Classification: multi class

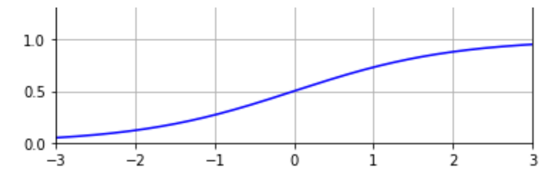
UNIVERSITY of WASHINGTON

Probabilistic interpretation of logistic regression

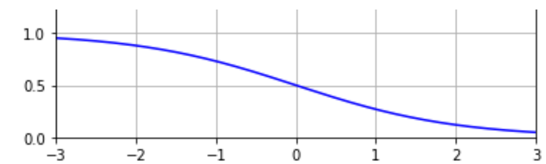
- just as Maximum Likelihood Estimator (MLE) under linear model and additive Gaussian noise model recovers **linear least squares**,
- we study a particular noise model that recovers **logistic regression** as MLE

- a probabilistic noise model for Binary labels:

$$\mathbb{P}(y_i = +1 \,|\, x_i) \;=\; \frac{1}{1+e^{-w^T x_i}}$$



$$\mathbb{P}(y_i = -1 \,|\, x_i) \;=\; \frac{1}{1+e^{w^T x_i}}$$



with a ground truth model parameter $w \in \mathbb{R}^d$

$w^T x_i$

- this function $\sigma(z) = \dfrac{1}{1+e^{-z}}$ is called a **logistic function** (not to be confused with logistic loss, which is different) or a **sigmoid function**
- if we know that the data came from such a model, but do not know the ground truth parameter $w \in \mathbb{R}^d$, we can apply MLE to find the best w
- this MLE recovers the logistic regression algorithm, exactly

Maximum Likelihood Estimator (MLE)

- if the data came from a probabilistic model model: $\Big(\underbrace{\frac{1}{1+e^{-w^Tx}}}_{\mathbb{P}(y_i=+1|x_i)}, \underbrace{\frac{1}{1+e^{w^Tx}}}_{\mathbb{P}(y_i=-1|x_i)}\Big)$

- log-likelihood of observing a data point (x_i, y_i) is

$$\text{log-likelihood} = \log\Big(\mathbb{P}(y_i|x_i)\Big) = \begin{cases} \log\left(\frac{1}{1+e^{-w^Tx_i}}\right) & \text{if } y_i = +1 \\ \log\left(\frac{1}{1+e^{w^Tx_i}}\right) & \text{if } y_i = -1 \end{cases}$$

- Maximum Likelihood Estimator is the one that maximizes the sum of all log-likelihoods on training data points

$$\hat{w}_{\text{MLE}} = \arg\max_w \ \mathbb{P}(\{y_1, \ldots, y_n\} \,|\, \{x_1, \ldots, x_n\})$$

$$= \arg\max_w \prod_{i=1}^{n} \mathbb{P}(y_i \,|\, x_i)$$ **(independence)**

$$= \arg\max_w \sum_{i:y_i=-1} \log\Big(\frac{1}{1+e^{w^Tx_i}}\Big) + \sum_{i:y_i=1} \log\Big(\frac{1}{1+e^{-w^Tx_i}}\Big)$$ **(substitution)**

- notice that this is exactly the **logistic regression:**

$$\hat{w}_{\text{logistic}} = \arg\min_{w} \frac{1}{n}\Big(\sum_{i:y_i=-1} \log(1+e^{w^T x_i}) + \sum_{i:y_i=1} \log(1+e^{-w^T x_i}) \Big)$$

- once we have trained a model $\hat{w}_{\text{logistic}}$, we can make a hard prediction $\hat{v}$ of the label at an input example x

$$\hat{v} = \begin{cases} +1 & \text{if } \mathbb{P}(+1|x) \geq \mathbb{P}(-1|x) \\ -1 & \text{otherwise} \end{cases}$$

$$= \begin{cases} +1 & \text{if } \frac{1}{1+e^{-w^T x}} \geq \frac{1}{1+e^{w^T x}} \\ -1 & \text{otherwise} \end{cases}$$

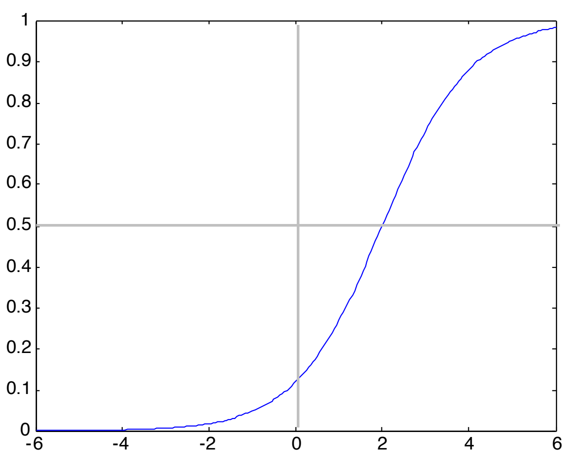
$$= \begin{cases} +1 & \text{if } 1 \leq e^{2w^T x} \\ -1 & \text{otherwise} \end{cases}$$

$$= \text{sign}(w^T x)$$

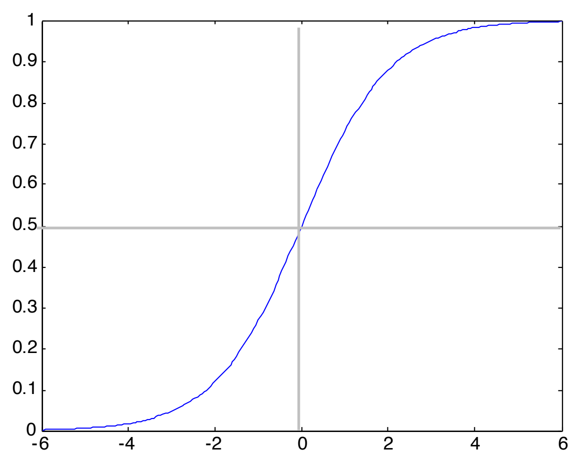
Understanding the sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

