

CSE 417T

Introduction to Machine Learning

Lecture 17

Instructor: Chien-Ju (CJ) Ho

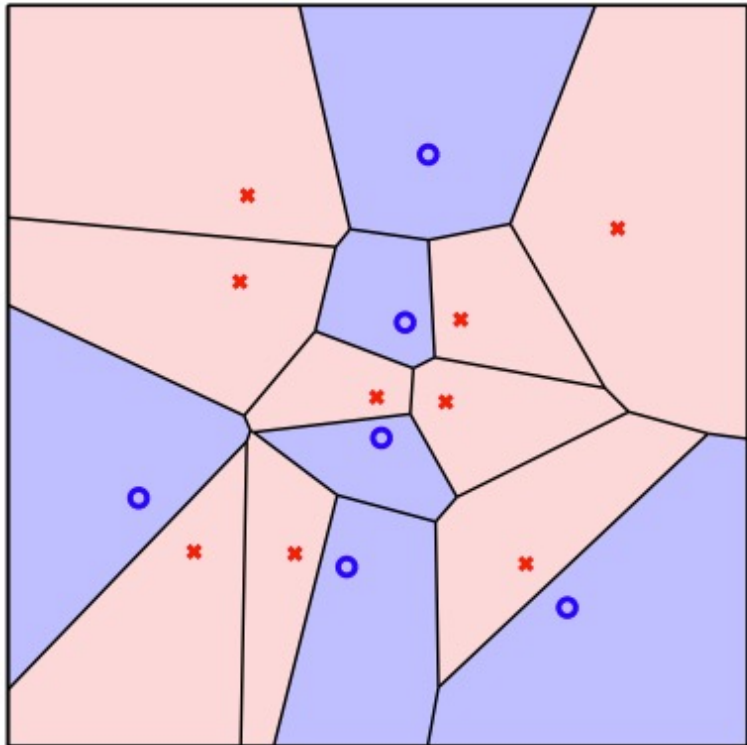
Logistics

- Homework 4 is due April 1 (Friday)
- 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

Recap

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?
- Nearest Neighbor is 2-Optimal
 - When $N \rightarrow \infty$, with high probability,
 $E_{out} \leq 2E_{out}^*$

k -Nearest Neighbor (K-NN)

- k -nearest neighbor (K-NN)
 - $g(\vec{x}) = \text{sign}(\sum_{i=1}^k y_{[i]}(\vec{x}))$
- How to choose k ?
 - Making the choice of k a function of N , denoted by $k(N)$
 - Theorem:
 - For $N \rightarrow \infty$, if $k(N) \rightarrow \infty$ and $\frac{k(N)}{N} \rightarrow 0$
 - Then $E_{in}(g) \rightarrow E_{out}(g)$ and $E_{out}(g) \rightarrow E_{out}(g^*)$
 - E.g., $k(N) = \sqrt{N}$
 - Other practical rules of thumb:
 - $k = 3$ is often a good enough choice
 - Using validation to choose k

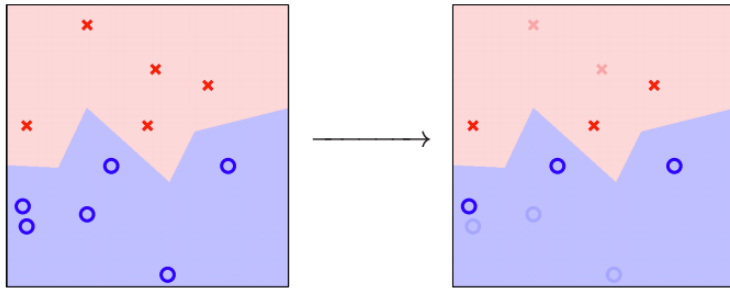
With suitable choice of k , when $N \rightarrow \infty$, we can recover the optimal hypothesis.

Summary of k -NN Properties

- Pros
 - Simple algorithm
 - Good interpretations
 - Nice theoretical guarantees
 - Easy to adapt to regression (average of nearest neighbors) and multi-class classification (majority voting)
- Cons
 - Curse of dimensionality
 - When dimensionality is high, most points won't be close to each other
 - For points to be close, they need to be close in every dimension
 - Computational issue
 - Each prediction requires heavy computation

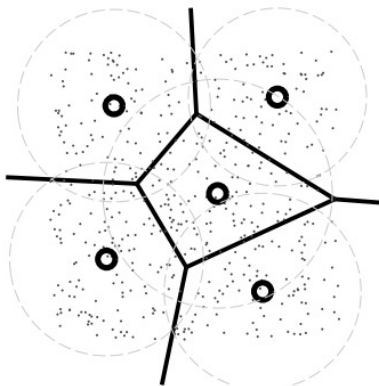
Dealing with Computational Issues

- Reduce the number of data points



- Intuition: remove points that will not impact the decision boundary.
- Generally a hard problem. But there are heuristic approaches (e.g., Condensed Nearest Neighbor).

