CSE417T – Lecture 19

- Please **mute** yourself and **turn off videos** to save bandwidth.

- If you have questions during the lecture
  - Use chatrooms to post your questions
    - I'll review chatrooms in batches
  - You can also un-mute yourself and ask the questions directly
- The slides are posted on the course website

- RECORD THE LECTURE!
  - Please remind me if I forget to do so.

Logistics: Homework

- Homework 4 will be due April 13 (Monday)
  - Please start it early
    - It was on average the most time consuming assignment for students in the past
  - Keep track of your own late days
    - Gradescope doesn’t allow separate deadlines
    - Your submissions won’t be graded if you exceed the late-day limit
- Homework 5 will have a tighter deadline
  - Tentative dates (still subject to change)
    - announce on April 7, due on April 19, 11AM

Recap

Nearest Neighbor

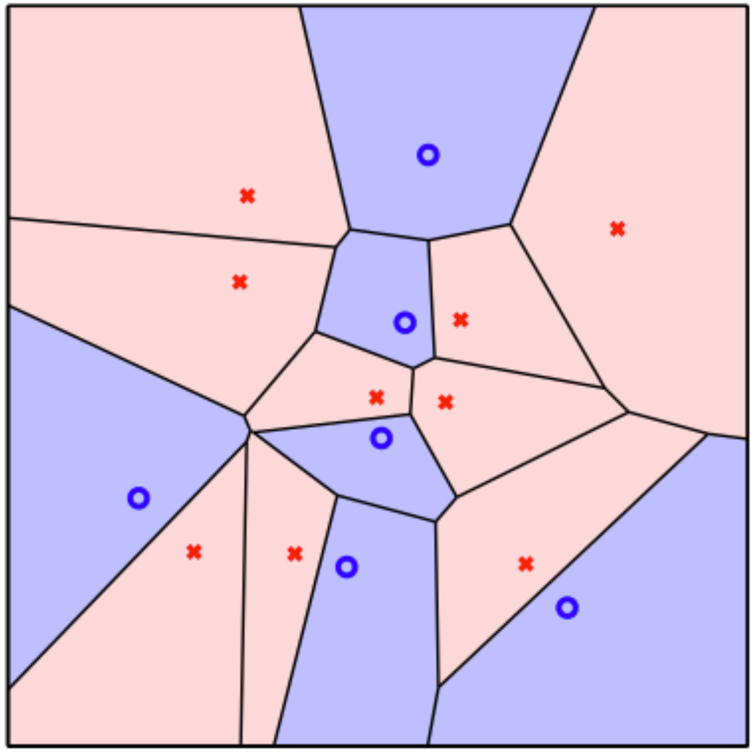
- Predict $\vec{x}$ according to its nearest neighbor
  - Given $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$
  - Let $\vec{x}_{[1]}$ be $\vec{x}$'s nearest neighbor, i.e., the closest point to $\vec{x}$ in D
  - Let $y_{[i]}(\vec{x})$ or $y_{[i]}$ be the label of $\vec{x}_{[i]}$

- Nearest neighbor hypothesis

$$g(\vec{x}) = y_{[1]}(\vec{x})$$

Nearest Neighbor

$g(\vec{x})$ looks like a Voronoi diagram



- Properties of Nearest Neighbor (NN)
  - No training is needed
  - Good interpretability
  - In-sample error $E_{in} = 0$
  - VC dimension is ∞
- This seems to imply bad learning models from what we talk about so far? Why we care?
- What we really care about is E_{out}
  - VC analysis: $E_{out} \leq E_{in} +$ Generalization error
    - We can infer E_{out} through E_{in} and model complexity
  - NN has nice guarantees outside of VC analysis

Nearest Neighbor is 2-Optimal

- Given mild conditions, for nearest neighbor, when $N \to \infty$, with high probability,

$$E_{out} \leq 2E_{out}^{*}$$

- That is, we can not infer E_{out} from E_{in}, but we know it cannot be much worse than the best anyone can do.

k-Nearest Neighbor (K-NN)

- Instead of ”single” nearest neighbor
  - Making predictions according to k nearest neighbors

- k-nearest neighbor (K-NN)
  - $g(\vec{x}) = sign\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$
  - (k is often odd for binary classification)

How to Choose k

- Making the choice of k a function of N, denoted by $k(N)$
  - Theorem:
    - If $k(N) \to \infty$ as $N \to \infty$ and $\frac{k(N)}{N} \to 0$ as $N \to \infty$
    - Then $E_{in}(g) \to E_{out}(g)$ and $E_{out}(g) \to E_{out}(g^*)$
  - Example: $k(N) = \sqrt{N}$ satisfies the condition
- Practical rule of thumb:
  - $k = 3$ is often a good enough choice
  - Using (cross-)validation to choose k

Summary of k-NN

- Pros
  - Simple algorithm
  - Good interpretations
  - Nice theoretical guarantee
  - Easy to adapt to regression (average of nearest neighbors) and multi-class classification (majority voting)

- Cons
  - Computational issue [See LFD 6.2.2 for more discussion]
  - Curse of dimensionality

- We have skipped some more discussion. See LFD Chap 6 if interested.

