

Linear classifiers: Outline

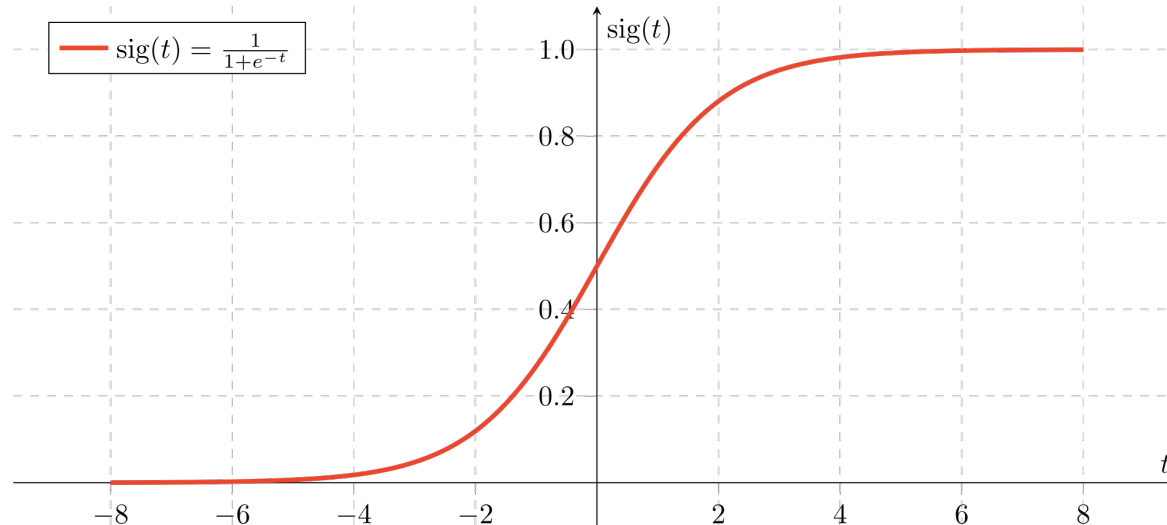
- Empirical loss minimization framework
- Linear classification models
 1. Linear regression
 2. Logistic regression
 3. Perceptron training algorithm
 4. Support vector machines
- Multi-class classification
 - Multi-class perceptrons
 - Multi-class SVM
 - Softmax

Review: Logistic regression

Review: Logistic regression

- Probability of the input x having a positive label is given by the *sigmoid function* of the linear response $w^T x$:

$$P_w(y = 1|x) = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)}$$



Review: Logistic loss

- Given: $\{(x_i, y_i), i = 1, \dots, n\}, y_i \in \{-1, 1\}$
- Classifier model: $P_w(y = 1|x) = \sigma(w^T x)$
- Maximum (conditional) likelihood estimate: find w that maximizes

$$\prod_{i=1}^n P_w(y_i|x_i)$$

- Equivalently, find w that minimizes

$$\hat{L}(w) = -\frac{1}{n} \sum_{i=1}^n \log P_w(y_i|x_i)$$

Review: Logistic loss

- Per-point loss:

$$l(w, x_i, y_i) = -\log P_w(y_i|x_i)$$

- If $y_i = 1$:

$$P_w(y_i|x_i) = \sigma(w^T x_i)$$

- If $y_i = -1$:

