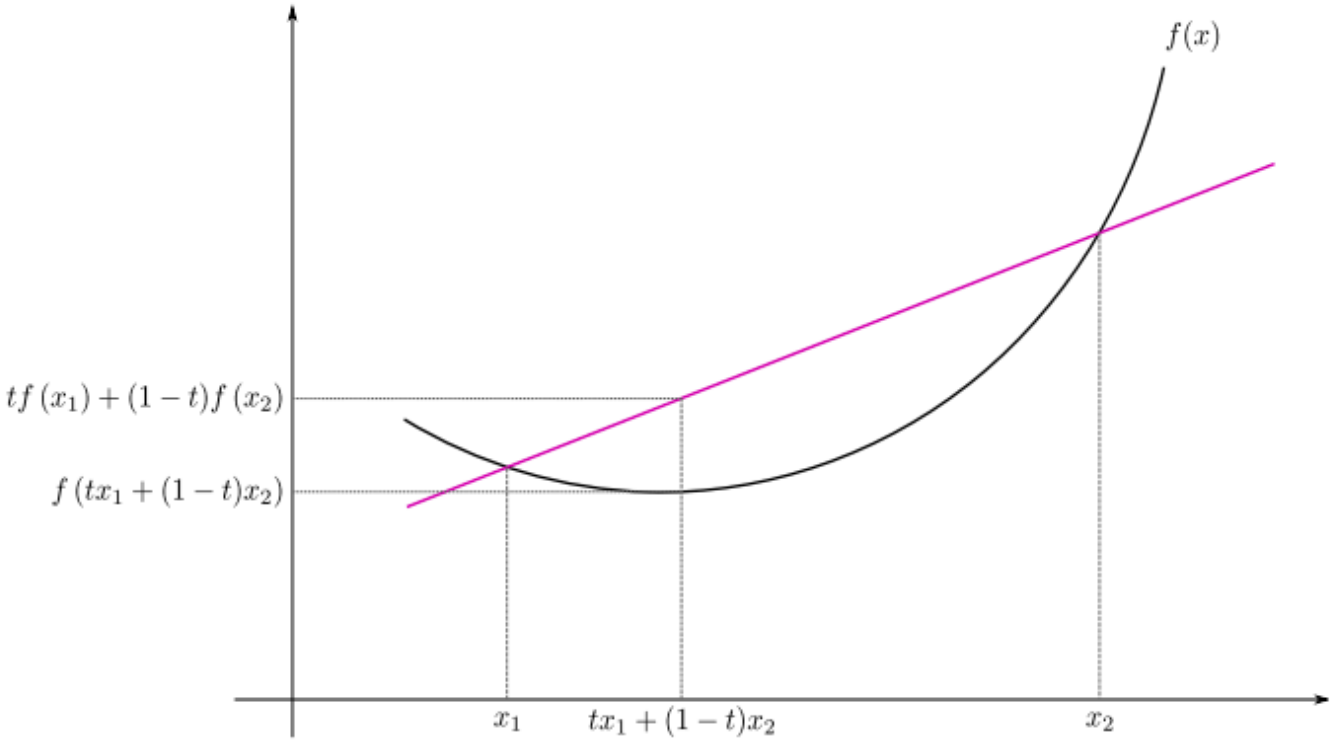
- Loss over training data + regularizer

How to learn (how to optimize the objective function?)

