Linear classification

- **Loss function:**

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

- **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:**

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

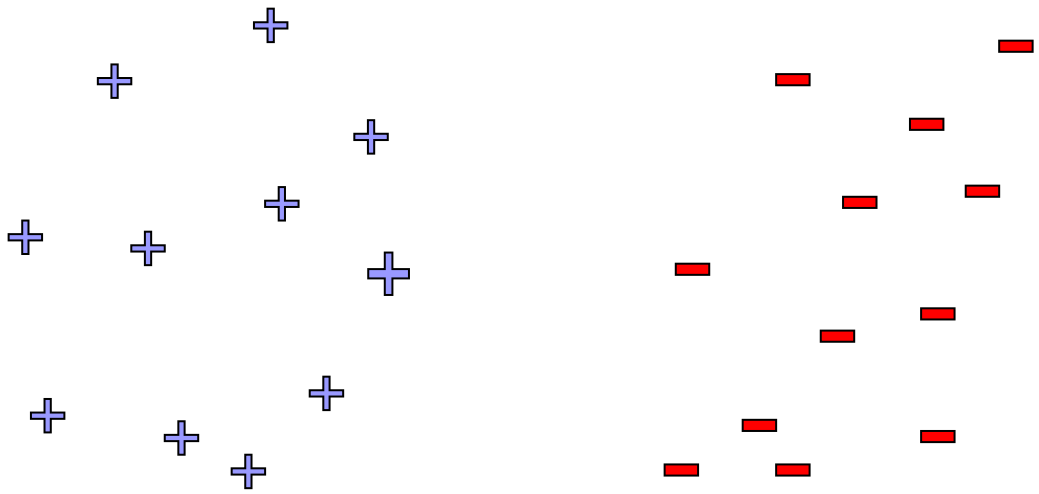
- **Model of logistic regression:**

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

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

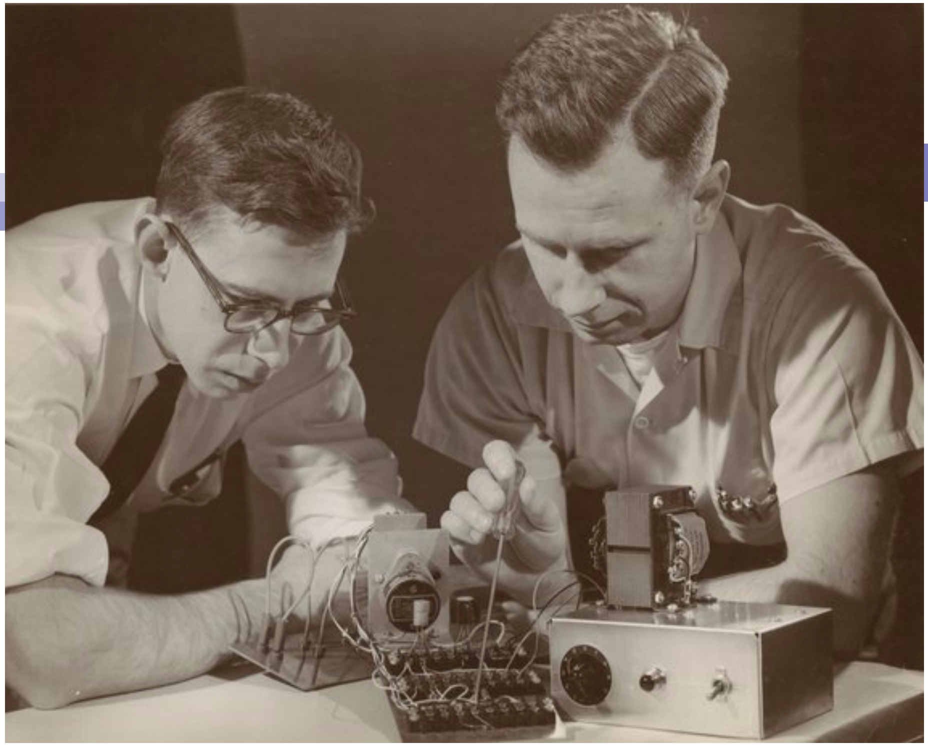
The Perceptron Algorithm [Rosenblatt '58, '62]