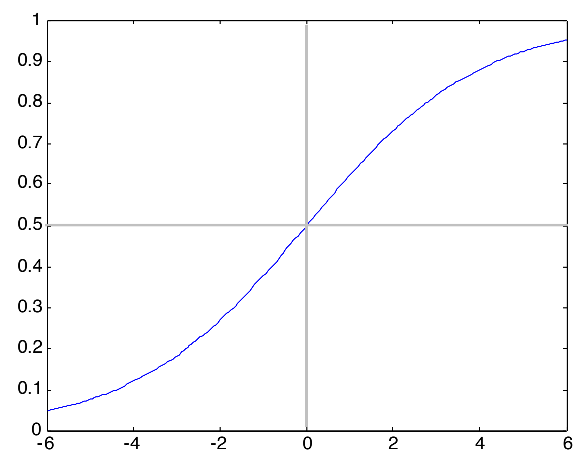
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5

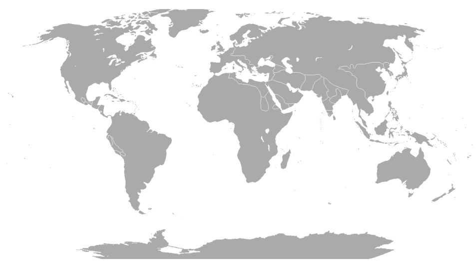


Multi-class regression

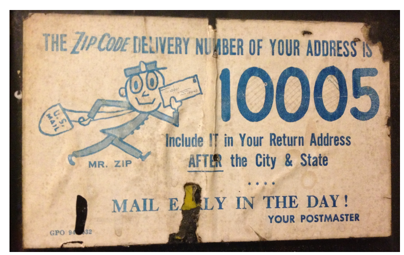
How do we encode categorical data *y*?

- so far, we considered Binary case where there are two categories
- encoding y is simple: {+1,-1}

- multi-class classification predicts categorial y
- taking values in $C = \{c_1, \ldots, c_k\}$
- c_j's are called **classes** or **labels**
- examples:



Country of birth
(Argentina, Brazil, USA,...)



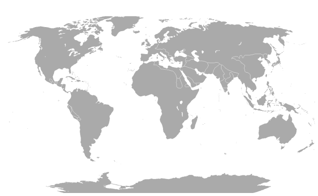
Zipcode
(10005, 98195,...)

All English words

- a **k-class classifier** predicts y given x

Embedding c_j's in real values

- for optimization we need to **embed** raw categorical c_j's into real valued vectors
- there are many ways to embed categorial data
  - True->1, False->-1
  - Yes->1, Maybe->0, No->-1
  - Yes->(1,0), Maybe->(0,0), No->(0,1)
  - Apple->(1,0,0), Orange->(0,1,0), Banana->(0,0,1)
  - Ordered sequence:
    (Horse 3, Horse 1, Horse 2) -> (3,1,2)
- we use **one-hot embedding** (a.k.a. **one-hot encoding**)
  - each class is a standard basis vector in $k-$dimension



Country of birth
(Argentina, Brazil, USA,...)

196 categories

1-hot encoding

x	$h_1(x)$	$h_2(x)$	...	$h_{195}(x)$	$h_{196}(x)$
Brazil		1			
Zimbabwe					1

196 features

Multi-class logistic regression

- data: categorical y in $\{c_1, \ldots, c_k\}$ with k categories

  we use one-hot encoding, s.t. $y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ implies that $y = c_1$

- model: linear vector-function makes a linear prediction $\hat{y} \in \mathbb{R}^k$

$$\hat{y}_i \;=\; f(x_i) \;=\; w^T x_i \quad \in \mathbb{R}^k$$

with model parameter matrix $w \in \mathbb{R}^{d \times k}$ and sample $x_i \in \mathbb{R}^d$

$$f(x_i) \;=\; \begin{bmatrix} f_1(x_i) \\ f_2(x_i) \\ \vdots \\ f_k(x_i) \end{bmatrix} \;=\; \underbrace{\begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & \cdots \\ w_{2,0} & w_{2,1} & w_{2,2} & \cdots \\ \vdots & & & \\ w_{k,0} & w_{k,1} & w_{k,2} & \cdots \end{bmatrix}}_{w^T} \underbrace{\begin{bmatrix} 1 \\ x_i[1] \\ \vdots \\ x_i[d] \end{bmatrix}}_{x_i} \;=\; \begin{bmatrix} w_{1,0} + w_{1,1}x_i[1] + w_{1,2}x_i[2] + \cdots \\ w_{2,0} + w_{2,1}x_i[1] + w_{2,2}x_i[2] + \cdots \\ \vdots \\ w_{k,0} + w_{k,1}x_i[1] + w_{k,2}x_i[2] + \cdots \end{bmatrix}$$

$$w = \begin{bmatrix} w[:,1] & w[:,2] & \cdots & w[:,k] \end{bmatrix}$$

- Logistic regression

2 classes

$$\mathbb{P}(y_i = -1 \,|\, x_i) = \frac{1}{1 + e^{w^T x_i}}$$

$$\mathbb{P}(y_i = +1 \,|\, x_i) = \frac{1}{1 + e^{-w^T x_i}} = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}}$$

k classes

$$\mathbb{P}(y_i = c_1 \,|\, x_i) = \frac{e^{w[:,1]^T x_i}}{e^{w[:,1]^T x_i} + \cdots + e^{w[:,k]^T x_i}}$$

$$\vdots$$

$$\mathbb{P}(y_i = c_k \,|\, x_i) = \frac{e^{w[:,k]^T x_i}}{e^{w[:,1]^T x_i} + \cdots + e^{w[:,k]^T x_i}}$$