Lecture Notes Today

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

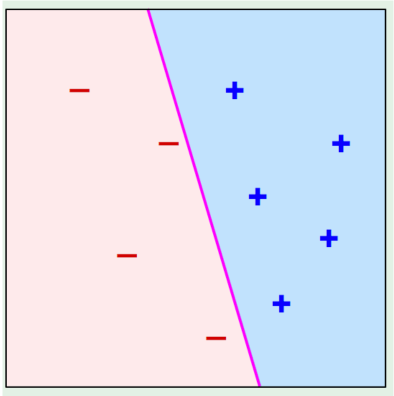
Let me know if you spot errors.

Support Vector Machines (SVM)

What Do We Know about Linear Classification?

- What we discussed so far:
  - PLA: Find a linear separator that separates the data within finite steps, if data is linear separable.
  - Pocket algorithm: empirically keep the best separator during PLA.
  - Surrogate loss: Using logistic regression for linear classification.

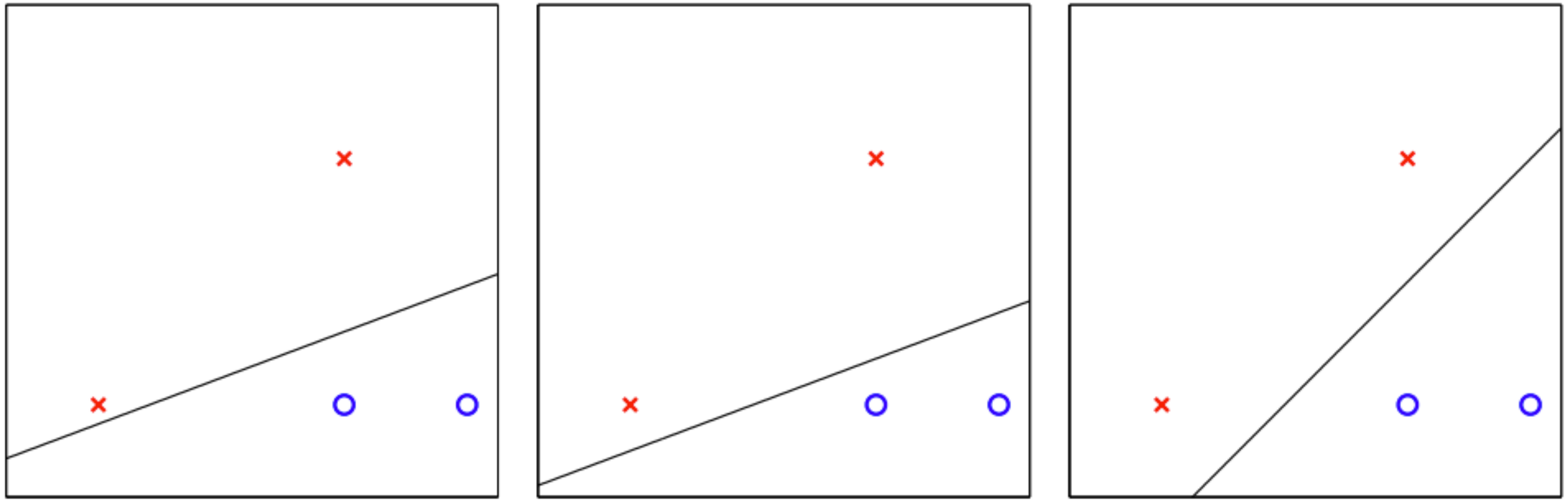
- Challenges
  - Binary classification error is hard to optimize,
  - We cannot use "gradient descent" type of algorithm minimize E_{in}.



- Support vector machines (SVM) tries to look at things a bit differently.

Linear Classification

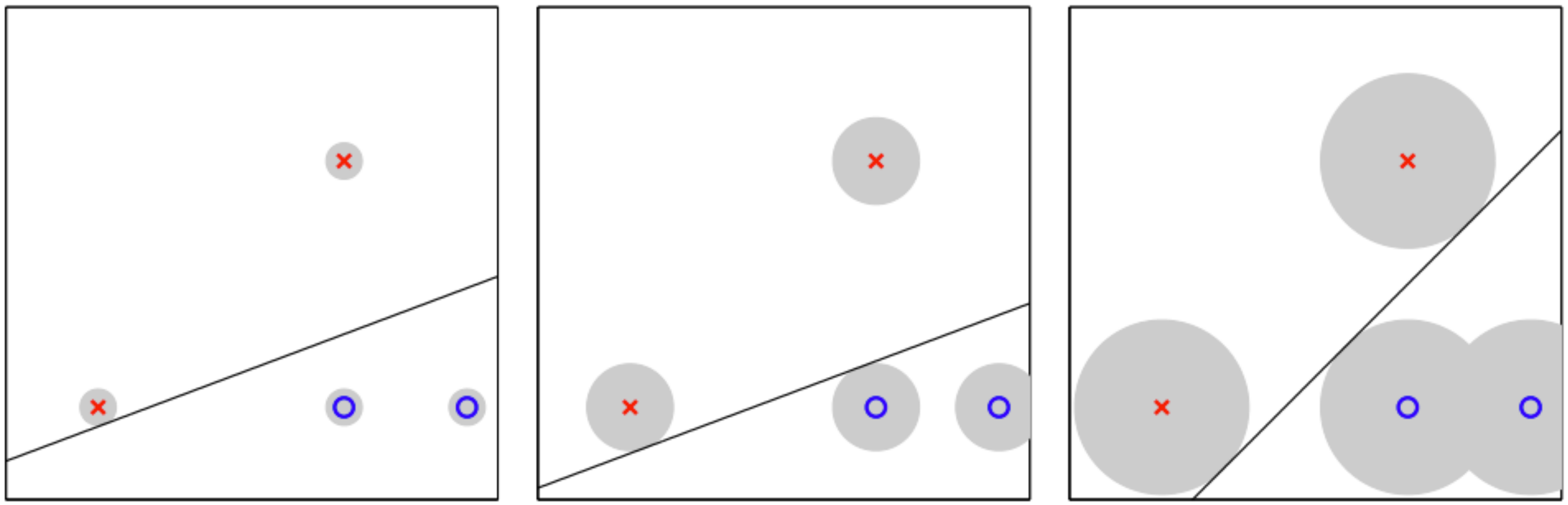
- Which separator would you choose?