- Store the data in some data structure to speed up searching



- Intuition: Clustering data points
- For a new data point, we might be able to “ignore” some clusters when searching for nearest neighbor.

Radial Basis Function (RBF)

- Using **distance** to the points as the basis function to form hypothesis

- Radial Basis Function:

- $g(\vec{x}) = \frac{1}{Z(\vec{x})} \sum_{n=1}^N \phi\left(\frac{\|\vec{x} - \vec{x}_n\|}{r}\right) y_n$

- $\phi(s)$: a monotonically decreasing function

- Gaussian RBF (we have seen this in SVM): $\phi(s) = e^{-s}$

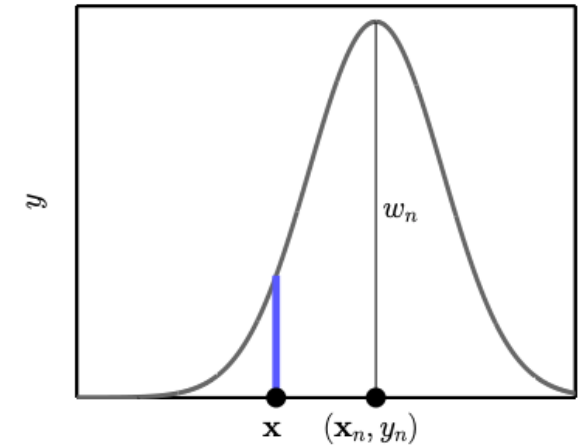
- This is for regression. We can take a sign and make it a classification.

- $Z(\vec{x}) = \sum_{m=1}^N \phi\left(\frac{\|\vec{x} - \vec{x}_m\|}{r}\right)$ is for normalization

Nonparametric and Parametric RBF

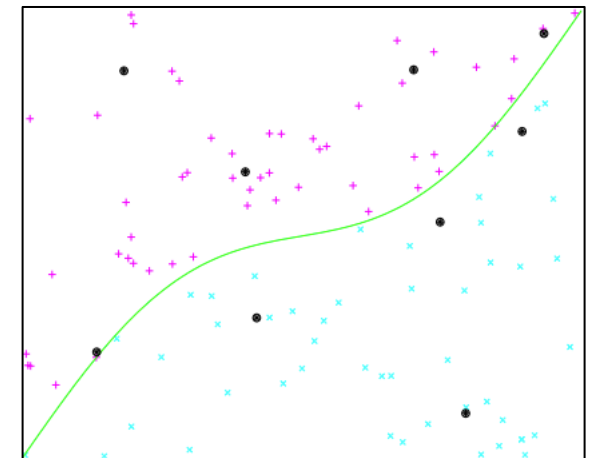
- Nonparametric RBF

- $g(\vec{x}) = \sum_{n=1}^N \frac{y_n}{Z(\vec{x})} \phi\left(\frac{\|\vec{x} - \vec{x}_n\|}{r}\right)$
- $g(\vec{x}) = \sum_{n=1}^N w_n(\vec{x}) \phi\left(\frac{\|\vec{x} - \vec{x}_n\|}{r}\right)$
- The hypothesis is defined by dataset



- Parametric RBF hypothesis set

- $h(\vec{x}) = \sum_{k=1}^K w_k \phi\left(\frac{\|\vec{x} - \vec{\mu}_k\|}{r}\right)$
- Find K represented points (e.g., clustering) $\vec{\mu}_1, \dots, \vec{\mu}_K$
- Learn w_k from data



Connection to Other Hypothesis Sets

- $h(\vec{x}) = \sum_{k=1}^K w_k \phi\left(\frac{\|\vec{x} - \vec{\mu}_k\|}{r}\right)$
- Connection to linear models
 - Parametric RBF is essentially linear models with nonlinear transformation
- Connection to nearest neighbor
 - RBF is based on the similarity to a set of points
- Connection to SVM with RBF Kernel
 - Using K representative points vs. using support vectors
- Connection to Neural Networks
 - RBF can be graphically represented as a one-hidden layer network

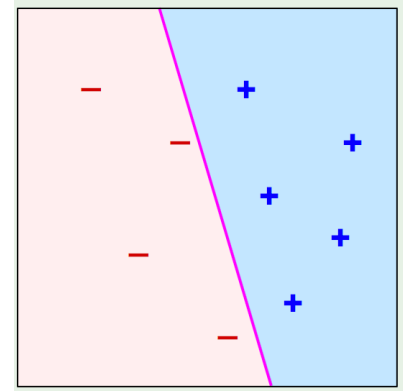
Today's Lecture

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)