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$

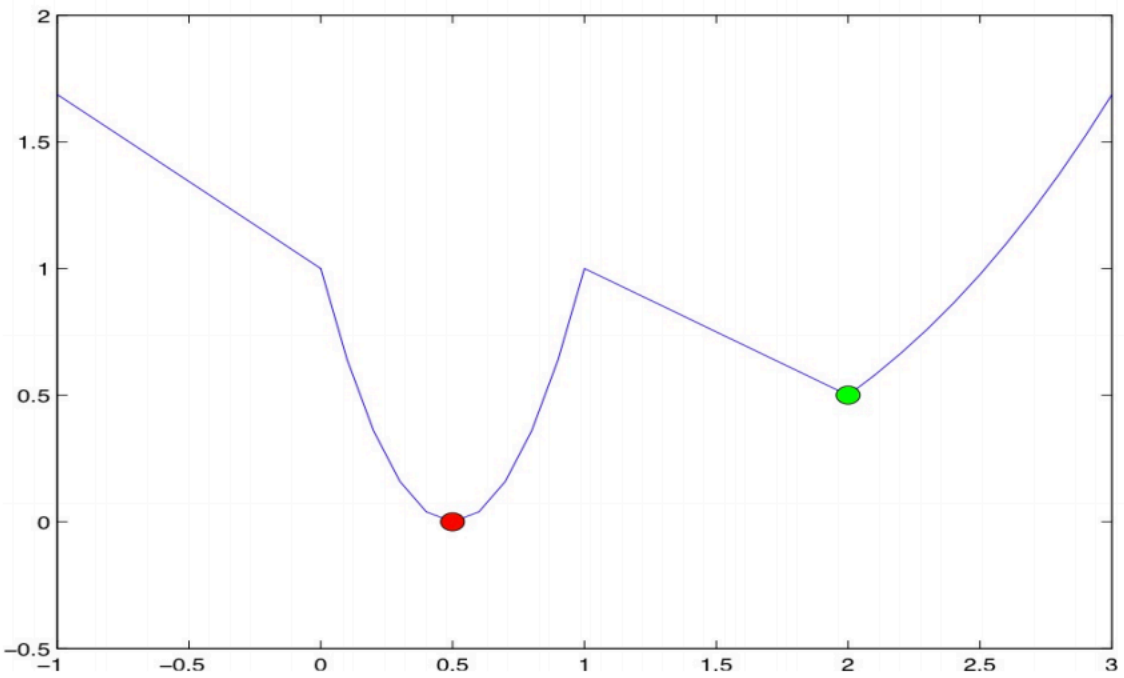
- This function is convex
- Many convex optimization methods can be used
  - Stochastic (sub)-gradient descent
  - Coordinate descent methods
  - Newton methods
  - LBFGS