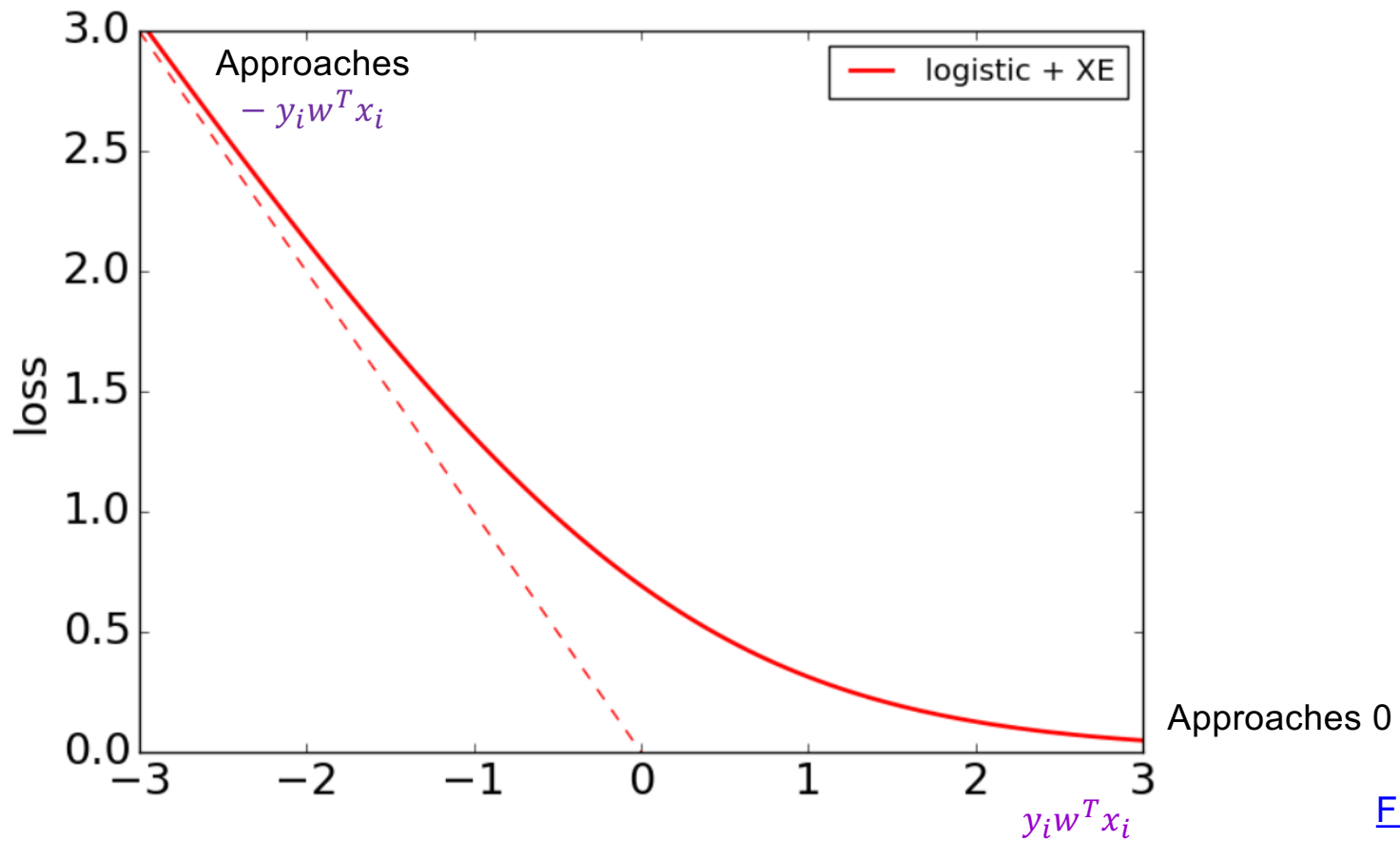
$$P_w(y_i|x_i) = 1 - \sigma(w^T x_i) = \sigma(-w^T x_i)$$

- Thus,

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$

Review: Logistic loss

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$



[Figure source](#)

Logistic loss: Optimization

- Given: $\{(x_i, y_i), i = 1, \dots, n\}, y_i \in \{-1, 1\}$
- Find w that minimizes

$$\begin{aligned}\hat{L}(w) &= -\frac{1}{n} \sum_{i=1}^n \log P_w(y_i | x_i) \\ &= -\frac{1}{n} \sum_{i=1}^n \log \sigma(y_i w^T x_i)\end{aligned}$$

- How do we minimize this loss?

Stochastic gradient descent (SGD)

- At each iteration, take a *single data point* (x_i, y_i) and perform a parameter update using $\nabla l(w, x_i, y_i)$, the gradient of the loss for that point:

$$w \leftarrow w - \eta \nabla l(w, x_i, y_i)$$

SGD for logistic regression

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$

- Let's find the gradient:

$$\begin{aligned}\nabla l(w, x_i, y_i) &= -\nabla_w \log \sigma(y_i w^T x_i) \\ &= -\frac{\nabla_w \sigma(y_i w^T x_i)}{\sigma(y_i w^T x_i)}\end{aligned}$$

- Derivative of log:

$$[\log(g(a))]' = \frac{g'(a)}{g(a)}$$

SGD for logistic regression

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$

- Let's find the gradient:

$$\begin{aligned}\nabla l(w, x_i, y_i) &= -\nabla_w \log \sigma(y_i w^T x_i) \\ &= -\frac{\nabla_w \sigma(y_i w^T x_i)}{\sigma(y_i w^T x_i)} \\ &= -\frac{\sigma(y_i w^T x_i) \sigma(-y_i w^T x_i) y_i x_i}{\sigma(y_i w^T x_i)}\end{aligned}$$