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

$$\text{sign}(w^T x_i + b)$$

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

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

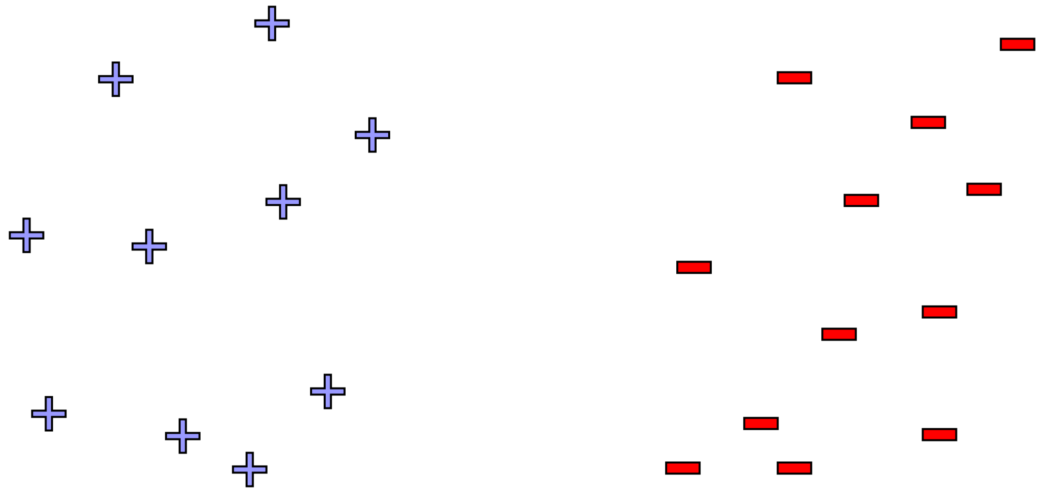


Rosenblatt 1957

"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



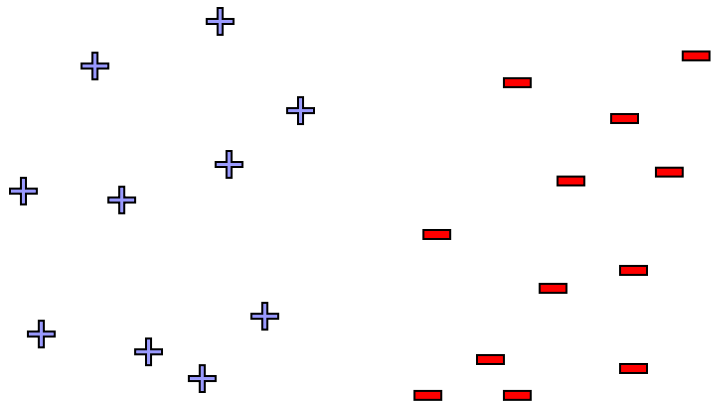
- Perceptron guaranteed to converge if
  - Data linearly separable:

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

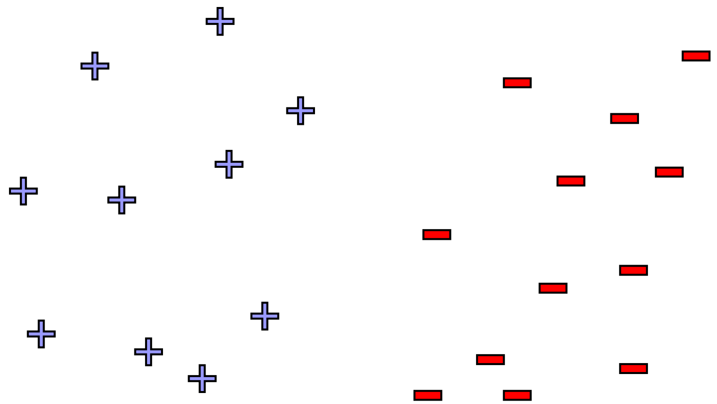
- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data



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????