Convexity



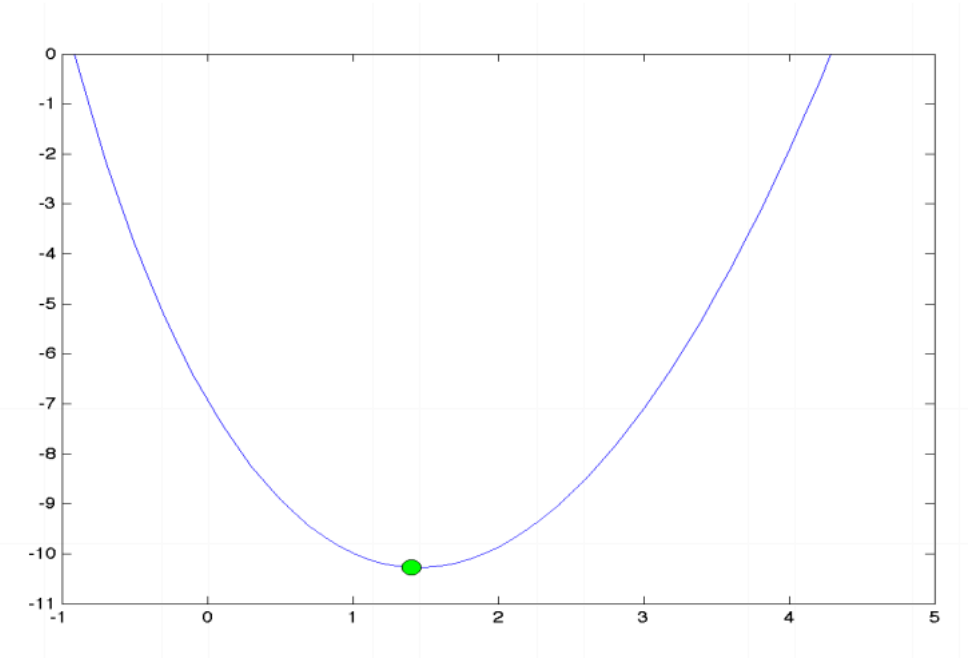
Non convex minimization is hard

- You may end up with some local minimum

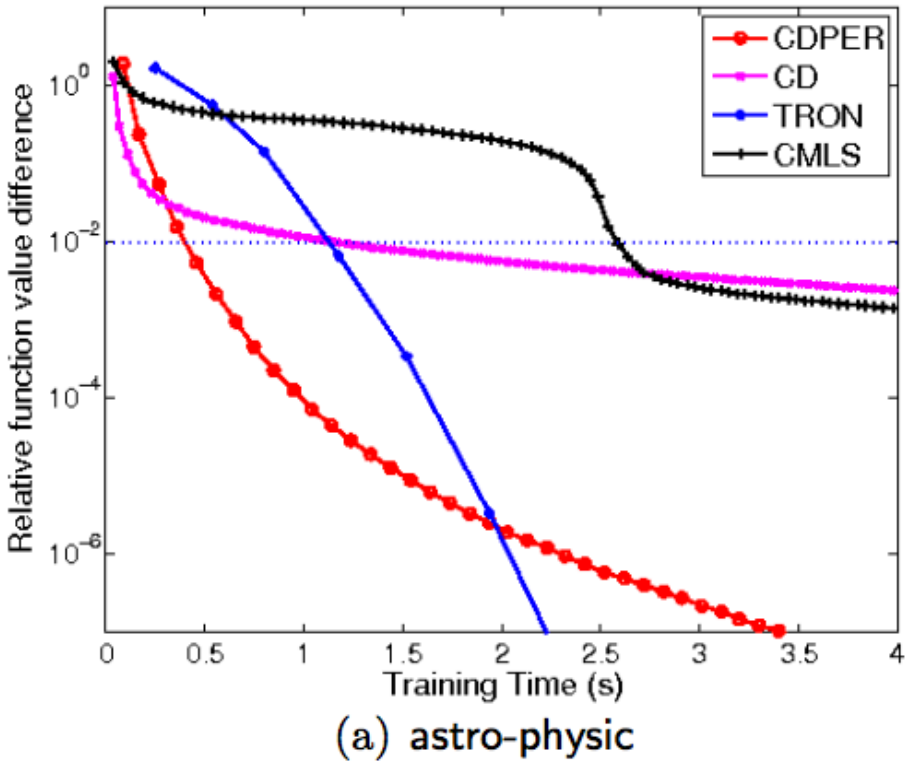


Convex optimization is relatively easy

- Ensure that there are no local minima
- Note: need special design for functions that are not differentiable (e.g., hinge loss)