Derivative of sigmoid:

$$\sigma'(a) = \sigma(a)(1 - \sigma(a)) = \sigma(a)\sigma(-a)$$

SGD for logistic regression

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$

- Let's find the gradient:

$$\begin{aligned}\nabla l(w, x_i, y_i) &= -\nabla_w \log \sigma(y_i w^T x_i) \\ &= -\frac{\nabla_w \sigma(y_i w^T x_i)}{\sigma(y_i w^T x_i)} \\ &= -\frac{\sigma(y_i w^T x_i) \sigma(-y_i w^T x_i) y_i x_i}{\sigma(y_i w^T x_i)}\end{aligned}$$

- We also used the *chain rule*: $[g_2(g_1(a))]' = g_2'(g_1(a))g_1'(a)$

SGD for logistic regression

$$l(w, x_i, y_i) = -\log \sigma(y_i w^T x_i)$$

- Let's find the gradient:

$$\begin{aligned}\nabla l(w, x_i, y_i) &= -\nabla_w \log \sigma(y_i w^T x_i) \\ &= -\frac{\nabla_w \sigma(y_i w^T x_i)}{\sigma(y_i w^T x_i)} \\ &= -\frac{\sigma(y_i w^T x_i) \sigma(-y_i w^T x_i) y_i x_i}{\sigma(y_i w^T x_i)} \\ &= -\sigma(-y_i w^T x_i) y_i x_i\end{aligned}$$

- SGD update:

$$w \leftarrow w + \eta \sigma(-y_i w^T x_i) y_i x_i$$

SGD for logistic regression

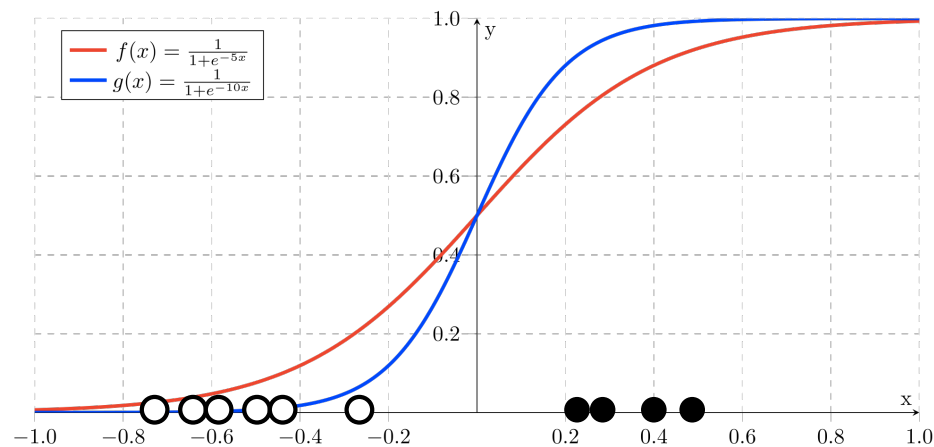
- Let's take a closer look at the SGD update:

$$w \leftarrow w + \eta \sigma(-y_i w^T x_i) y_i x_i$$

- What happens if x_i is *incorrectly*, but confidently, classified?
 - The signs of $w^T x_i$ and y_i are *opposite* and $\sigma(-y_i w^T x_i)$ approaches 1
 - The update rule approaches $w \leftarrow w + \eta y_i x_i$
- What happens if x_i is *correctly*, and confidently, classified?
 - The signs of $w^T x_i$ and y_i are *the same* and $\sigma(-y_i w^T x_i)$ approaches 0
 - The update approaches 0 – *but never actually reaches it*