Support Vector Machines

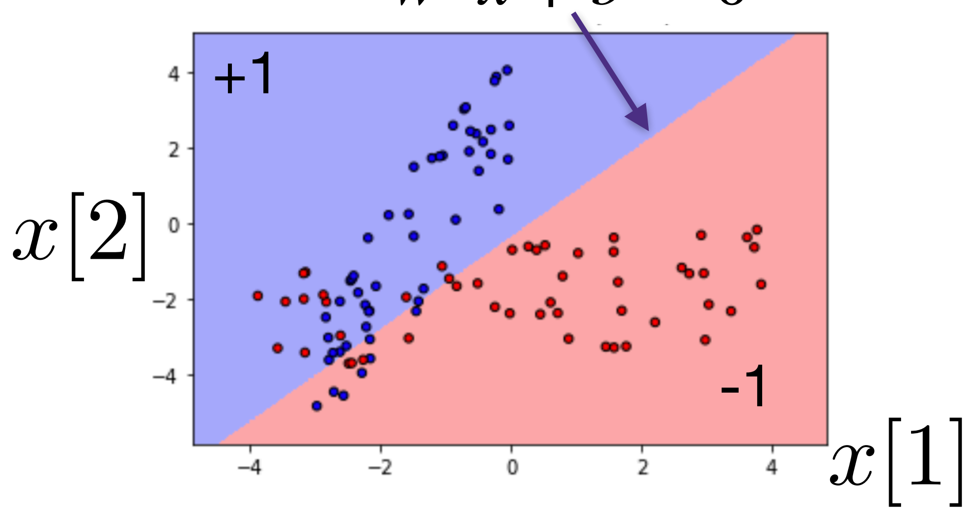
Logistic regression for binary classification

- Data $\mathcal{D} = \{(x_i \in \mathbb{R}^d, y_i \in \{-1, +1\})\}_{i=1}^n$
- Model: $\hat{y} = x^T w + b$
- Loss function: logistic loss $\ell(\hat{y}, y) = \log(1 + e^{-y\hat{y}})$
- Optimization: solve for

$$(\widehat{b}, \widehat{w}) = \arg\min_{b,w} \sum_{i=1}^{n} \log(1 + e^{-y_i(b + x_i^T w)})$$

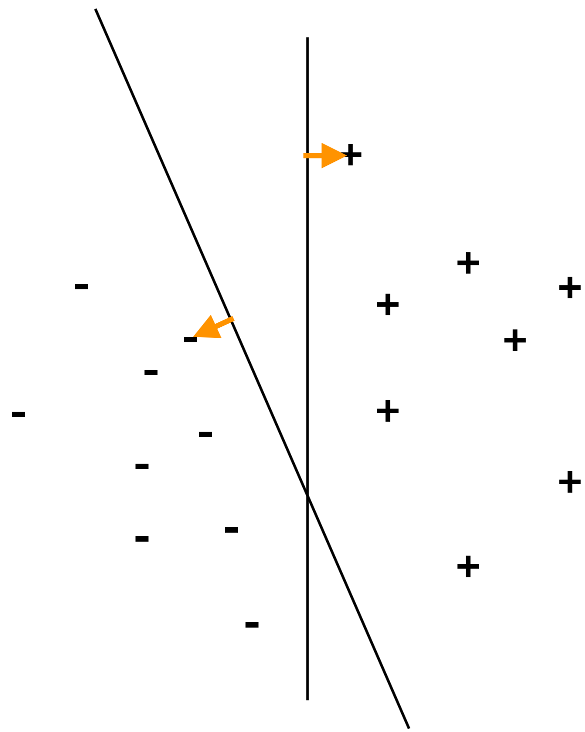
- As this is a **smooth convex** optimization, it can be solved efficiently using gradient descent
- Prediction: $\text{sign}(b + x^T w)$

decision boundary at $w^T x + b = 0$



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

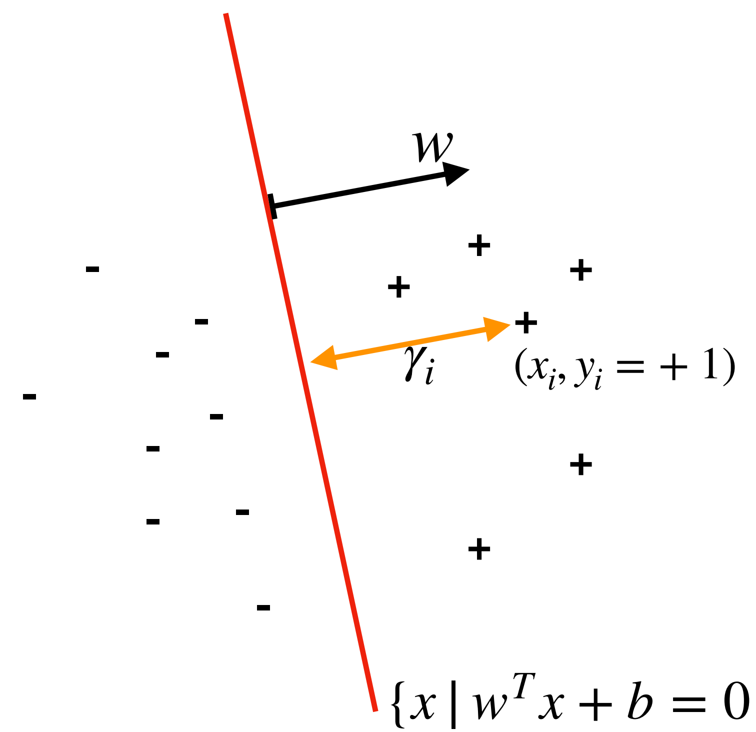


Geometric margin

- 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 hyperplane orthogonal to w with a shift of b
- we define **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