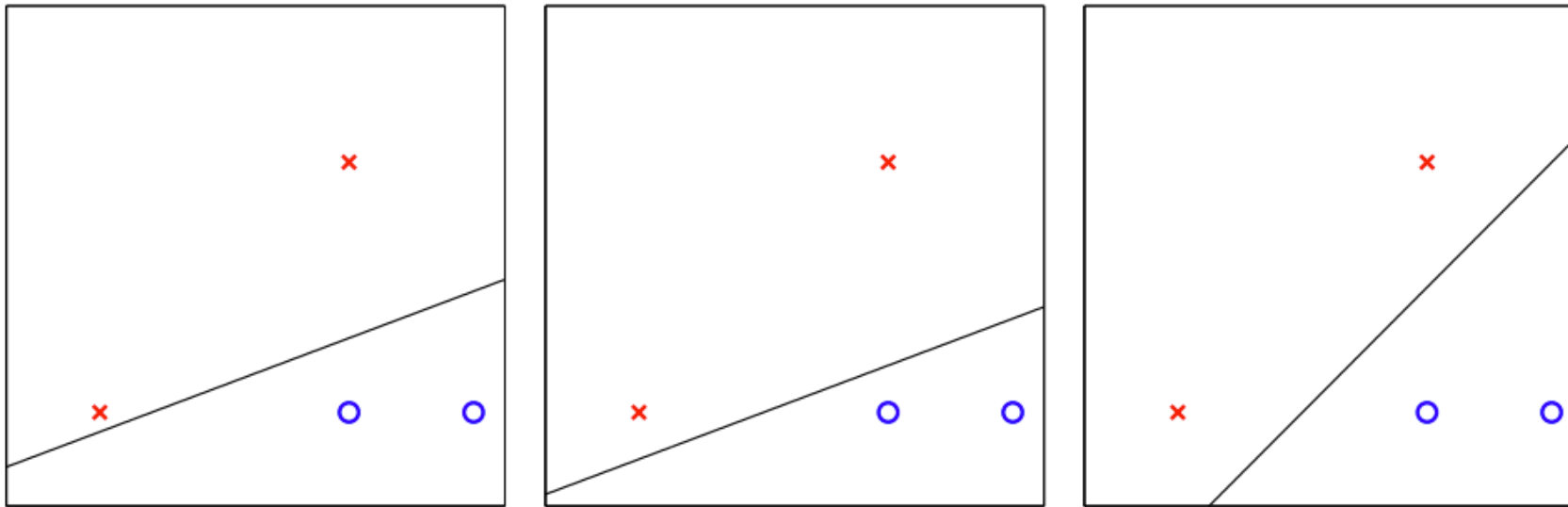
What Do We Know about Linear Classification?

- 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

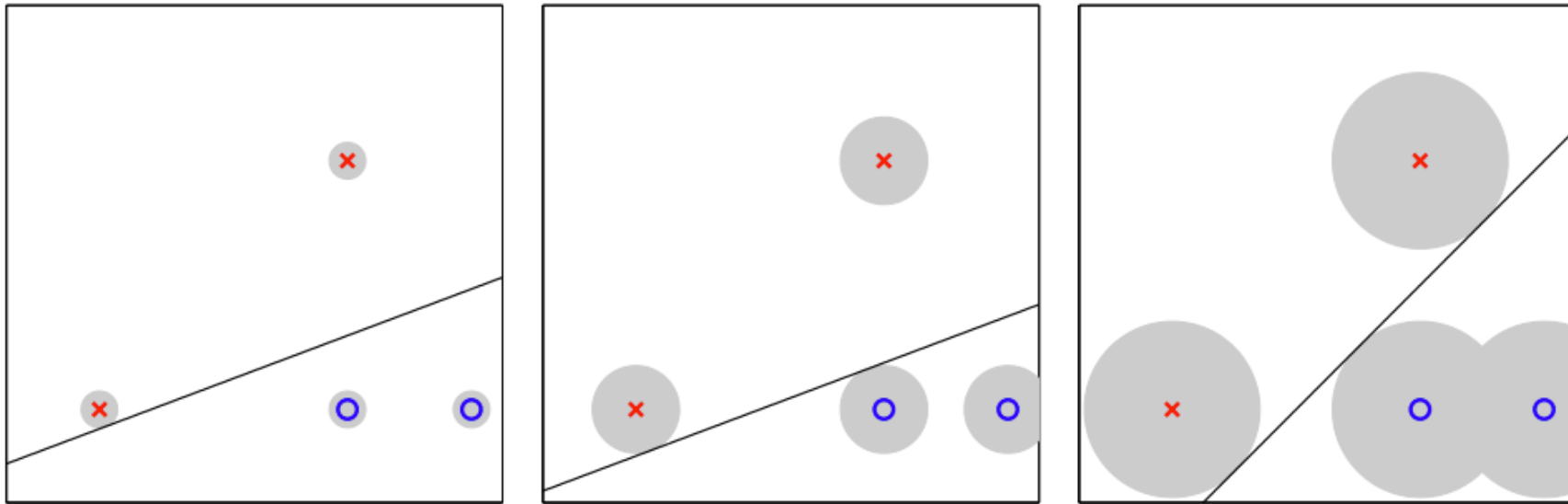
- Which separator would you choose?



Probably the right one.
Why?