Probably the right one.
Why?

Linear Classification

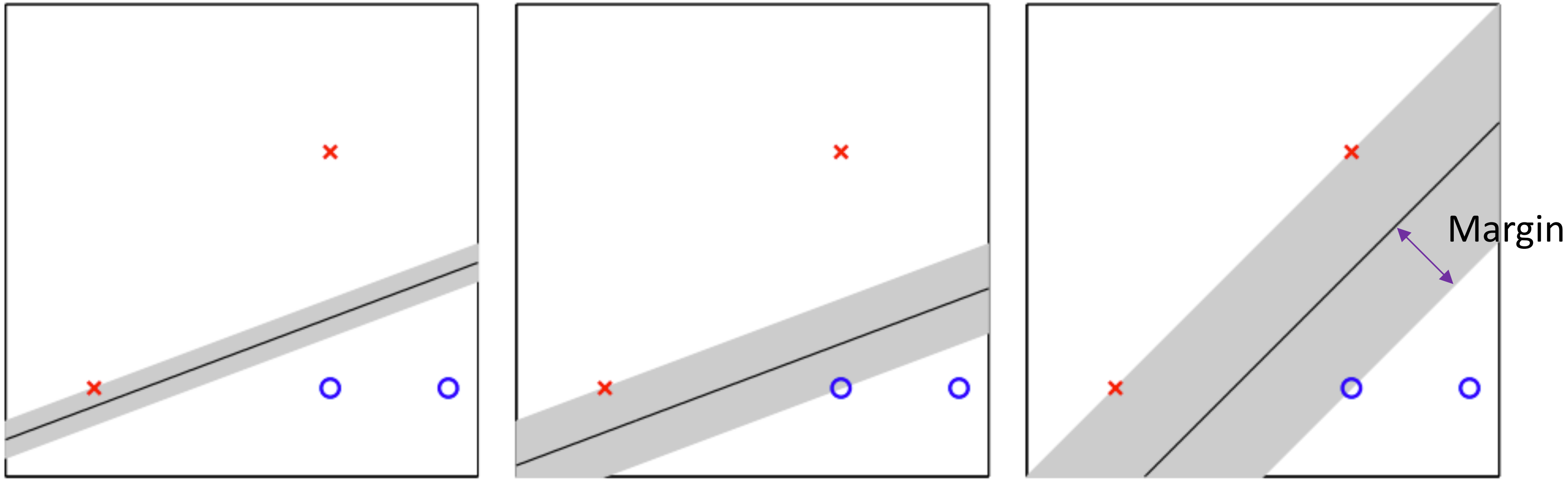
- Which separator would you choose?



More robust to noise (e.g., measurement error of $\vec{x}$)

Linear Classification

- Which separator would you choose?



Margin: shortest distance from the separator to the points in D

(Informal argument)

Higher margin => more “constrained” hypothesis => lower VC dimension

Support Vector Machine

- Goal:
  - Find the max-margin linear separator that separates the data
  - Recall the goal of PLA: Find the linear separator that separates the data

- Notations:

<u>Notations we used so far:</u>

- $\vec{x} = (x_0, x_1, \dots, x_d)$
- $\vec{w} = (w_0, w_1, \dots, w_d)$
- Linear separator

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

<u>Notations we will use in SVM</u>

- $\vec{x} = (x_1, \dots, x_d)$
- $\vec{w} = (w_1, \dots, w_d)$
- Linear separator

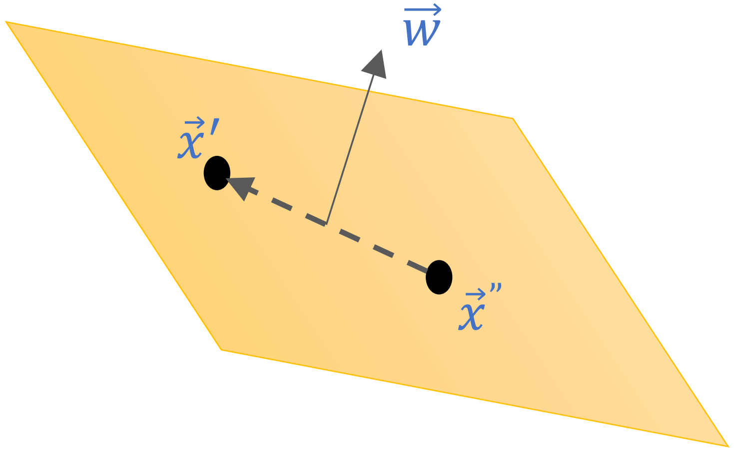
$$h(\vec{x}) = sign(\vec{w}^T \vec{x} + b)$$

Separating the bias/intercept b is important for us to characterize the margin.