Different optimization techniques



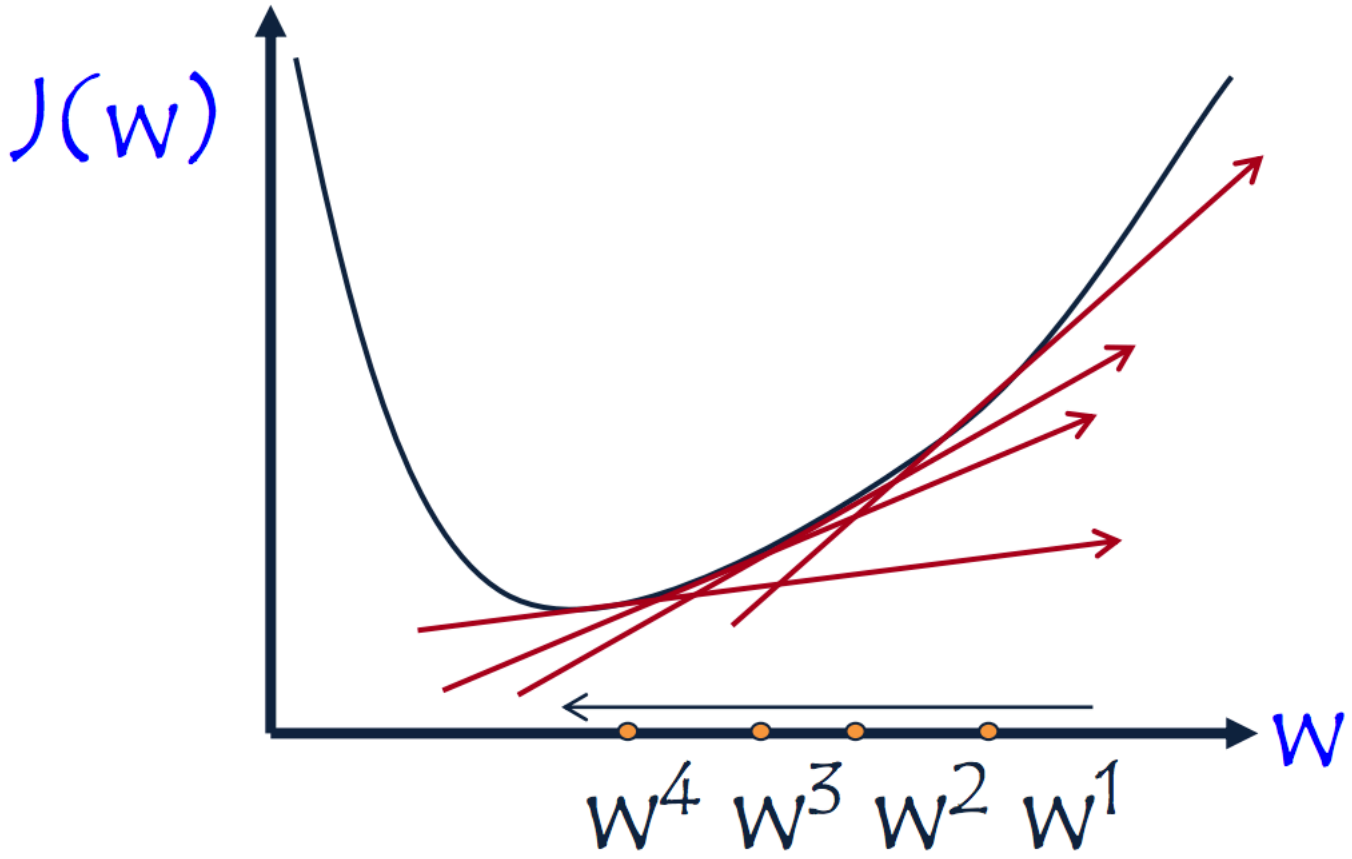
(a) astro-physic

Some methods (e.g., SGD, CD) are fast in the early stage of optimization

Some methods (e.g., Newton methods) converge faster

Results from http://www.cs.virginia.edu/~kc2wc/papers/ChangHsLi08.pdf

Gradient descent



Why not gradient descent?

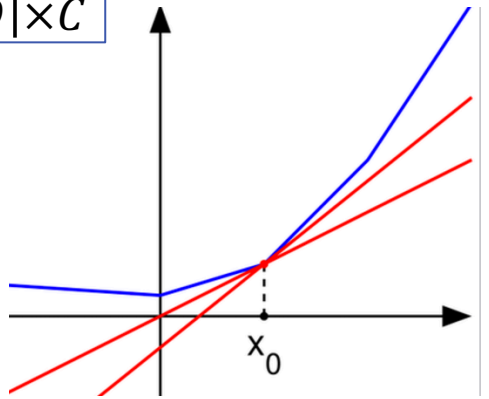
- For some functions, gradient may not exist

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \ \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$

- Solution: use sub-gradient

$$f(w) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \ \mathrm{y_i}\mathbf{w}^{\mathrm{T}}\mathbf{x_i})$$
$$= \frac{\mathbf{1}}{|\boldsymbol{D}|}\sum_i(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C'\max(0,\ 1 - \ \mathrm{y_i}\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$
$$\equiv \frac{\mathbf{1}}{|\boldsymbol{D}|}\sum_i f_i(w)$$
$$\nabla f(w) = \frac{1}{|D|}\sum_i \nabla f_i(w)$$
$$\nabla f_i(w) \equiv \begin{cases} w & if\ \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}) > 1 \\ w - C'y_ix_i & otherwise \end{cases},$$