Linear Classification

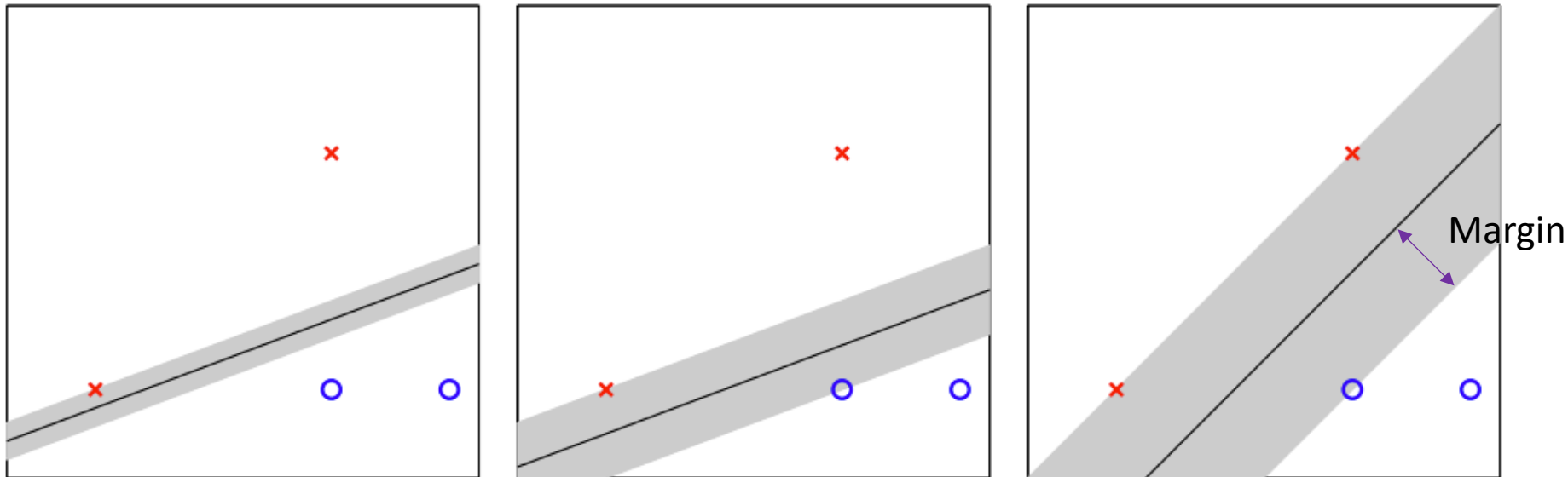
- Which separator would you choose?



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 \Rightarrow more “constrained” hypothesis \Rightarrow 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:

Notations we used so far:

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

Notations we will use in SVM

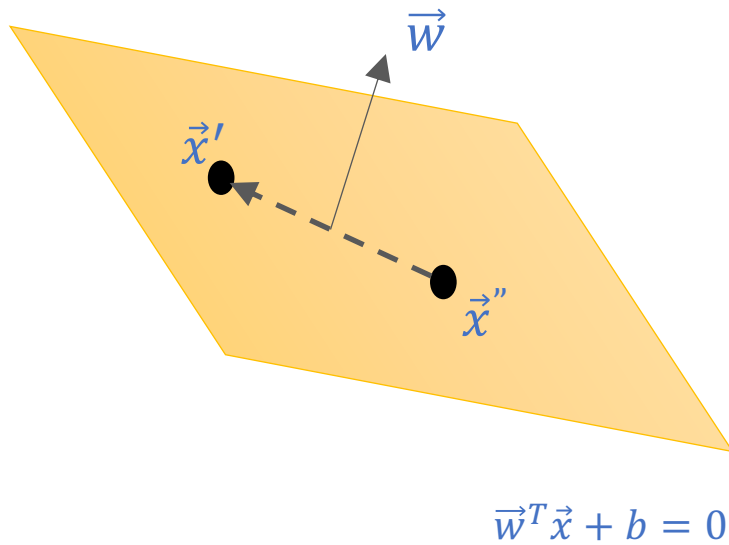
- $\vec{x} = (x_1, \dots, x_d)$
- $\vec{w} = (w_1, \dots, w_d)$
- Linear separator
$$h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x} + b)$$