We will use $(\vec{w}, b)$ to characterize the hypothesis

Relevant Review of Linear Algebra

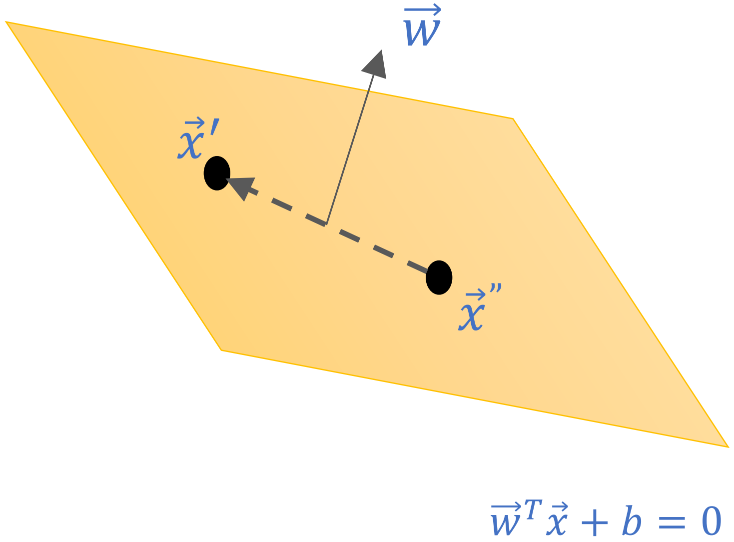
- Claim: $\vec{w}$ is the norm vector of the hyperplane $\vec{w}^T\vec{x} + b = 0$



$\vec{w}^T\vec{x} + b = 0$

Relevant Review of Linear Algebra

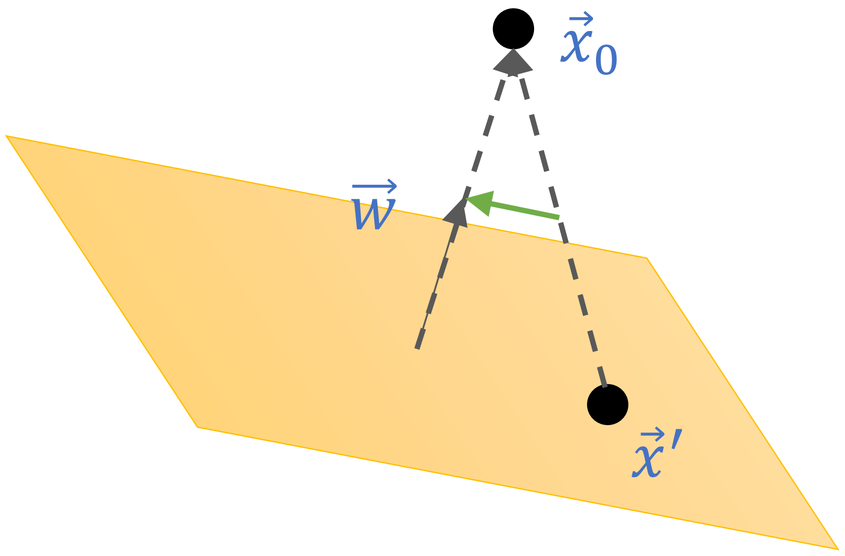
- Claim: $\vec{w}$ is the norm vector of the hyperplane $\vec{w}^T\vec{x} + b = 0$



- Consider any two points $\vec{x}'$ and $\vec{x}''$ on the hyperplane
  - $\vec{w}^T\vec{x}' + b = 0$
  - $\vec{w}^T\vec{x}'' + b = 0$
- Combining the above
  - $\vec{w}^T(\vec{x}' - \vec{x}'') = 0$
- $\vec{w}$ is orthogonal to the hyperplane
- $\vec{w}$ is the norm vector of the hyperplane

Relevant Review of Linear Algebra

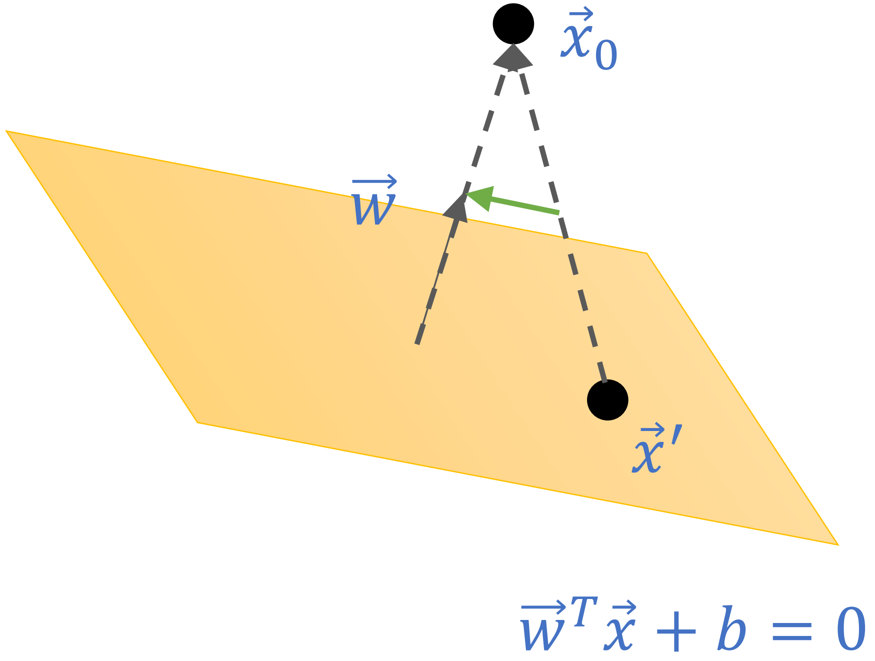
- What is the distance between a point $\vec{x}_0$ and a hyperplane $\vec{w}^T\vec{x} + b = 0$