SGD for logistic regression

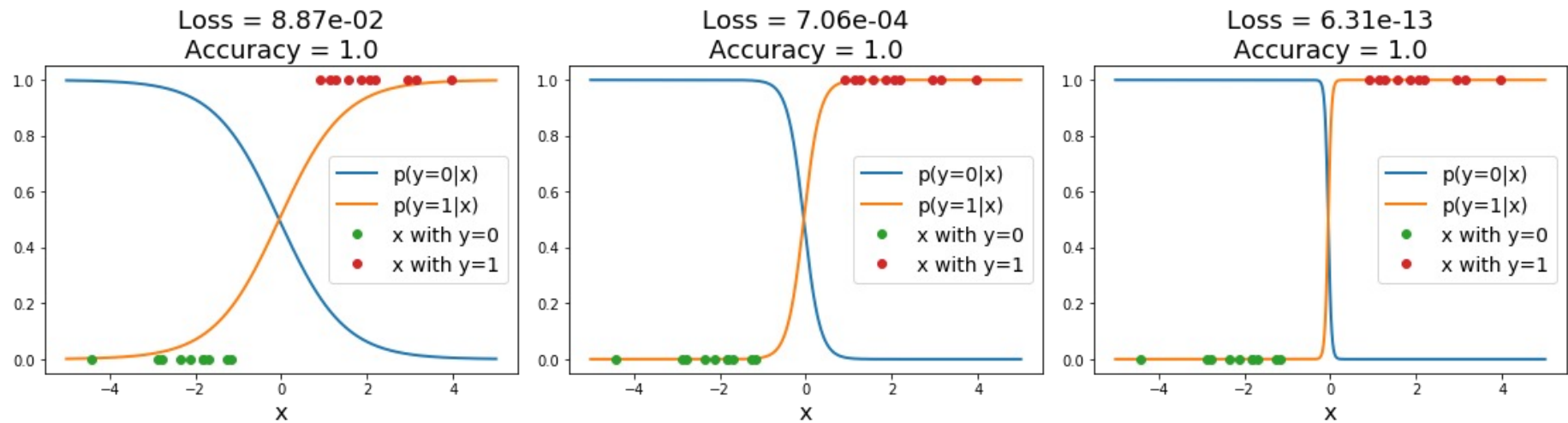
- Logistic regression *does not converge* for linearly separable data!
 - Scaling w by ever larger constants makes the classifier more confident and keeps increasing the likelihood of the data



[Image source](#)

SGD for logistic regression

- Logistic regression *does not converge* for linearly separable data!
 - Scaling w by ever larger constants makes the classifier more confident and keeps increasing the likelihood of the data



Source: [J. Johnson](#)

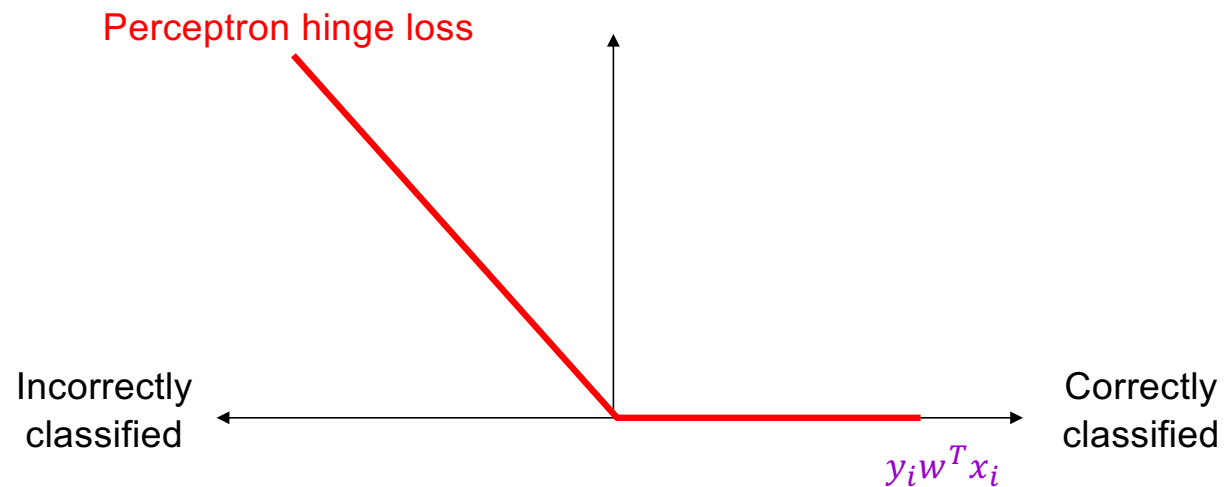
Linear classifiers: Outline

- Empirical loss minimization framework
- Linear classification models
 1. Linear regression
 2. Logistic regression
 3. Perceptron training algorithm
 4. Support vector machines

Perceptron

- Let's define the *perceptron hinge loss*:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$



Perceptron

- Let's define the *perceptron hinge loss*:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

- Training: find w that minimizes

$$\hat{L}(w) = \frac{1}{n} \sum_{i=1}^n l(w, x_i, y_i) = \frac{1}{n} \sum_{i=1}^n \max(0, -y_i w^T x_i)$$

- Once again, there is no closed-form solution, so let's go straight to SGD

Deriving the perceptron update

- Let's differentiate the perceptron hinge loss:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

(Strictly speaking, this loss is not differentiable, so we need to find a *sub-gradient*)