$C' = |D| \times C$



Stochastic gradient descent

$$f(w) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}\mathbf{w}^{\mathrm{T}}\mathbf{x_i})$$
$$= \frac{\mathbf{1}}{|\boldsymbol{D}|}\sum_i(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C'\max(0,\ 1 - \mathrm{y_i}\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$
$$\equiv f_i(w)$$
$$\nabla f(w) = \frac{1}{|D|}\sum_i \nabla f_i(w) = E_{i\sim D}\nabla f_i(w)$$

- Approximate the true gradient by a gradient at a single example at a time

Repeat until converge:
  Randomly pick one sample (x_i, y_i)
  Update $\mathrm{w} \leftarrow \mathrm{w} - \eta\nabla\mathrm{f_i(w)}$

Stochastic Sub-gradient Descent

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. Update $w \leftarrow w - \eta \nabla f(w)$
5. Return $\boldsymbol{w}$

$$f(w) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i}))$$

Stochastic (sub)-gradient descent for SVM

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{1}$
5. $\boldsymbol{w} \leftarrow (1 - \eta)\boldsymbol{w} + \eta\, C\, y\boldsymbol{x}$
6. else
7. $\boldsymbol{w} \leftarrow (1 - \eta)\boldsymbol{w}$
8. Return $\boldsymbol{w}$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

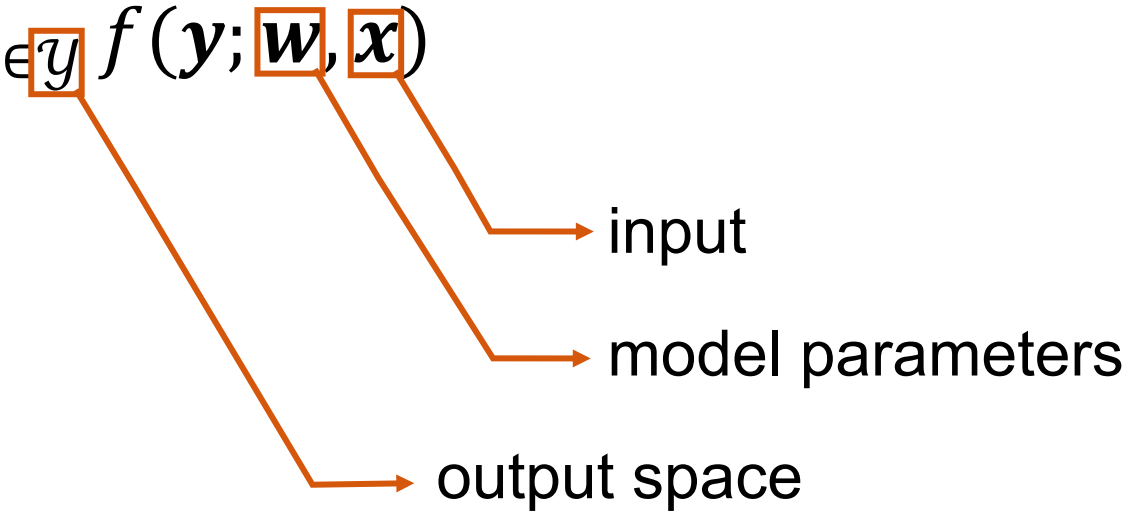
Prediction: y^{test}

Perceptron effectively minimizing:

$$\sum_i \max(0,\ 1 - \ y_i(\mathbf{w}^\mathrm{T}\mathbf{x}_i))$$

A General Formula

$$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$$



- Inference/Test: given $\boldsymbol{w}, \boldsymbol{x}$, solve argmax
- Learning/Training: find a good $\boldsymbol{w}$
- Today: $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}, \mathcal{Y} = \{-1, 1\}$ (binary classification)

Binary Linear Classifiers

$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$

- $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}, \mathcal{Y} = \{-1, 1\}$
- $f(y; \boldsymbol{w}, \boldsymbol{x}) \stackrel{\text{def}}{=} y(\boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b}) = y(\sum_i w_i x_i + \boldsymbol{b})$
- $\text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x}) = \begin{cases} 1, \boldsymbol{w}^\top \boldsymbol{x} + b \geq 0 \\ -1, \boldsymbol{w}^\top \boldsymbol{x} + b < 0 \end{cases}$
  $= \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b})$

(break ties arbitrarily)