$\vec{w}^T\vec{x} + b = 0$

Relevant Review of Linear Algebra

- What is the distance between a point $\vec{x}_0$ and a hyperplane $\vec{w}^T\vec{x} + b = 0$



- Consider an arbitrary point $\vec{x}'$ on the hyperplane

- Distance between the point $\vec{x}$ and the hyperplane

$$dist(\vec{x}_0, \vec{w}, b) = \left|\frac{\vec{w}^T}{\|\vec{w}\|}(\vec{x}_0 - \vec{x}')\right|$$
$$\left|\frac{1}{\|\vec{w}\|}(\vec{w}^T\vec{x}_0 - \vec{w}^T\vec{x}')\right|$$
$$\left|\frac{1}{\|\vec{w}\|}(\vec{w}^T\vec{x}_0 + b)\right|$$

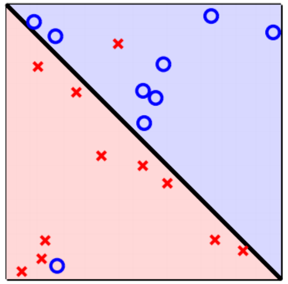
Outline of Our Discussion

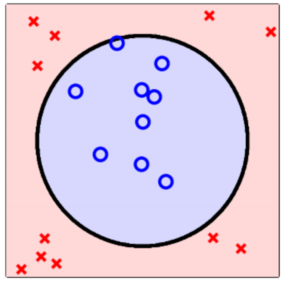
- Assume data is linearly separable
  - Formulate the hard-margin SVM

Given D, find separator $(\vec{w}, b)$ that

maximize margin$(\vec{w}, b)$
s.t. all points in D is correctly classified

- When data is not linearly separable
  - Tolerate some noise
    - Soft-margin SVM
  - Nonlinear transform
    - Dual formulation and kernel tricks





Hard-Margin SVM

Hard-Margin SVM

- Goal
  - Given $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$ that is linearly separable
  - Find separator $(\vec{w}, b)$ that (1) maximizes the margin and (2) separates D

- $(\vec{w}, b)$ separates D (making correct predictions for all points in D)
  - $y_n = sign(\vec{w}^T \vec{x}_n + b)$ for all n
  - $y_n(\vec{w}^T \vec{x}_n + b) \geq 0$ for all n

- Margin: shortest distance from the separator to points in D

$$\begin{aligned} \text{margin}(\vec{w}, b) &= min_n\, dist(\vec{x}_n, \vec{w}, b) \\ &= min_n \left| \frac{1}{\|\vec{w}\|} (\vec{w}^T \vec{x}_n + b) \right| \\ &= min_n \frac{1}{\|\vec{w}\|} y_n (\vec{w}^T \vec{x}_n + b) \end{aligned}$$

$dist(\vec{x}_0, \vec{w}, b) = \left| \frac{1}{\|\vec{w}\|} (\vec{w}^T \vec{x}_0 - b) \right|$

$y_n \in \{-1, +1\}$ and
$y_n(\vec{w}^T \vec{x}_n + b) \geq 0$

Hard-Margin SVM

- Goal
  - Given $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$ that is linearly separable
  - Find separator $(\vec{w}, b)$ that (1) maximizes the margin and (2) separates D

- Formulate it as a constrained optimization problem

$$\begin{aligned}
&\text{maximize}_{\vec{w},b} && \text{margin}(\vec{w}, b) \\
&\text{subject to} && y_n(\vec{w}^T\vec{x}_n + b) \geq 0, \forall n \\
& && \text{margin}(\vec{w}, b) = min_n \frac{1}{\|\vec{w}\|} y_n(\vec{w}^T\vec{x}_n + b)
\end{aligned}$$

Hard-Margin SVM

- The constrained optimization problem

$$\begin{aligned} &\text{maximize}_{\vec{w},b} && \text{margin}(\vec{w}, b) \\ &\text{subject to} && y_n(\vec{w}^T\vec{x}_n + b) \geq 0, \forall n \\ & && \text{margin}(\vec{w}, b) = min_n \frac{1}{||\vec{w}||} y_n(\vec{w}^T\vec{x}_n + b) \end{aligned}$$

- normalizing $(\vec{w}, b)$
  - Note that $\vec{w}^T\vec{x} + b = 0$ is equivalent to $c\vec{w}^T\vec{x} + cb = 0$ for any c
  - We will normalize $(\vec{w}, b)$ such that $min_n\, y_n(\vec{w}^T\vec{x}_n + b) = 1$
    - $y_n(\vec{w}^T\vec{x}_n + b) \geq 1, \forall n$
    - $\text{margin}(\vec{w}, b) = \frac{1}{||\vec{w}||}$