Deriving the perceptron update

- Let's differentiate the perceptron hinge loss:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

$$\nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0] y_i x_i$$

$$\frac{d}{da} \max(0, a) =$$

Deriving the perceptron update

- Let's differentiate the perceptron hinge loss:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

$$\nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0] y_i x_i$$

- We also used the chain rule: $[g_2(g_1(a))]' = g_2'(g_1(a))g_1'(a)$

Deriving the perceptron update

- Let's differentiate the perceptron hinge loss:

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

$$\nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0] y_i x_i$$

- Corresponding SGD update ($w \leftarrow w - \eta \nabla l(w, x_i, y_i)$):

$$w \leftarrow w + \eta \mathbb{I}[y_i w^T x_i < 0] y_i x_i$$

- What if x_i is *correctly* classified?
 - Do nothing
- What if x_i is *incorrectly* classified?
 - Update becomes $w \leftarrow w + \eta y_i x_i$

Understanding the perceptron update rule

- **Perceptron update rule:** If $y_i \neq \text{sgn}(w^T x_i)$ then update weights:

$$w \leftarrow w + \eta y_i x_i$$

- The raw response of the classifier was $w^T x_i$ and now it becomes

$$(w + \eta y_i x_i)^T x_i = w^T x_i + \eta y_i \|x_i\|^2$$

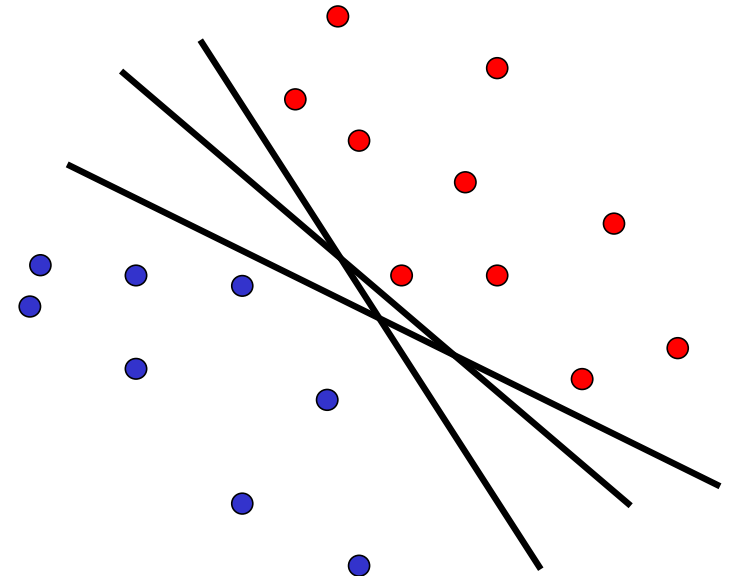
- How does the response change if $y_i = 1$?
 - The response $w^T x_i$ is initially *negative* and will be *increased*
- How does the response change if $y_i = -1$?
 - The response $w^T x_i$ is initially *positive* and will be *decreased*

Linear classifiers: Outline

- Empirical loss minimization framework
- Linear classification models
 1. Linear regression
 2. Logistic regression
 3. Perceptron training algorithm
 4. Support vector machines

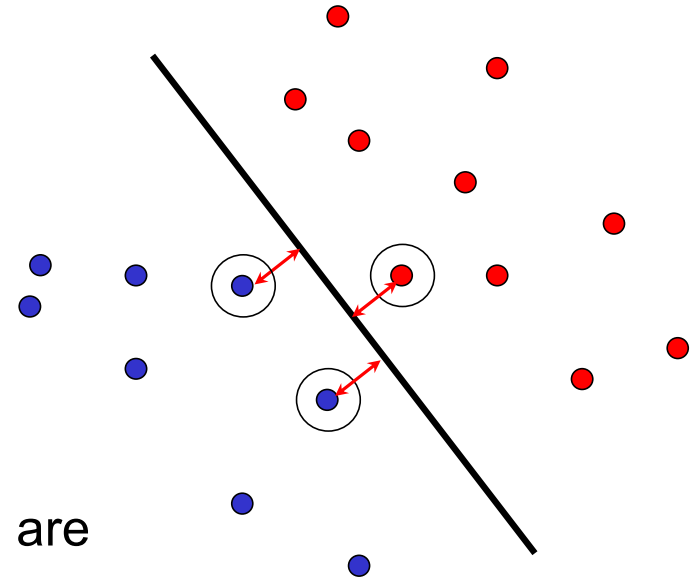
Support vector machines

- When the data is linearly separable, which of the many possible solutions should we prefer?
- **Perceptron training algorithm:**
no special criterion, solution depends on initialization