Without loss of generality setting w[:,1]=0 when $k = 2$ recovers the original binary class case

Maximum Likelihood Estimator

$$\text{maximize}_w \; \frac{1}{n} \sum_{i=1}^{n} \log(\mathbb{P}(y_i \,|\, x_i))$$

$$\text{maximize}_{w \in \mathbb{R}^d} \; \frac{1}{n} \sum_{i=1}^{n} \log\left(\frac{1}{1 + e^{-y_i w^T x_i}} \right)$$

$$\text{maximize}_{w \in \mathbb{R}^{d \times k}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbf{I}\{y_i = c_j\} \log\left(\frac{e^{w[:,j]^T x_i}}{\sum_{j'=1}^{k} e^{w[:,j']^T x_i}} \right)$$

$\mathbf{I}\{y_i = j\}$ is an indicator that is one only if $y_i = j$

Linear classification

- Learn: f:**X** —>Y
  - **X** – features
  - **Y – target classes**

$$Y \in \{-1, 1\}$$

- **Expected loss of f:**
-

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] = 1 - P(Y = f(x)|X = x)$$

- **Bayes optimal classifier:**

$$f(x) = \arg\max_y \mathbb{P}(Y = y|X = x)$$

- **Model of logistic regression:**

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

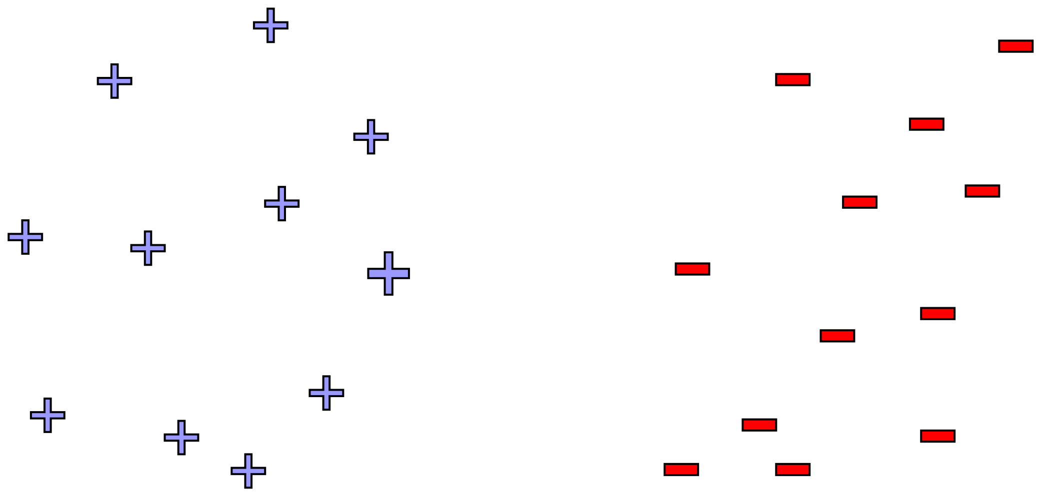
- **Loss function:**

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

What if the model is wrong?

Binary Classification

- Perceptron guaranteed to converge if
  - Data linearly separable:



Can we do classification without a model of $\mathbb{P}(Y=y|X=x)$?

The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: y in {-1,+1}
- Linear model
  - Prediction:

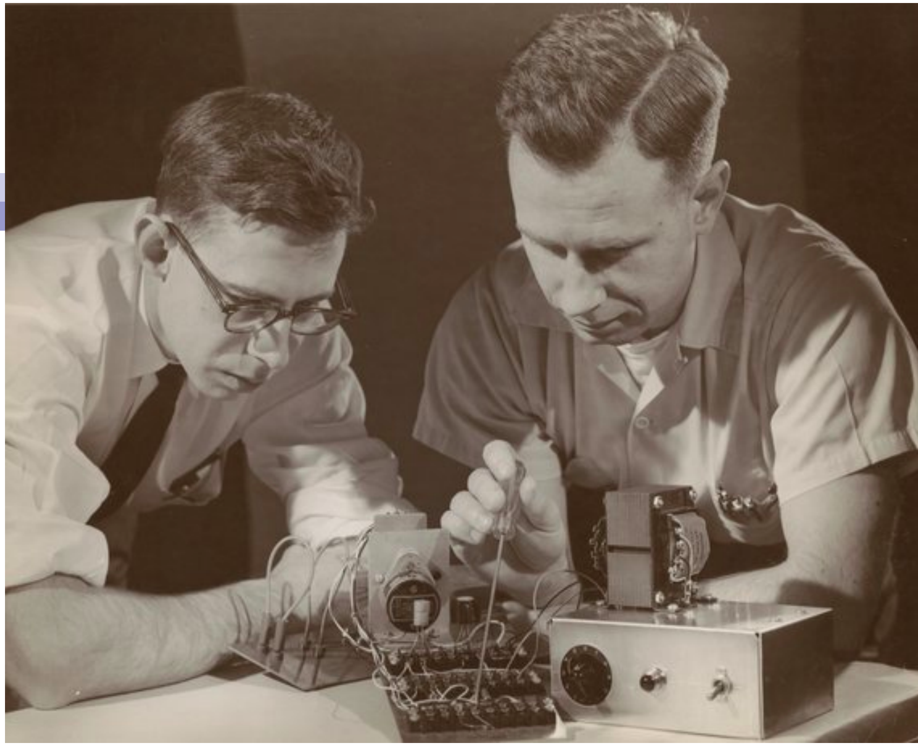
- Training:
  - Initialize weight vector:
  - At each time step:
    - Observe features:
    - Make prediction:
    - Observe true class:

    - Update model:
      - If prediction is not equal to truth

The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: y in {-1,+1}
- Linear model $\text{sign}(w^T x_i + b)$
  - Prediction:
- Training: $w_0 = 0, b_0 = 0$
  - Initialize weight vector:
  - At each time step: x_k
    - Observe features: $\text{sign}(x_k^T w_k + b_k)$
    - Make prediction:
    - Observe true class: y_k
    - Update model:
      - If prediction is not equal to truth

$$\begin{bmatrix} w_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} w_k \\ b_k \end{bmatrix} + y_k \begin{bmatrix} x_k \\ 1 \end{bmatrix}$$

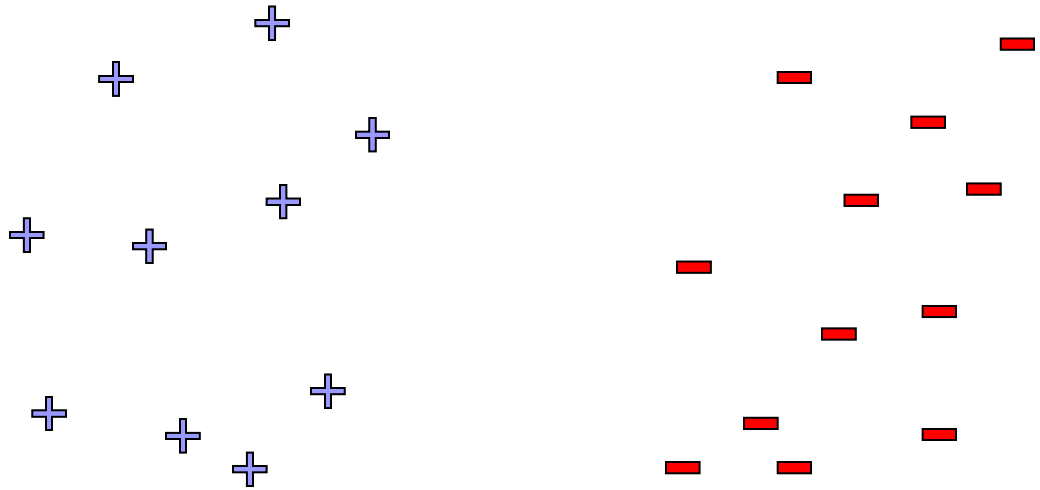


Rosenblatt 1957

"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."

The New York Times, 1958

Linear Separability



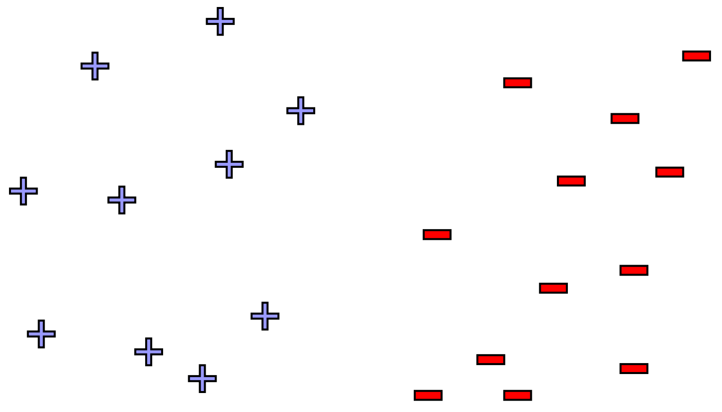
- Perceptron guaranteed to converge if
  - Data linearly separable:

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples:
  - Each feature vector has bounded norm:
  - If dataset is linearly separable:
- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

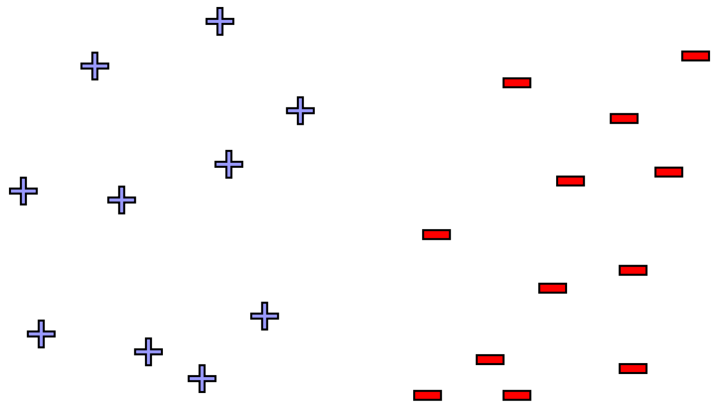
Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data



Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data



- Perceptron is useless in practice!
  - Real world not linearly separable
  - If data not separable, cycles forever and hard to detect
  - Even if separable may not give good generalization accuracy (small margin)

What is the Perceptron Doing???

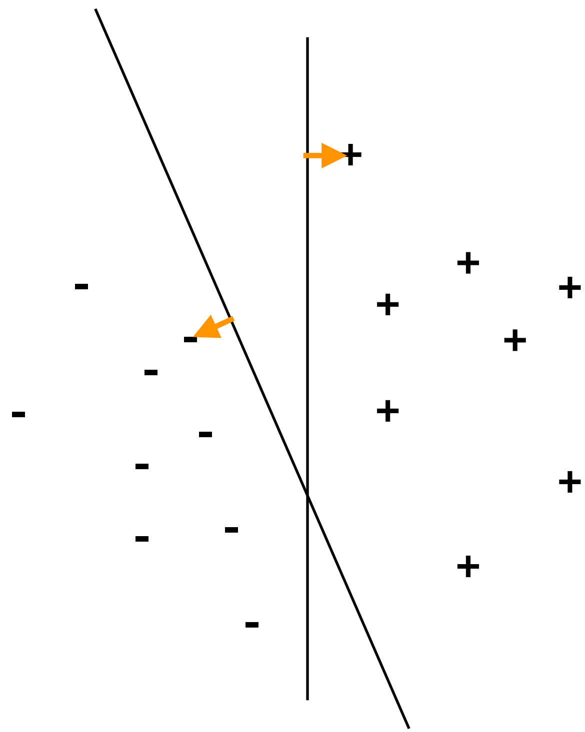
- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood
- When we discussed the Perceptron:
  - Started from description of an algorithm
- What is the Perceptron optimizing????