Hard-Margin SVM

- The constrained optimization problem

$$\begin{aligned}&\text{maximize}_{\vec{w},b} \quad \frac{1}{||\vec{w}||}\\ &\text{subject to} \quad y_n(\vec{w}^T\vec{x}_n + b) \geq 1, \forall n\end{aligned}$$

- Some final adjustments

$$\begin{aligned}&\text{minimize}_{\vec{w},b} \quad \frac{1}{2}\vec{w}^T\vec{w}\\ &\text{subject to} \quad y_n(\vec{w}^T\vec{x}_n + b) \geq 1, \forall n\end{aligned}$$

Final Form of Hard-Margin SVM

$$\begin{aligned} &\text{minimize}_{\vec{w},b} && \frac{1}{2}\vec{w}^T\vec{w} \\ &\text{subject to} && y_n(\vec{w}^T\vec{x}_n + b) \geq 1, \forall n \end{aligned}$$

- How to solve it?
  - Hard-margin SVM is a Quadratic Program
  - Standard form of Quadratic Program (QP)

$$\begin{aligned} &\text{minimize}_{\vec{u}} && \frac{1}{2}\vec{u}^T Q\vec{u} + \vec{p}^T\vec{u} \\ &\text{subject to} && A\vec{u} \geq \vec{c} \end{aligned}$$

  - There exist efficient QP solvers we can utilize

Short Break and Questions

Connection to Regularization

- Weight decay regularization in linear model

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T\vec{w} \le C$

- Another way to look at SVM

minimize $\vec{w}^T\vec{w}$

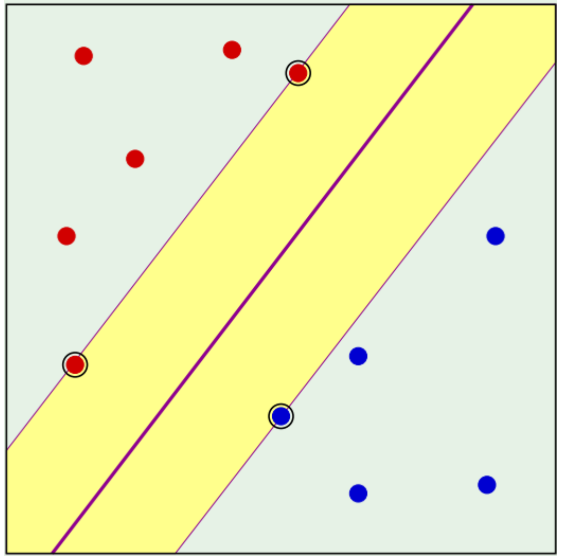
subject to $E_{in}(\vec{w}) = 0$

Support Vectors

- We call the points closest to the separator (candidate) support vectors
  - Since they support the separator

- What are the math properties of support vectors?
  - They are the points that the equality holds in the constraints
    - If $\vec{x}_n$ is a support vector, $y_n(\vec{w}^T \vec{x}_n + b) = 1$

$$\begin{array}{ll} \text{minimize}_{\vec{w},b} & \frac{1}{2}\vec{w}^T\vec{w} \\ \text{subject to} & y_n(\vec{w}^T\vec{x}_n + b) \geq 1, \forall n \end{array}$$

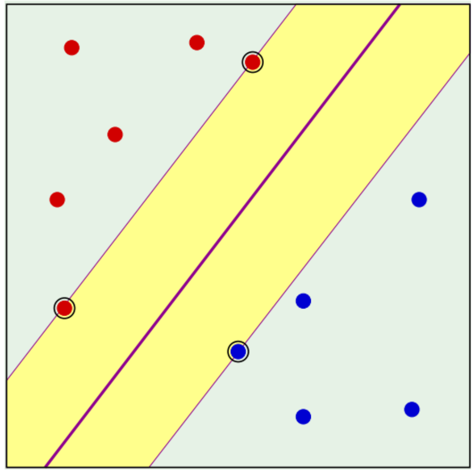


  - Removing the non-support vectors will not impact the linear separator

Leave-One-Out Cross Validation (LOOCV)

- Two things we know so far
  - Removing non-support vectors will not impact the separator
  - LOOCV error (when not used for model selection) is an unbiased estimate of $E_{out}(N-1)$ (E_{out} when trained on $N-1$ points)

- What's the LOOCV error for SVM?
  - $E_{LOOCV} \leq \dfrac{\#\,\text{support vectors}}{N}$