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

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

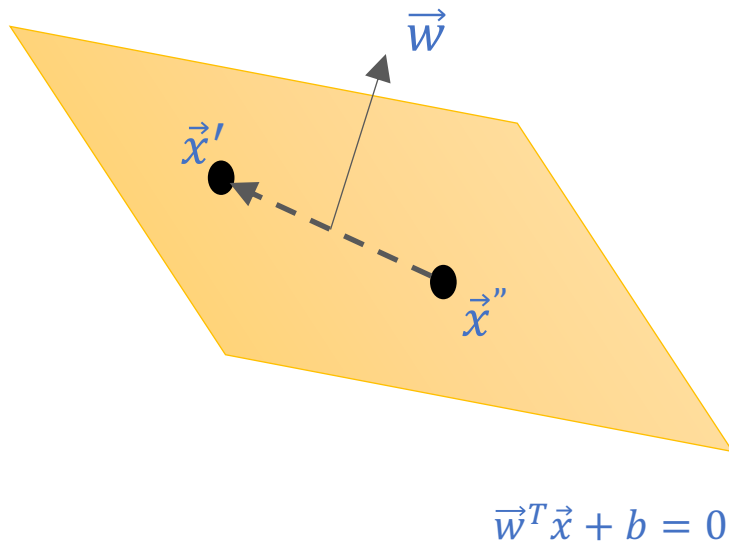
Relevant Review of Linear Algebra

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



Relevant Review of Linear Algebra

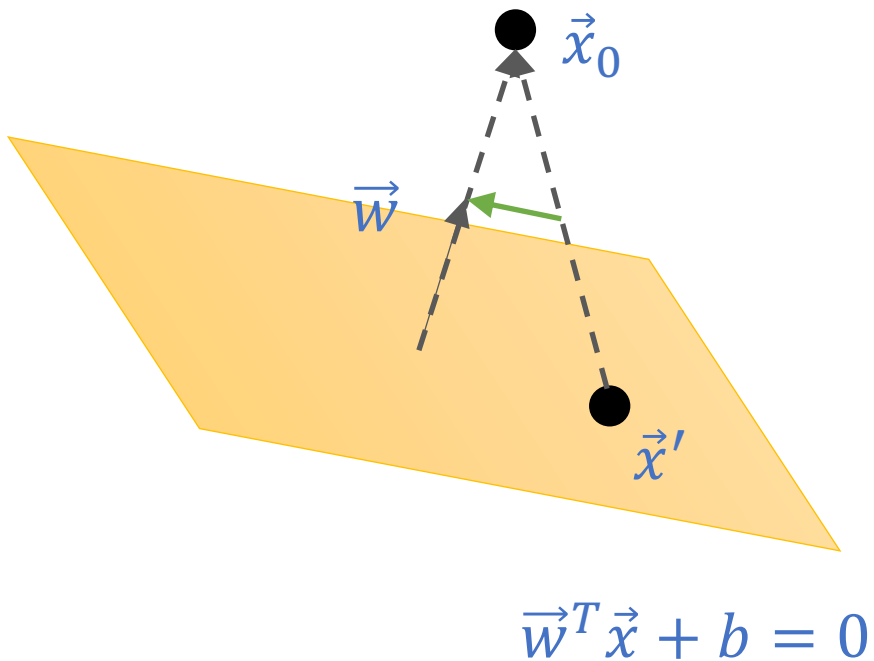
- Claim: \vec{w} is the norm vector of the hyperplane $\vec{w}^T \vec{x} + b = 0$



- 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

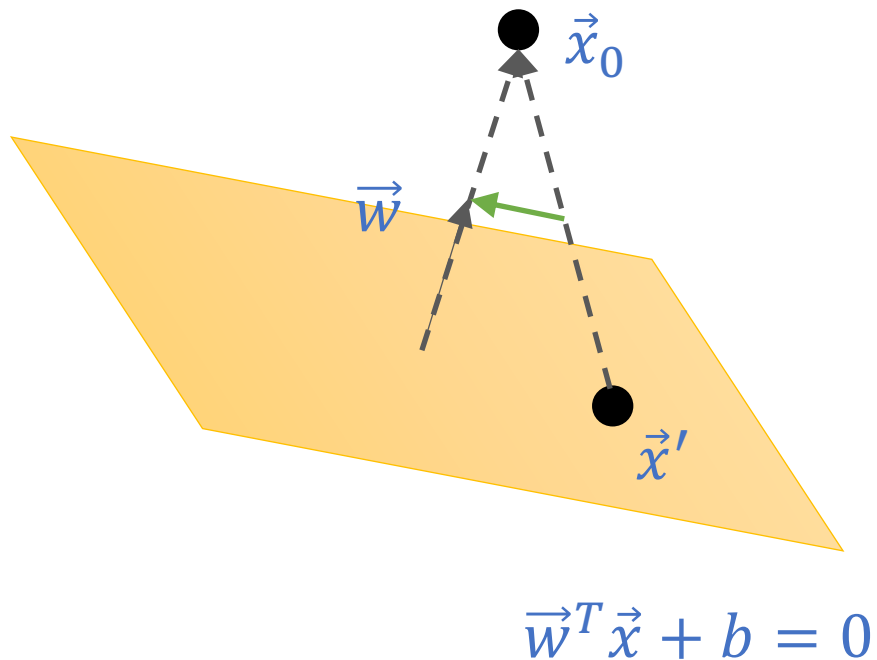
Relevant Review of Linear Algebra

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



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}_0 and the hyperplane

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

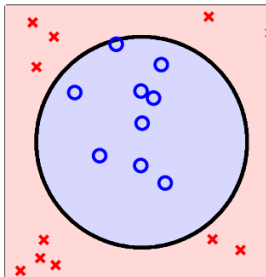
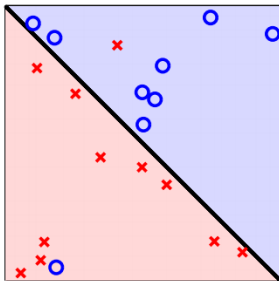
Outline of Our Discussion for SVM