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

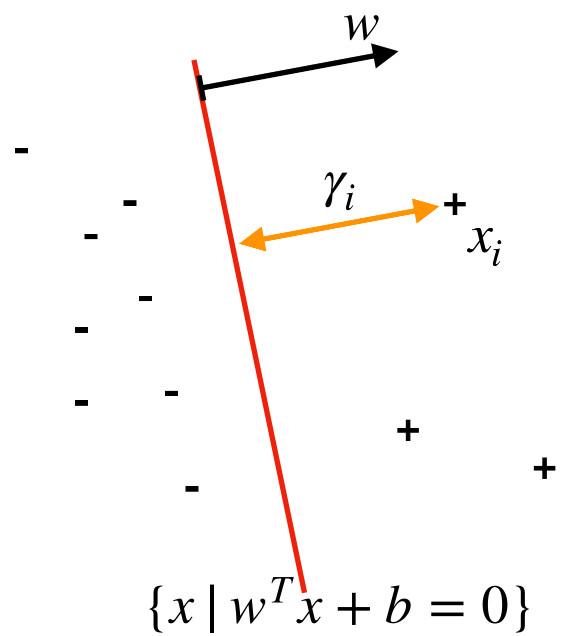
(The proof is on the next slide)



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

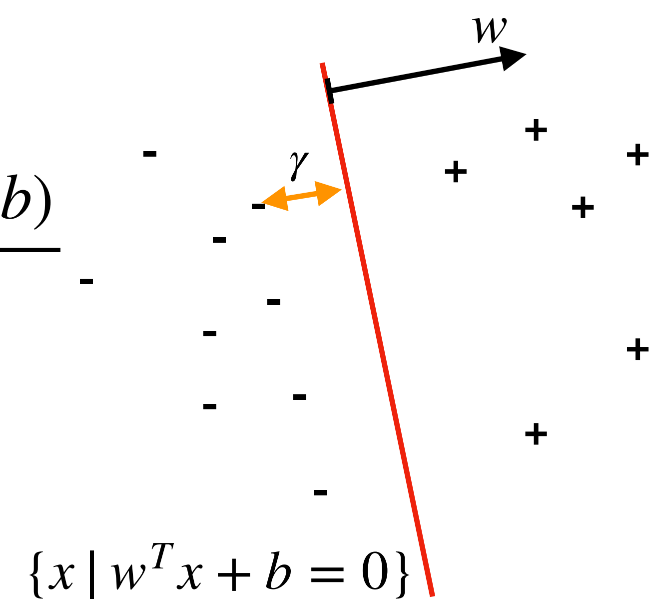
$$\gamma_i = \frac{w^T x_i + b}{\|w\|_2},$$

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

Maximum margin classifiers

- 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)

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

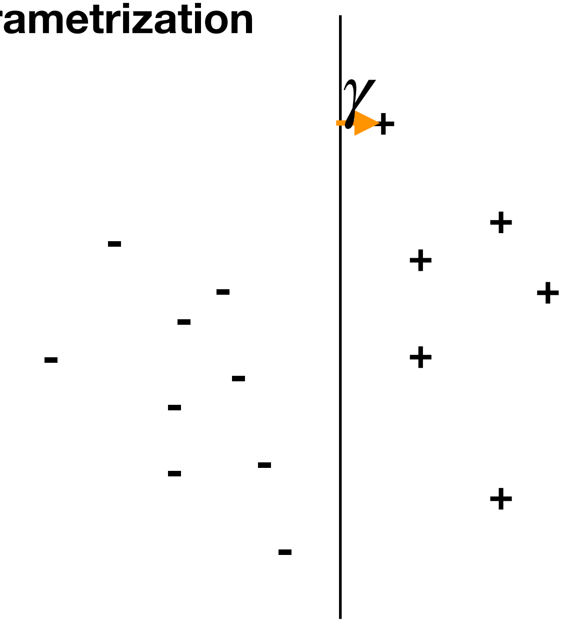
(s.t. γ is a lower bound on the margin)