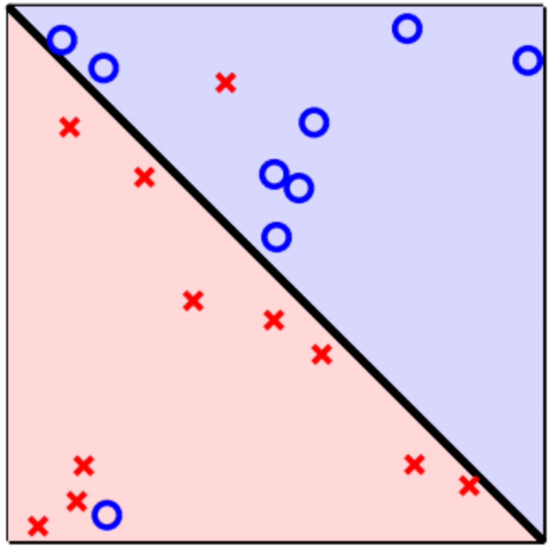


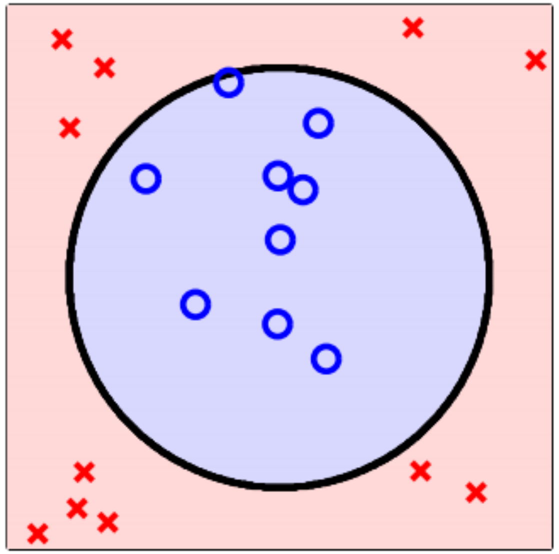
- Note that we know # support vectors after training
  - Count # points that satisfy $y_n(\vec{w}^T \vec{x}_n + b) = 1$
- Another method to estimate/bound E_{out} (counting # support vectors)

What if Data is Not Linearly Separable

Non-Separable Data

- Two scenarios





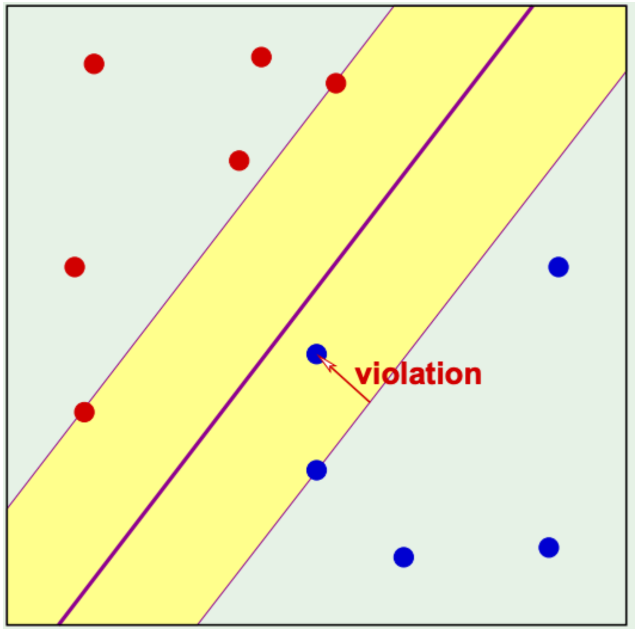
- Tolerate some noise
  - Soft-Margin SVM
- Nonlinear transform
  - Dual formulation and kernel tricks

Soft-Margin SVM

- Intuition: We want to tolerate small noises when maintaining large margin

- For each point $(\vec{x}_n, y_n)$, we allow a deviation $\xi_n \geq 0$
  - Instead of requiring $y_n(\vec{w}^T \vec{x}_n + b) \geq 1$
  - The constraint becomes

$$y_n(\vec{w}^T \vec{x}_n + b) \geq 1 - \xi_n$$

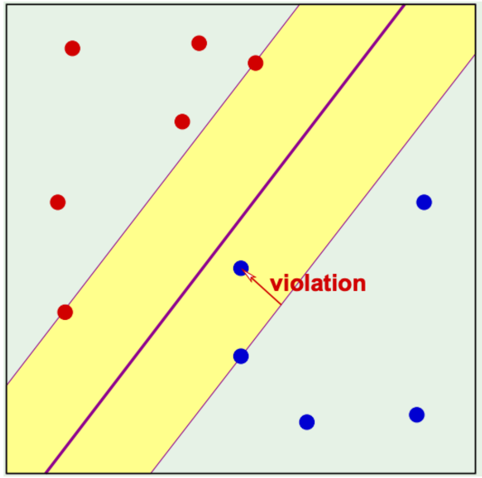


- We add a penalty for each deviation
  - Total penalty $C \sum_{n=1}^{N} \xi_n$

Soft-Margin SVM

- The constraint becomes: $y_n(\vec{w}^T\vec{x}_n + b) \geq 1 - \xi_n$
- We add a penalty for each deviation: Total penalty $C\sum_{n=1}^{N}\xi_n$

$$\begin{aligned}&\text{minimize}_{\vec{w},b,\vec{\xi}} && \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{n=1}^{N}\xi_n \\ &\text{subject to} && y_n(\vec{w}^T\vec{x}_n + b) \geq 1 - \xi_n, \forall n \\ & && \xi_n \geq 0, \forall n\end{aligned}$$