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

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

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

- 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 $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$ 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)
- Margin: shortest distance from the separator to points in D

Hard-Margin SVM

- Goal
 - Given $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$ 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)
 - $y_n = \text{sign}(\vec{w}^T \vec{x}_n + b)$ 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

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

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

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

Hard-Margin SVM

- Goal
 - Given $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$ 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

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

Hard-Margin SVM

- The constrained optimization problem

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

- 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 $\min_n y_n(\vec{w}^T \vec{x}_n + b) = 1$
 - $\text{margin}(\vec{w}, b) = \frac{1}{\|\vec{w}\|}$
 - $y_n(\vec{w}^T \vec{x}_n + b) \geq 1, \forall n$

Hard-Margin SVM

- The constrained optimization problem

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

- Some final adjustments

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

Final Form of Hard-Margin SVM

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

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

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

- There exist efficient QP solvers we can utilize

Short Break and Questions:
How to construct QP for hard-margin SVM

Linear Hard-Margin SVM with QP

- 1: Let $\mathbf{p} = \mathbf{0}_{d+1}$ ($(d+1)$ -dimensional zero vector) and $\mathbf{c} = \mathbf{1}_N$ (N -dimensional vector of ones). Construct matrices Q and A , where

$$Q = \begin{bmatrix} 0 & \mathbf{0}_d^T \\ \mathbf{0}_d & I_d \end{bmatrix}, \quad A = \underbrace{\begin{bmatrix} y_1 & -y_1 \mathbf{x}_1^T \\ \vdots & \vdots \\ y_N & -y_N \mathbf{x}_N^T \end{bmatrix}}_{\text{signed data matrix}}.$$

- 2: Calculate $\begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = \mathbf{u}^* \leftarrow \text{QP}(Q, \mathbf{p}, A, \mathbf{c})$.
- 3: Return the hypothesis $g(\mathbf{x}) = \text{sign}(\mathbf{w}^{*T} \mathbf{x} + b^*)$.

Connection to Regularization

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

- Another way to look at SVM

$$\begin{array}{ll} \text{minimize} & \vec{w}^T \vec{w} \\ \text{subject to} & E_{in}(\vec{w}) = 0 \end{array}$$

- Weight decay regularization

$$\begin{array}{ll} \text{minimize} & E_{in}(\vec{w}) \\ \text{subject to} & \vec{w}^T \vec{w} \leq C \end{array}$$

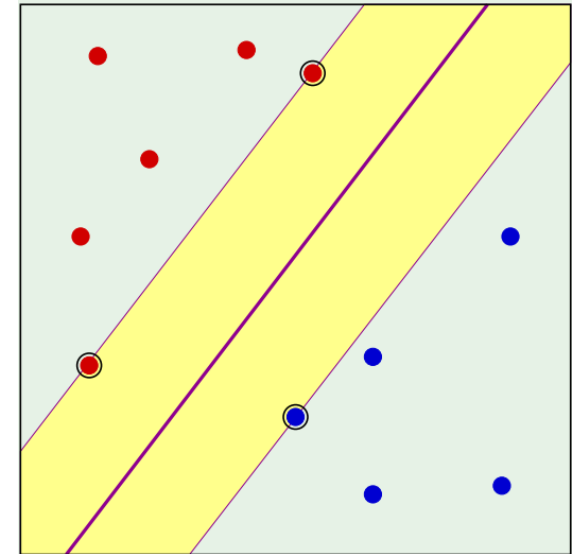
- Maximizing margin is similar to applying regularization!
- You'll see that these two interpretations are somewhat "equivalent" when we introduce Lagrangian next lecture.

Support Vectors

We'll more formally define support vectors next lecture.

- 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, $y_n(\vec{w}^T \vec{x}_n + b) = 1$
(the reverse might not be true)

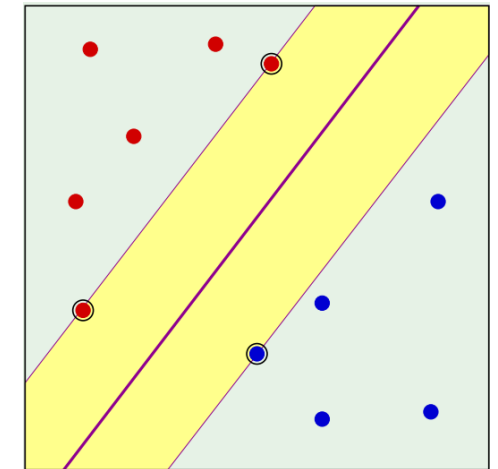
$$\begin{array}{ll} \text{minimize}_{\vec{w}, b} & \frac{1}{2} \vec{w}^T \vec{w} \\ \text{subject to} & y_n(\vec{w}^T \vec{x}_n + b) \geq 1, \forall n \end{array}$$