Support vector machines

- When the data is linearly separable, which of the many possible solutions should we prefer?
 - **Perceptron training algorithm:**
no special criterion, solution depends on initialization
 - **SVM criterion:** maximize the *margin*, or distance between the hyperplane and the closest training example
 - The examples closest to the hyperplane are known as *support vectors*



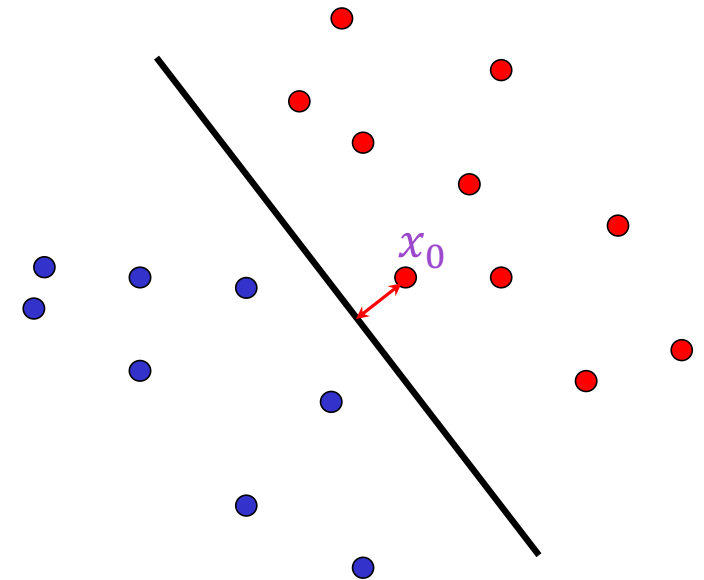
Finding the maximum margin hyperplane

- We want to maximize the margin, or distance between the hyperplane $w^T x = 0$ and the closest training example x_0
- This distance is given by

$$\frac{|w^T x_0|}{\|w\|} \quad (\text{derivation})$$

- Assuming the data is linearly separable, we can fix the scale of w so that $|w^T x_0| = 1$ and $|w^T x_i| \geq 1$ for all other points

- Then the margin becomes $\frac{1}{\|w\|}$



Finding the maximum margin hyperplane

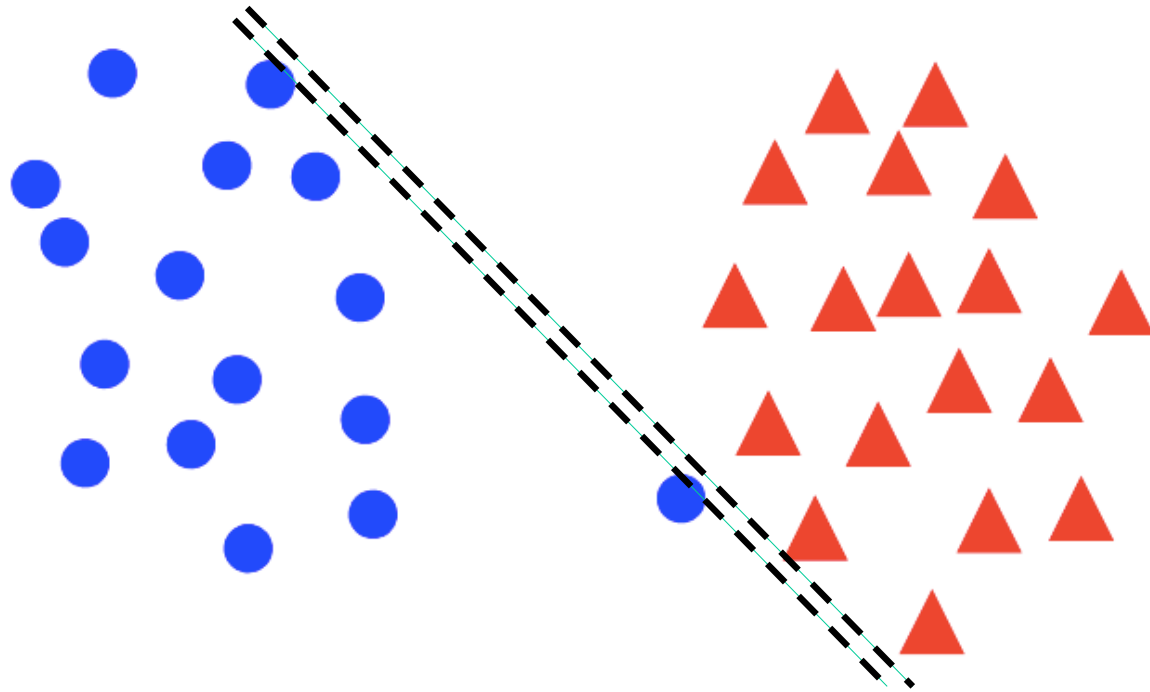
- We want to maximize margin $\frac{1}{\|w\|}$ while correctly classifying all training data: $y_i w^T x_i \geq 1$
- Equivalent problem:

$$\min_w \frac{1}{2} \|w\|^2 \quad \text{s. t.} \quad y_i w^T x_i \geq 1 \quad \forall i$$