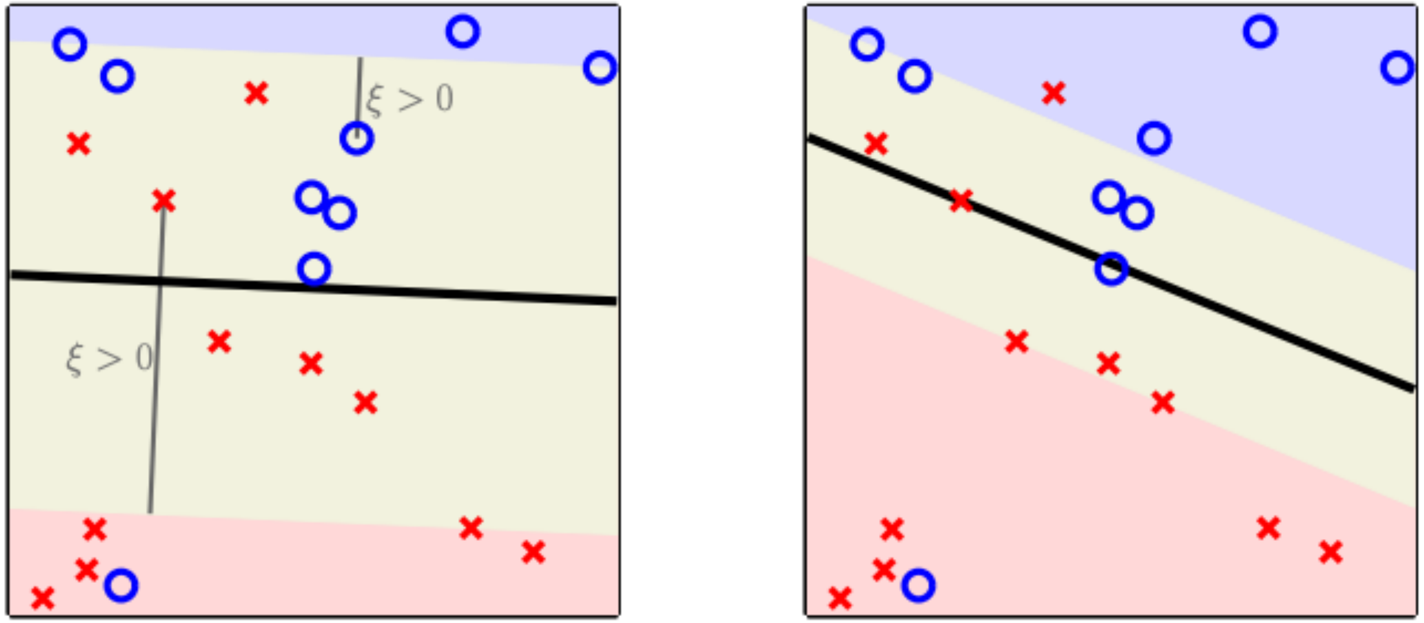


Remarks:

- C is a hyper-parameter we can choose, e.g., using validation
- Soft-margin SVM is still a Quadratic Program, with efficient solvers

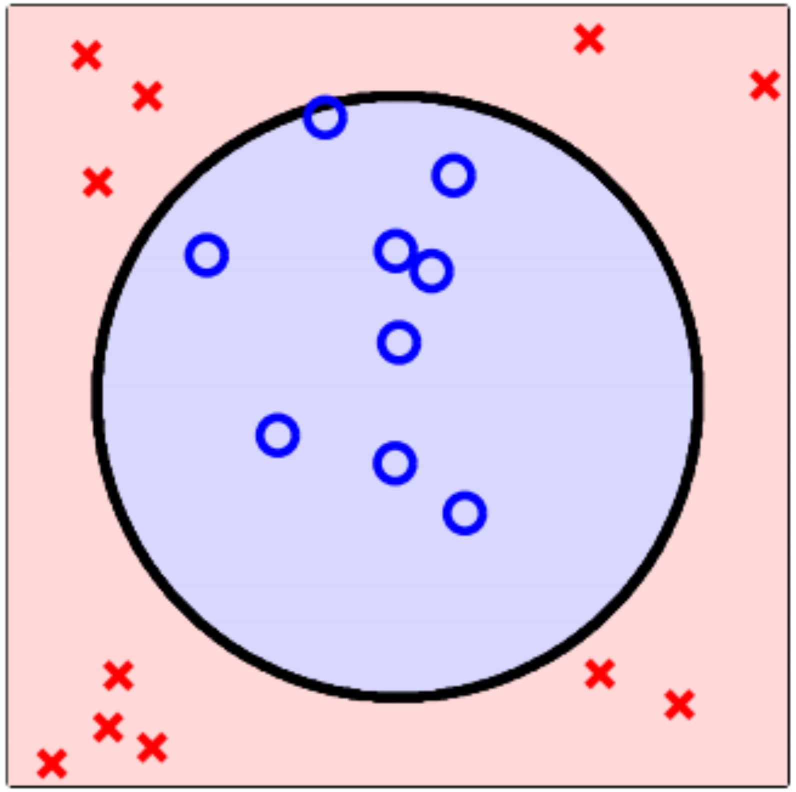
Impacts of C in Soft-Margin SVM

- Large C => less tolerate to noise, having smaller margin



$C = 1$ $C = 500$

What if Tolerating Small Noises Is Not Enough



Nonlinear transform

We can apply standard nonlinear transformation procedure we talked about before

In SVM, we can combine the ideas of dual formulation and kernel tricks for the transformation

This is one of the key ingredients that makes SVM powerful