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

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

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

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

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

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

  which simplifies to

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

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

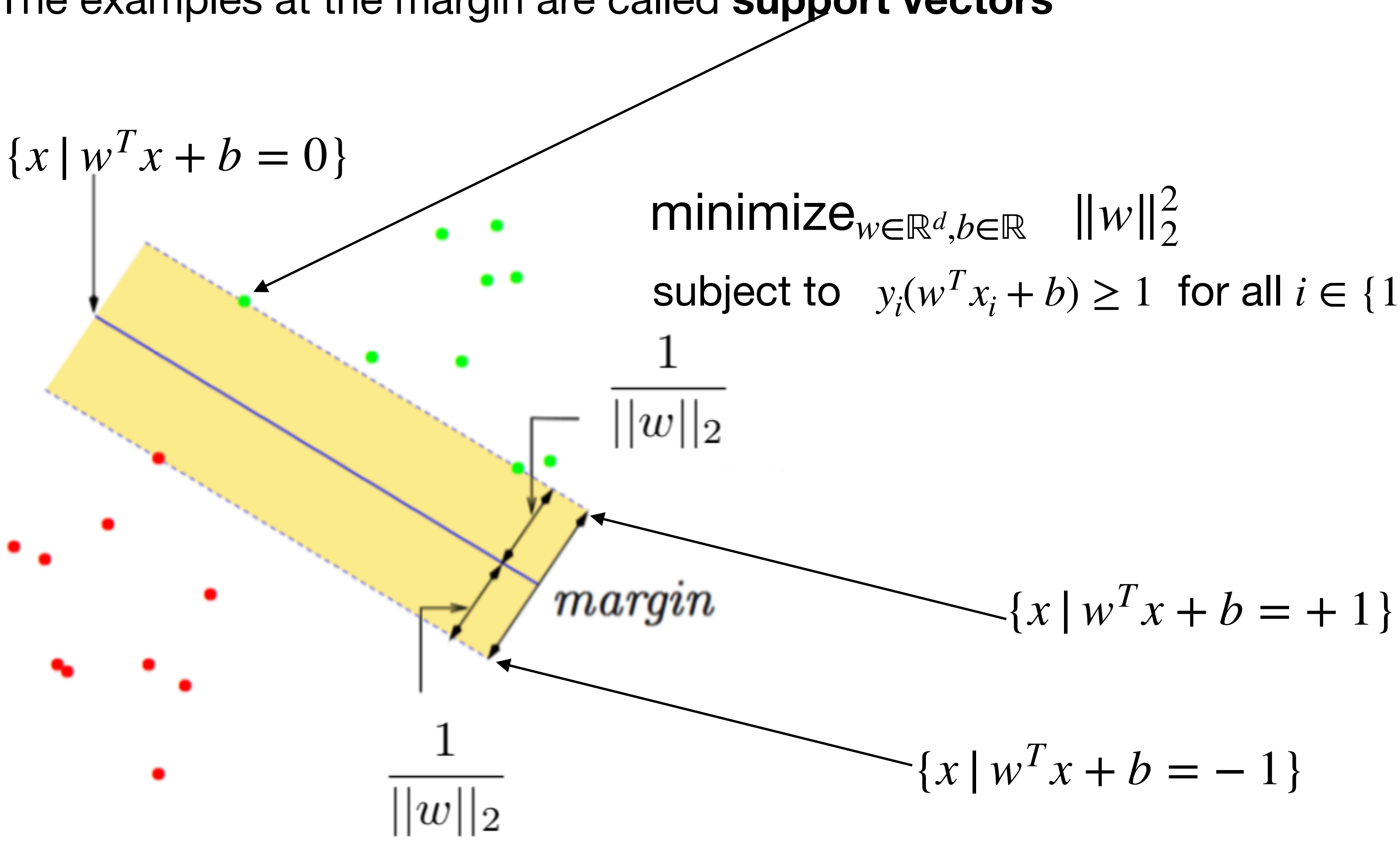
- This is a **quadratic program with linear constraints**, which can be easily solved
- Once the optimal solution is found, the margin of that classifier (w, b) is $\dfrac{1}{\|w\|_2}$

What if the data is not separable?