Lecture 16: Support Vector Machines

- how do we choose a better classifier?

How do we choose the best linear classifier?

- informally, **margin** of a set of examples to a decision boundary is the distance to the closest point to the decision boundary
- for linearly separable datasets, **maximum margin** classifier is a natural choice
- large margin implies that the decision boundary can change without losing accuracy, so the learned model is more robust against new data points

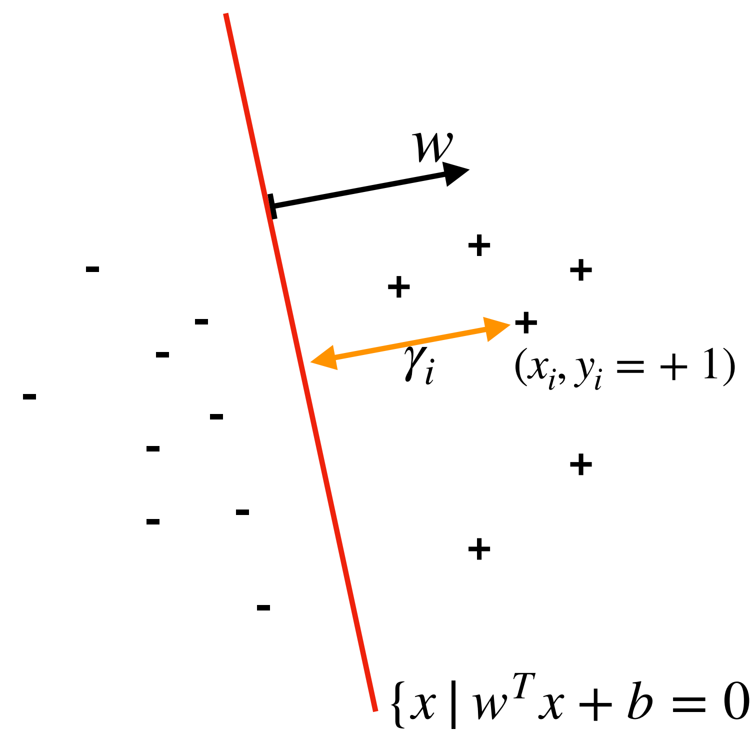


Geometric margin

- given a set of training examples $\{(x_i, y_i)\}_{i=1}^n$, with $y_i \in \{-1, +1\}$
- and a linear classifier $(w, b) \in \mathbb{R}^d \times \mathbb{R}$
- such that the decision boundary is
  a separating hyperplane $\{x \mid \underbrace{b + w_1 x[1] + w_2 x[2] + \cdots + w_d x[d]}_{w^T x + b} = 0\}$,
  which is the set of points that are orthogonal to w with a shift of b
- we define **functional margin** of (b, w) with respect to a training example (x_i, y_i) as the distance from the point (x_i, y_i) to the decision boundary, which is

$$\gamma_i = y_i \frac{(w^T x_i + b)}{\|w\|_2}$$

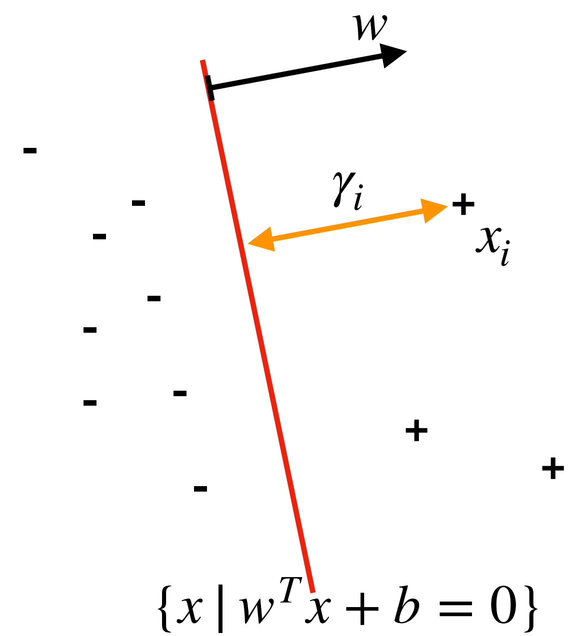
(The proof is on the next slide)



Geometric margin

- the distance γ_i from a hyperplane $\{x \mid w^T x + b = 0\}$ to a point x_i can be computed geometrically as follows
- We know that if you move from x_i in the negative direction of w by length γ_i, you arrive at the line, which can be written as

$$\left(x_i - \frac{w}{\|w\|_2} \gamma_i \right) \text{ is in } \{x \mid w^T x + b = 0\}$$