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

Leave-One-Out Cross Validation (LOOCV)

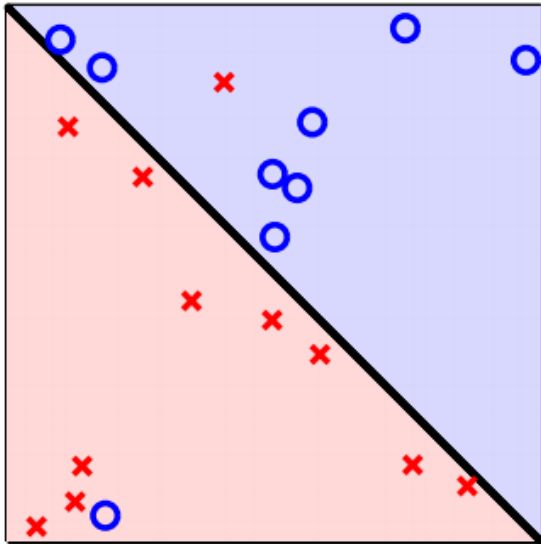
- Two things we know so far
 - Removing non-support vectors will not impact the separator
 - LOOCV error (when not used for model selection) is an unbiased estimate of $E_{out}(N - 1)$ (E_{out} when trained on $N - 1$ points)
- What's the upper bound of LOOCV error for SVM?
 - $E_{LOOCV} \leq \frac{\# \text{ support vectors}}{N}$
- 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)



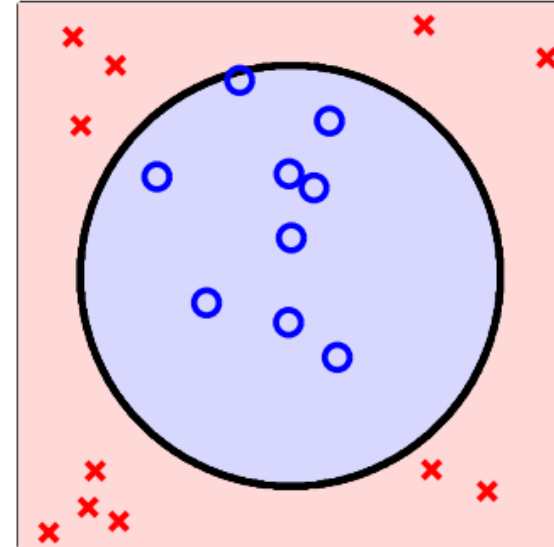
What if Data is Not Linearly Separable

Non-Separable Data

- Two scenarios



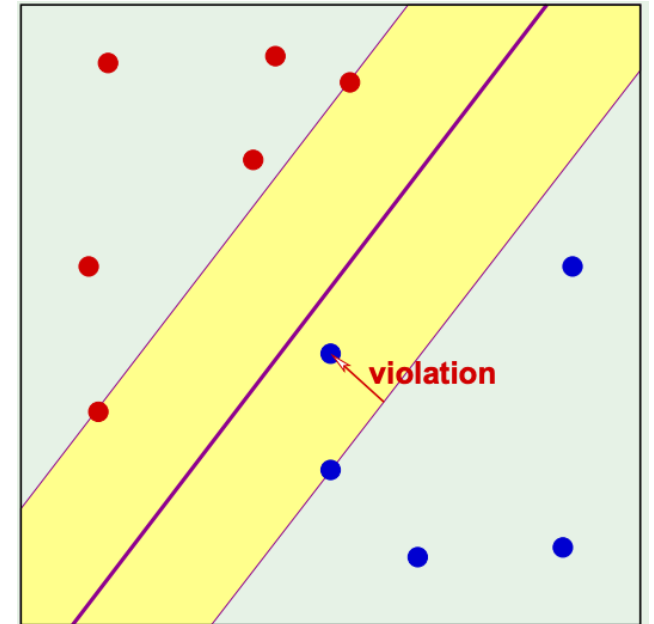
- Tolerate some noise
 - **Soft-Margin SVM**



- Nonlinear transform
 - **Dual formulation and kernel tricks**

Soft-Margin SVM

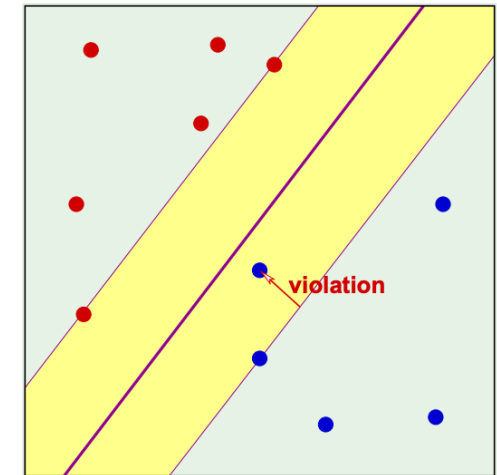
- Intuition: We want to tolerate small noises when maintaining large margin
- For each point (\vec{x}_n, y_n) , we allow a deviation $\xi_n \geq 0$
 - Instead of requiring $y_n(\vec{w}^T \vec{x}_n + b) \geq 1$
 - The constraint becomes
$$y_n(\vec{w}^T \vec{x}_n + b) \geq 1 - \xi_n$$
- We add a penalty 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 $C \sum_{n=1}^N \xi_n$