- We cheated a little in the sense that the reparametrization of $\|w\|_2 = \dfrac{1}{\gamma}$ 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**

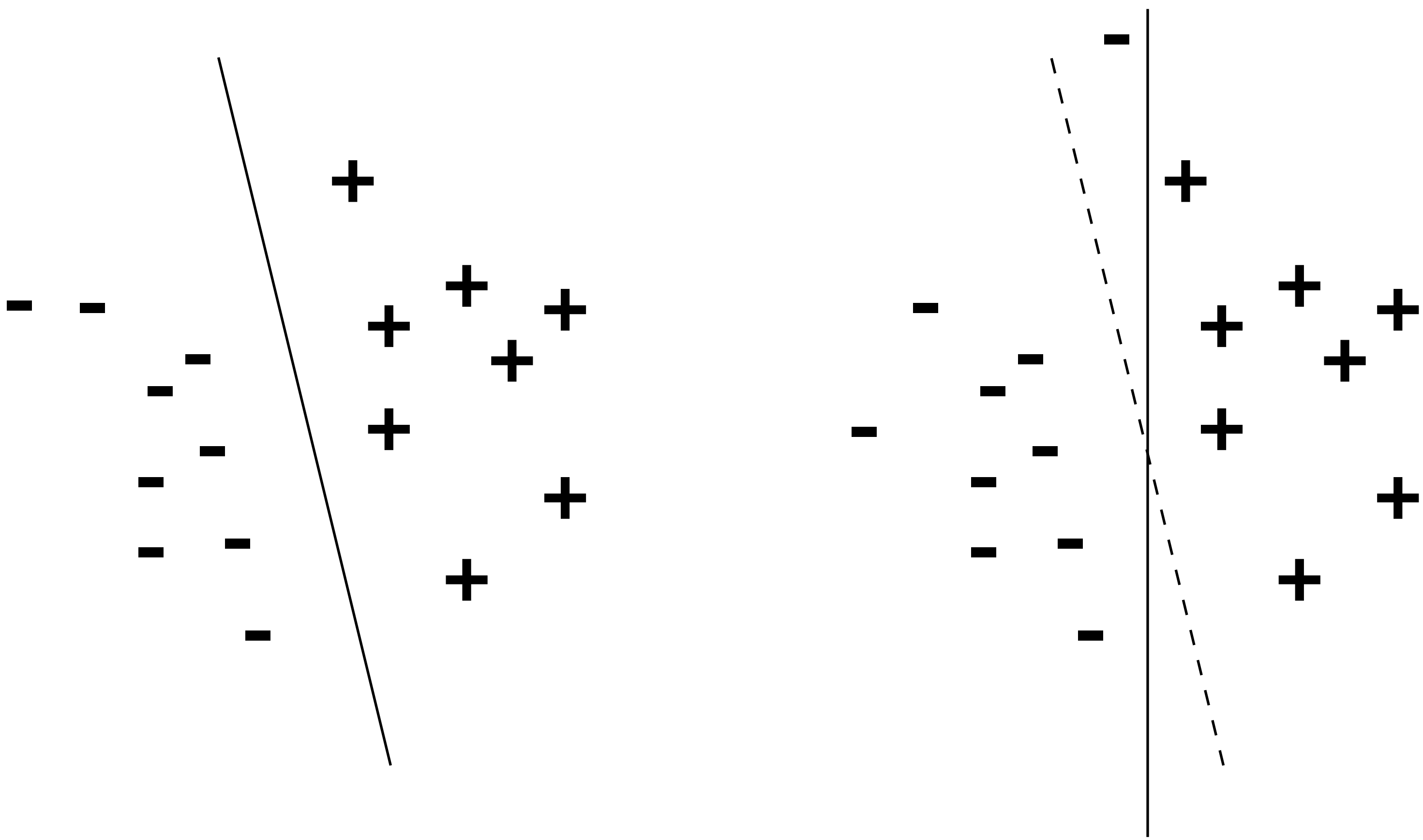
$$\text{minimize}_{w\in\mathbb{R}^d, b\in\mathbb{R}} \quad \|w\|_2^2$$