- so we can plug the point in the formula:

$$w^T \left(x_i - \frac{w}{\|w\|_2} \gamma_i \right) + b \;=\; 0$$

which is

$$w^T x_i - \frac{\|w\|_2^2}{\|w\|_2} \gamma_i + b \;=\; 0$$

and hence

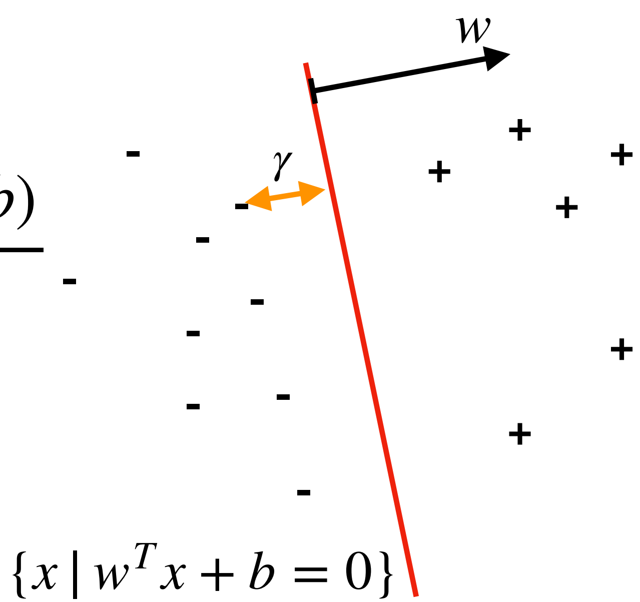
$$\gamma_i \;=\; \frac{w^T x_i + b}{\|w\|_2},$$

and we multiply it by y_i so that for negative samples we use the opposite direction of $-w$ instead of w

Geometric margin

- the **margin** with respect to a set is defined as

$$\gamma = \min_{i \in \{1,\ldots,n\}} \gamma_i = \min_i y_i \frac{(w^T x_i + b)}{\|w\|_2}$$



- among all linear classifiers, we would like to find one that has the **maximum margin**

- We will derive an algorithm that finds the maximum margin classifier, by transforming a difficult to solve optimization into an efficient one

Maximum margin classifier

(we transform the optimization into an efficient one)

- we propose the following optimization problem:

$$\text{maximize}_{w\in\mathbb{R}^d, b\in\mathbb{R}, \gamma\in\mathbb{R}} \quad \gamma$$

(maximize the margin)

$$\text{subject to} \quad \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq \gamma \quad \text{for all } i \in \{1,\ldots,n\}$$

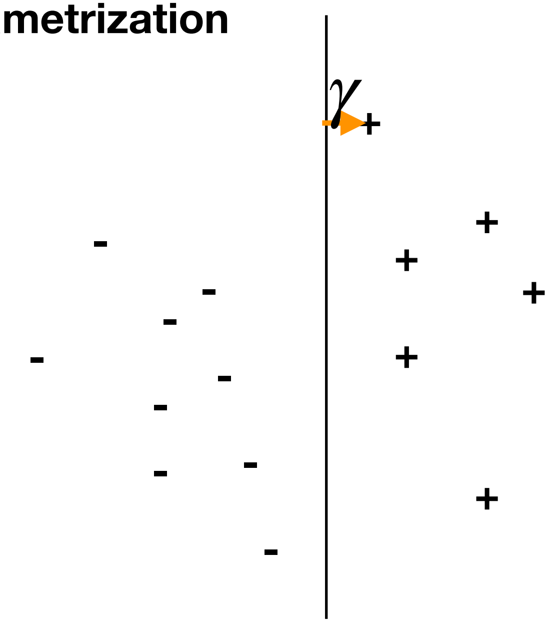
(s.t. γ is a lower bound on the margin)

- if we fix (w, b), the optimal solution of the optimization is the margin
- together with (w, b), this finds the classifier with the maximum margin
- note that this problem is **scale invariant** in (w, b), i.e. changing a (w, b) to $(2w, 2b)$ does not change either the feasibility or the objective value, hence the following reparametrization is valid
- the above optimization looks difficult, so we transform it using **reparametrization**

$$\text{maximize}_{w\in\mathbb{R}^d, b\in\mathbb{R}, \gamma\in\mathbb{R}} \quad \gamma$$

$$\text{subject to} \quad \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq \gamma \quad \text{for all } i \in \{1,\ldots,n\}$$

$$\|w\|_2 = \frac{1}{\gamma}$$



- Because of scale invariance, the optimal solution does not change, as the solutions to the original problem did not depend on $\|w\|_2$, and only depends on the direction of w

- maximize$_{w\in\mathbb{R}^d, b\in\mathbb{R}, \gamma\in\mathbb{R}}$ γ

  subject to $\dfrac{y_i(w^T x_i + b)}{\|w\|_2} \geq \gamma$ for all $i \in \{1,\ldots,n\}$

  $\|w\|_2 = \dfrac{1}{\gamma}$

- the above optimization still looks difficult, but can be transformed into

  maximize$_{w\in\mathbb{R}^d, b\in\mathbb{R}}$ $\dfrac{1}{\|w\|_2}$ **(maximize the margin)**

  subject to $\dfrac{y_i(w^T x_i + b)}{\|w\|_2} \geq \dfrac{1}{\|w\|_2}$ for all $i \in \{1,\ldots,n\}$ **(now $\dfrac{1}{\|w\|_2}$ plays the role of a lower bound on the margin)**