- This is a quadratic objective with linear constraints: convex optimization problem, global optimum can be found using well-studied methods

“Soft margin” formulation

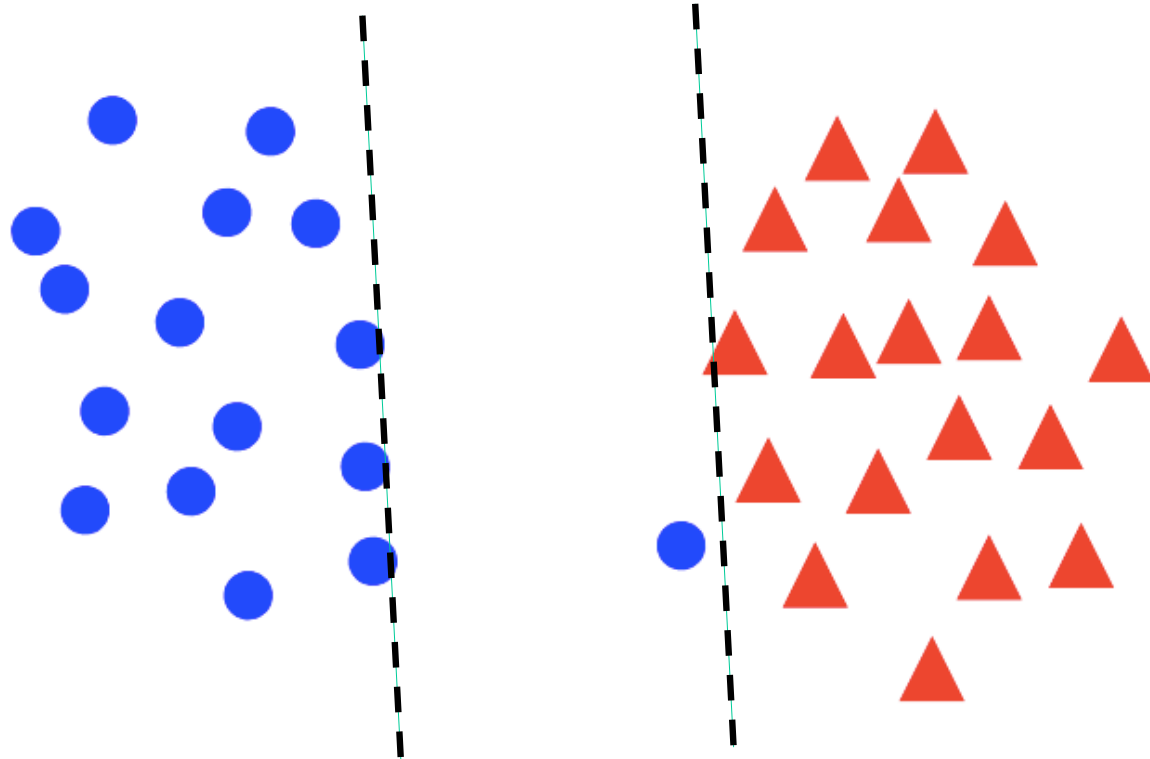
- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated



[Source](#)