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

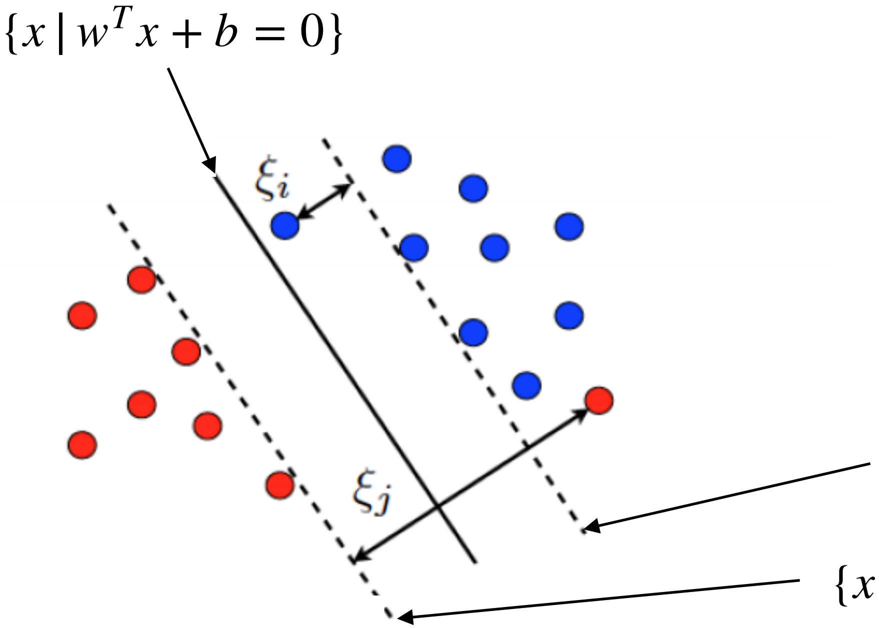


Two issues

- 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 $y_i(w^T x_i + b) < 1$

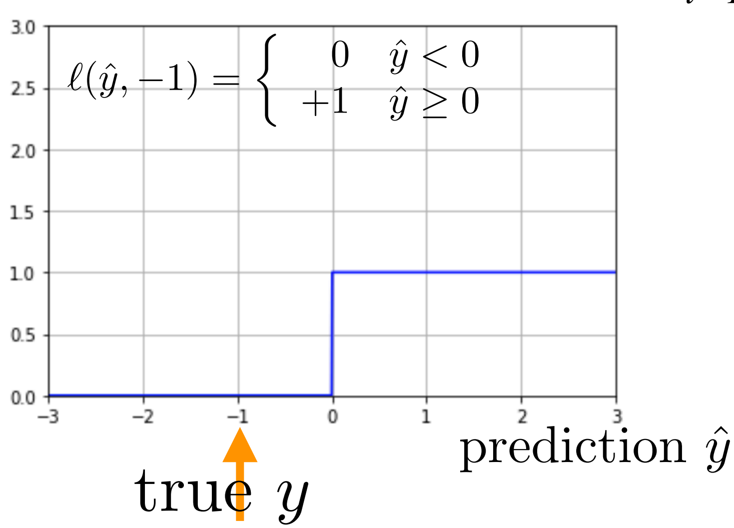
- So one can write
  - $\xi_i = \max\{0, 1 - y_i(w^T x_i + b)\}$, which gives

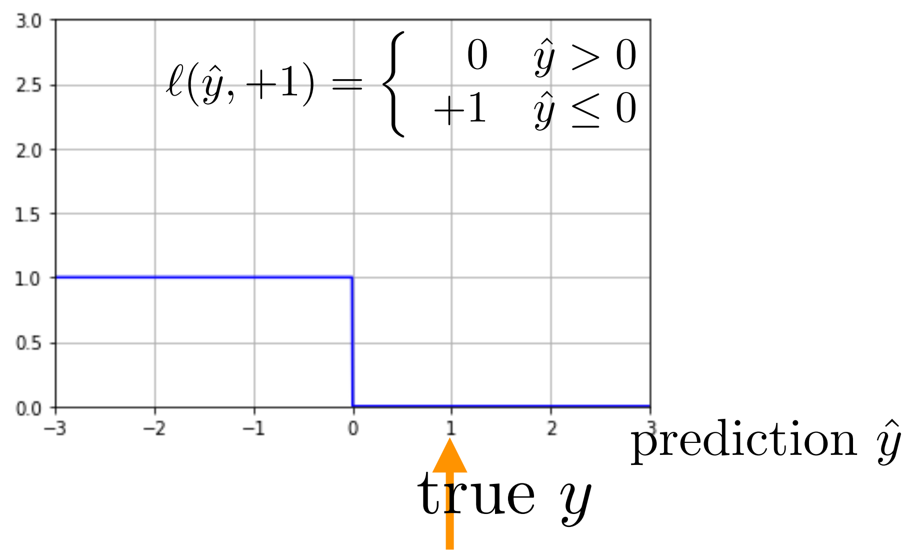
$$\text{minimize}_{w\in\mathbb{R}^d, b\in\mathbb{R}} \quad \frac{1}{c}\|w\|_2^2 + \sum_{i=1}^{n}\max\{0, 1 - y_i(w^T x_i + b)\}$$