  which simplifies to

  minimize$_{w\in\mathbb{R}^d, b\in\mathbb{R}}$ $\|w\|_2^2$

  subject to $y_i(w^T x_i + b) \geq 1$ for all $i \in \{1,\ldots,n\}$

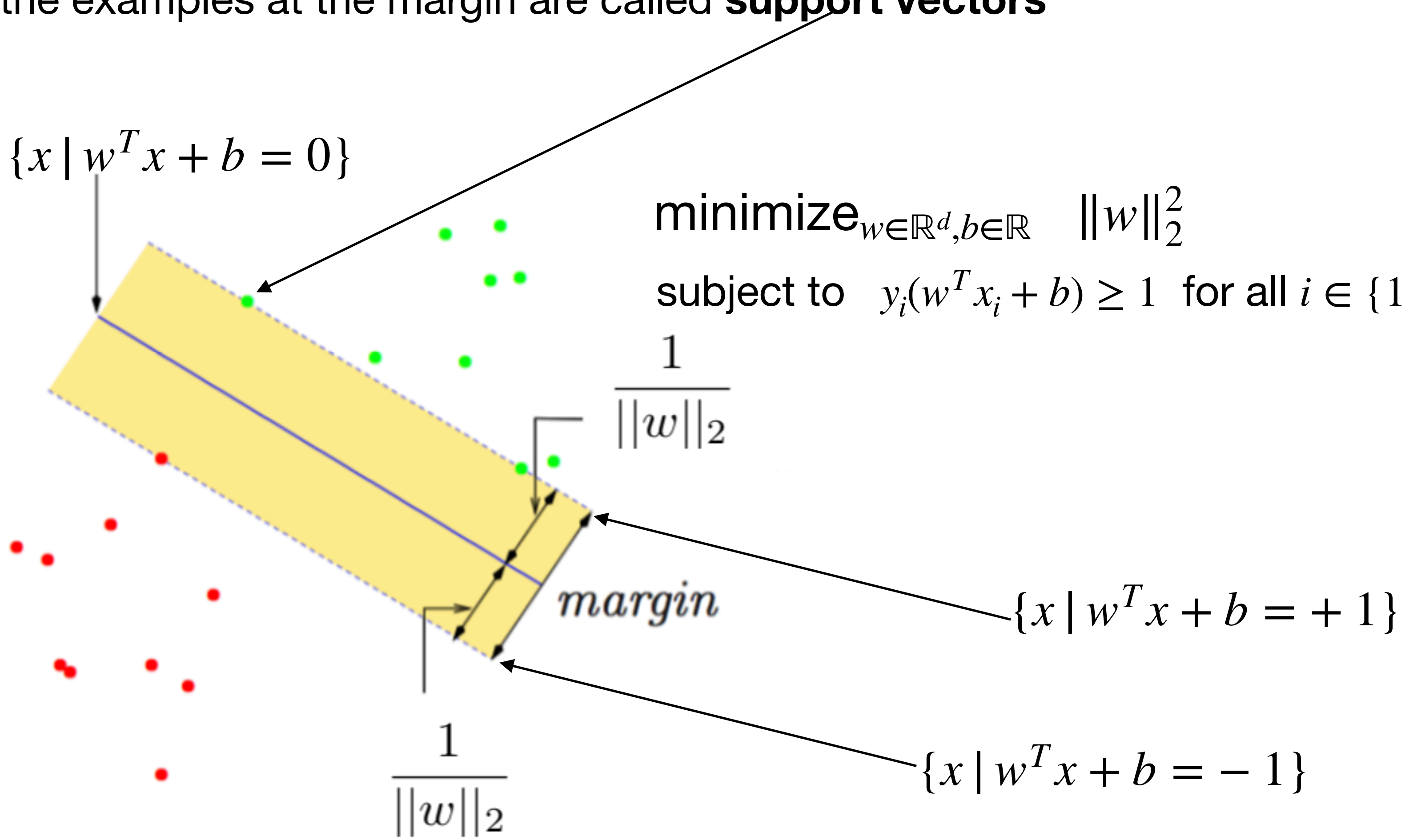
- this is a **quadratic program with linear constraints**, which can be easily solved
- once the optimal solution is found, the margin of that classifier (w, b) is $\dfrac{1}{\|w\|_2}$

What if the data is not separable?

- we cheated a little in the sense that the reparametrization of $\|w\|_2 = \frac{1}{\gamma}$ is possible only if the the margins are positive,
  i.e. the data is linearly separable with a positive margin
- otherwise, there is no feasible solution
- the examples at the margin are called **support vectors**

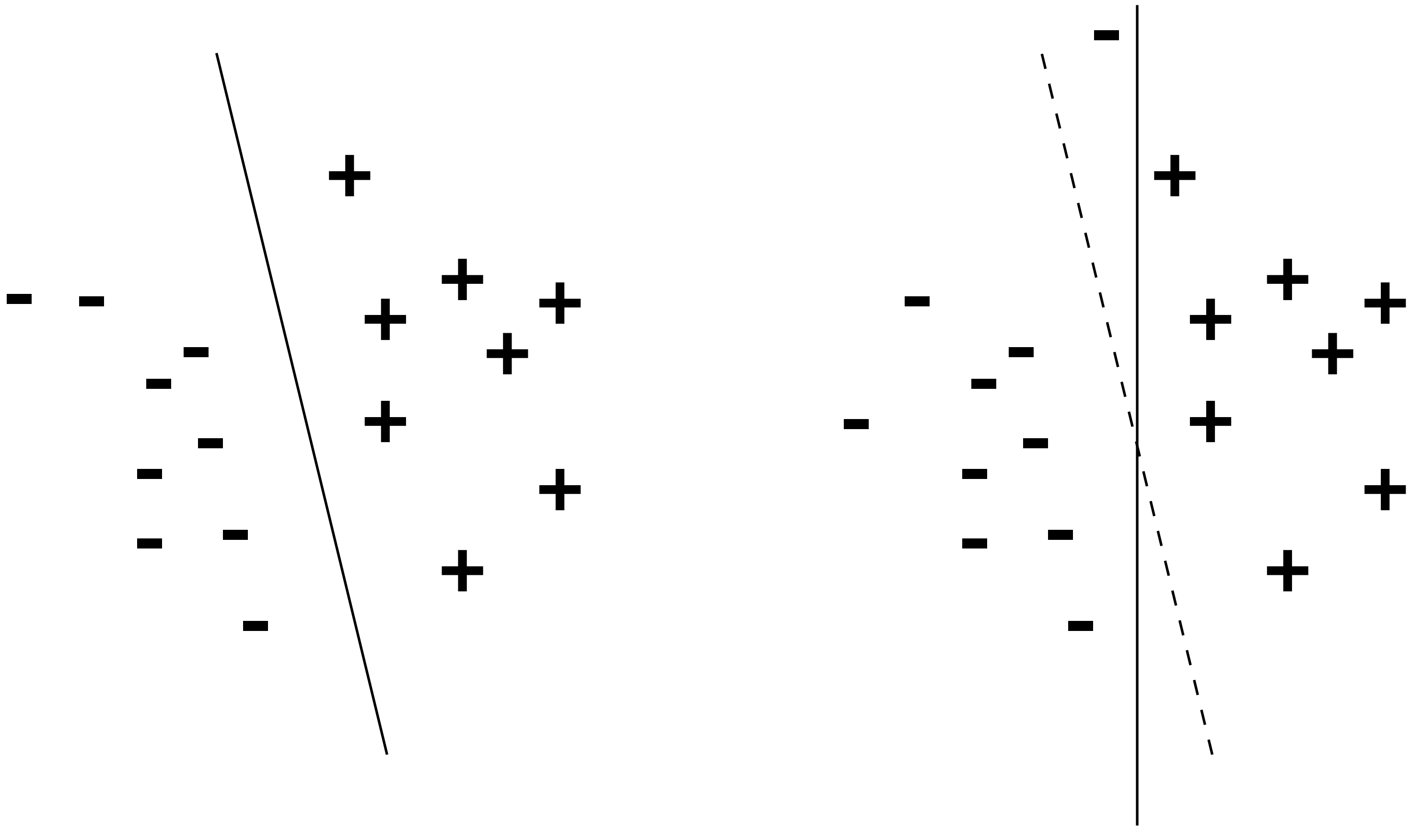
$$\text{minimize}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \|w\|_2^2$$