“Soft margin” formulation

- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated

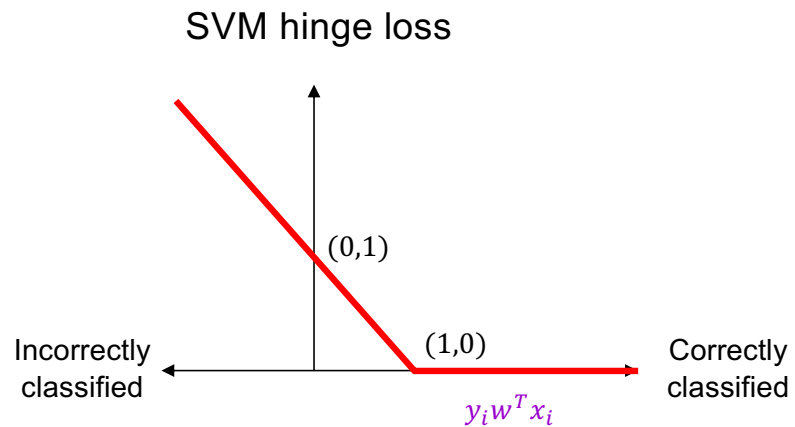
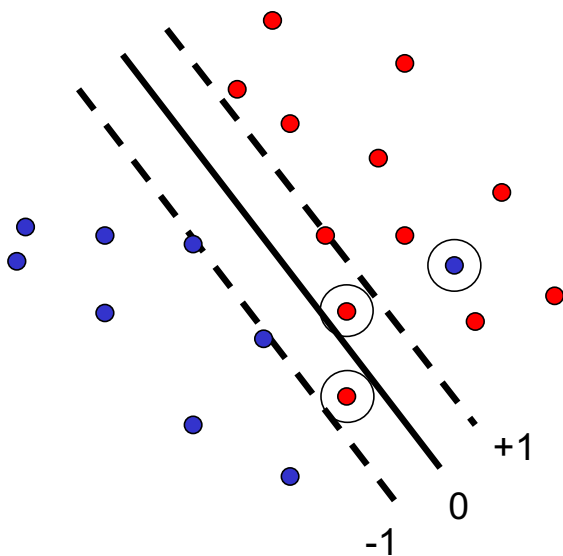


[Source](#)

SVM loss

- Trade off maximizing margin against penalizing margin violations:

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^n \max[0, 1 - y_i w^T x_i]$$

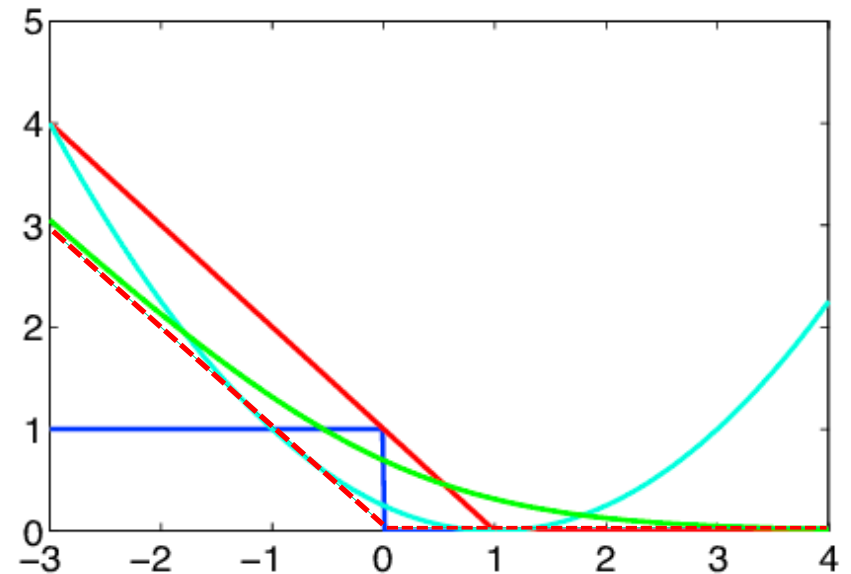


General recipe

- Find parameters w that minimize the sum of a *regularization loss* and a *data loss*:

$$\hat{L}(w) = \lambda R(w) + \frac{1}{n} \sum_{i=1}^n l(w, x_i, y_i)$$

empirical loss regularization data loss



SGD update for SVM

$$l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i]$$

$$\nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i$$

$$\text{Recall: } \frac{d}{da} \max(0, a) = \mathbb{I}[a > 0]$$

SGD update for SVM

$$l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i]$$

$$\nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i$$

- SGD update:
 - If $y_i w^T x_i \geq 1$: $w \leftarrow w - \eta \frac{\lambda}{n} w$ or $w \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w$
 - If $y_i w^T x_i < 1$: $w \leftarrow w + \eta \left(y_i x_i - \frac{\lambda}{n} w\right)$ or $w \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w + \eta y_i x_i$

SVM vs. perceptron update

- SVM loss: $l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i]$
- SVM update:
 - If $y_i w^T x_i \geq 1$: $w \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w$
 - If $y_i w^T x_i < 1$: $w \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w + \eta y_i x_i$
- Perceptron loss: $l(w, x_i, y_i) = \max[0, -y_i w^T x_i]$
- Perceptron update:
 - If $y_i w^T x_i < 0$: $w \leftarrow w + \eta y_i x_i$
 - Otherwise: do nothing