Recall: we were looking for a loss function

- We want a loss function that
  - approximates (captures the flavor of) the 0-1 loss
  - can be easily optimized (e.g. convex and/or non-zero derivatives)
- More formally, we want a **loss function**
  - with $\ell(\hat{y}, -1)$ small when $\hat{y} < 0$ and larger when $\hat{y} > 0$
  - with $\ell(\hat{y}, 1)$ small when $\hat{y} > 0$ and larger when $\hat{y} < 0$
  - which has other nice characteristics, e.g., differentiable or convex
- We now have a new loss function from the SVM optimization problem:

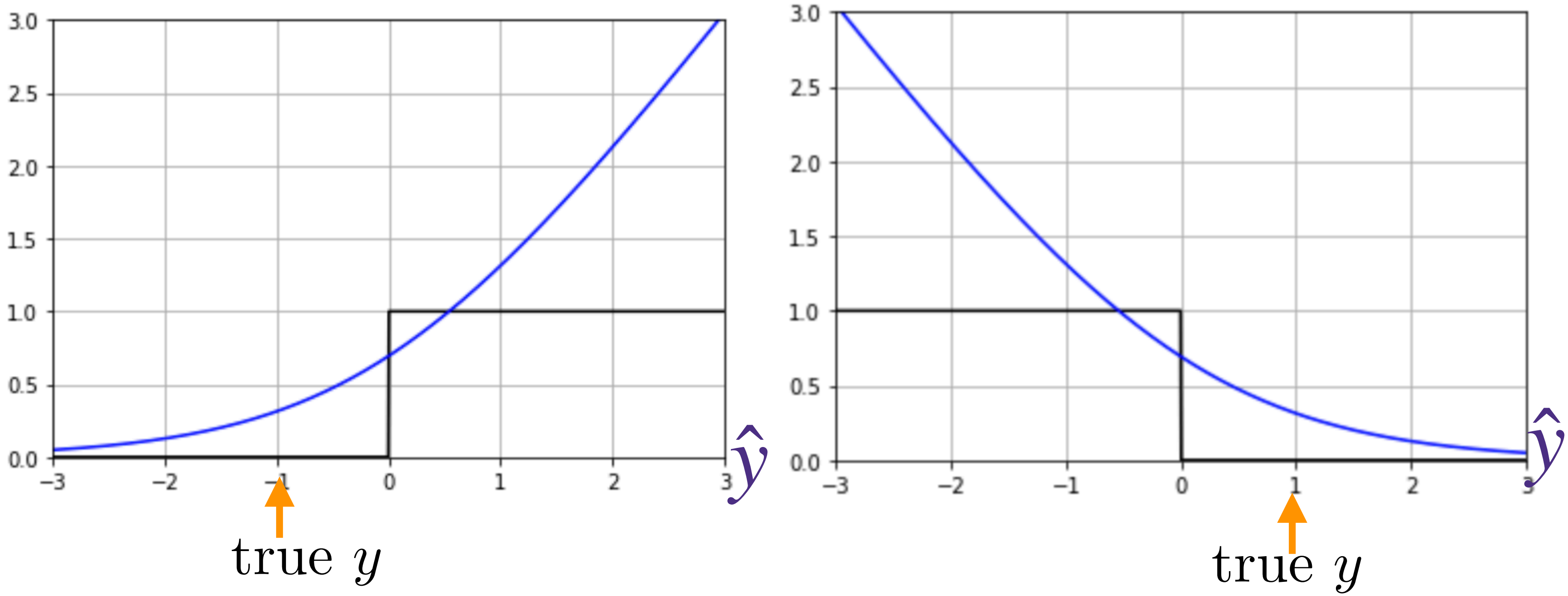
$$\text{minimize}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \frac{1}{c}\|w\|_2^2 + \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i + b)\}$$





Logistic loss $\ell(\hat{y}, y) = \log(1 + e^{-y\hat{y}})$

$\ell(\hat{y}, -1) = \log(1 + e^{\hat{y}})$ $\ell(\hat{y}, +1) = \log(1 + e^{-\hat{y}})$



- Differentiable and convex in $\hat{y}$
- Approximation of 0-1 loss
- Most popular choice of a loss function for classification problems

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

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

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