subject to $y_i(w^T x_i + b) \geq 1$ for all $i \in \{1, \ldots, n\}$

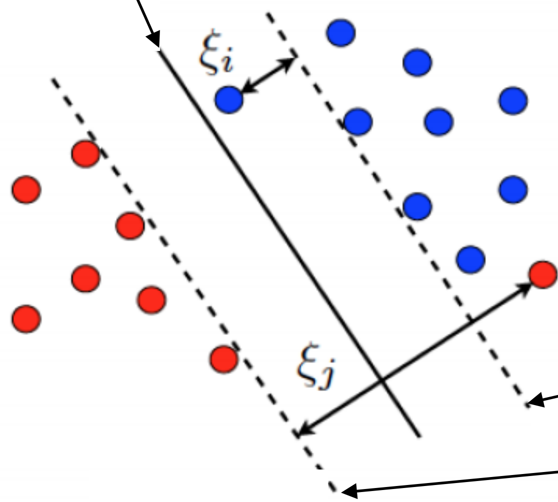


Two issues

- it does not generalize to non-separable datasets
- max-margin formulation we proposed is sensitive to outliers



What if the data is not separable?



- we introduce slack so that some points can violate the margin condition
$$y_i(w^T x_i + b) \;\geq\; 1 - \xi_i$$

- this gives a new optimization problem with some positive constant $c \in \mathbb{R}$

$$\text{minimize}_{w\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \quad \|w\|_2^2 + c\sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{for all } i \in \{1,\ldots,n\}$$

$$\xi_i \;\geq 0 \quad \text{for all } i \in \{1,\ldots,n\}$$

the (re-scaled) margin (for each sample) is allowed to be less than one,
but you pay $c\xi_i$ in the cost, and c balances the two goals:
maximizing the margin for most examples vs. having small number of violations

Support Vector Machine

- for the optimization problem

$$\text{minimize}_{w\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \quad \|w\|_2^2 + c \sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{for all } i \in \{1,\ldots,n\}$$

$$\xi_i \geq 0 \quad \text{for all } i \in \{1,\ldots,n\}$$

notice that at optimal solution, ξ_i's satisfy

- $\xi_i = 0$ if margin is big enough $y_i(w^T x_i + b) \geq 1$, or
- $\xi_i = 1 - y_i(w^T x_i + b)$, if the example is within the margin $y_i(w^T x_i + b) < 1$

- so one can write
    - $\xi_i = \max\{0, 1 - y_i(w^T x_i + b)\}$, which gives

$$\text{minimize}_{w\in\mathbb{R}^d, b\in\mathbb{R}} \quad \frac{1}{c}\|w\|_2^2 + \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i + b)\}$$

Sub-gradient descent for SVM

- SVM is the solution of

$$\text{minimize}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \frac{1}{c}\|w\|_2^2 + \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i + b)\}$$

- as it is non-differentiable, we solve it using sub-gradient descent
- which is exactly the same as gradient descent, except when we are at a non-differentiable point, we take one of the sub-gradients instead of the gradient (recall sub-gradient is a set)
- this means that we can take (a generic form derived from previous page)

$$\partial_w \ell(w^T x_i + b, y_i) \;=\; \mathbf{I}\{y_i(w^T x_i + b) \le 1\}(-y_i x_i)$$

and apply

$$w^{(t+1)} \;\leftarrow\; w^{(t)} - \eta \left(\sum_{i=1}^{n} \mathbf{I}\{y_i((w^{(t)})^T x_i + b^{(t)}) \le 1\}(-y_i x_i) \;+\; \frac{2}{c} w^{(t)} \right)$$

$$b^{(t+1)} \;\leftarrow\; b^{(t)} - \eta \sum_{i=1}^{n} \mathbf{I}\{y_i((w^{(t)})^T x_i + b^{(t)}) \le 1)\}(-y_i)$$