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

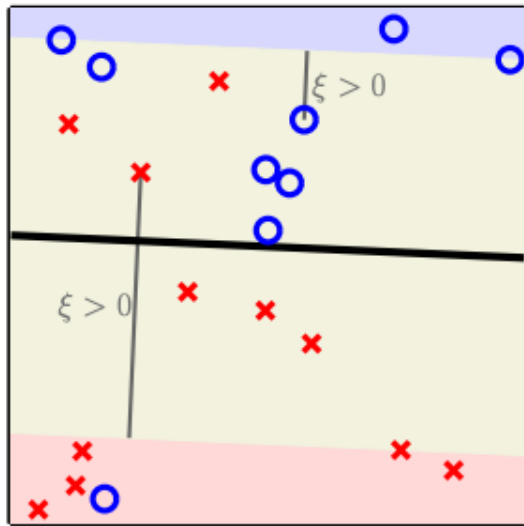


Remarks:

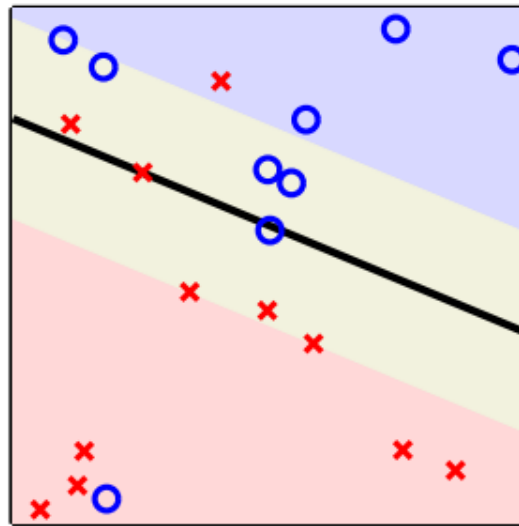
- C is a hyper-parameter we can choose, e.g., using validation
- Soft-margin SVM is still a Quadratic Program, with efficient solvers

Impacts of C in Soft-Margin SVM

- What happens when C is larger
 - less tolerate to noise, having smaller margin



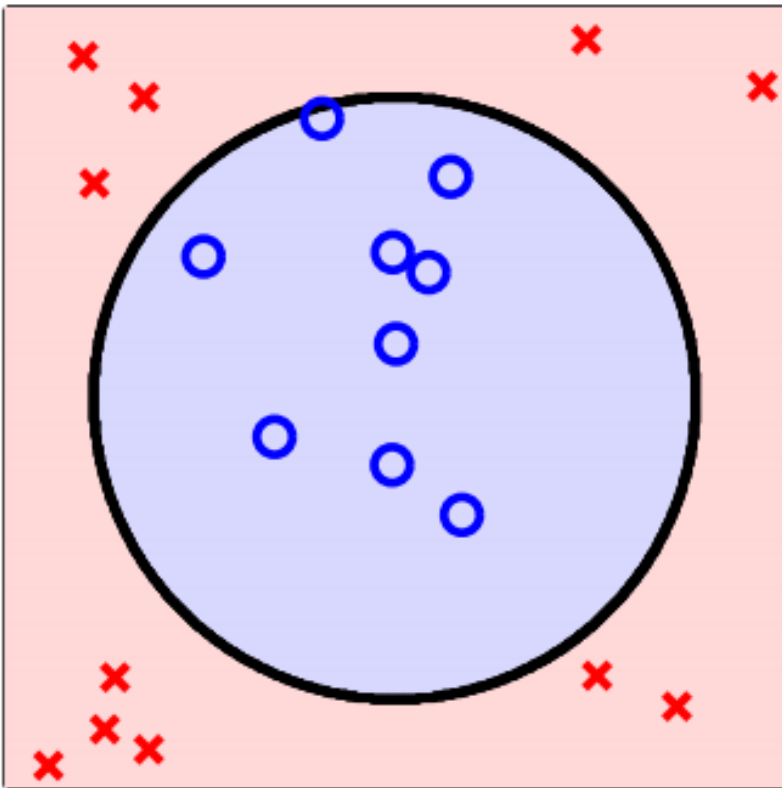
$C = 1$



$C = 500$

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

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

Nonlinear Transform: $\vec{z} = \Phi(\vec{x})$

- Consider hard-margin SVM in the \vec{z} space

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

Involves changing \vec{w} and \vec{z} .
The computation grows as the dimension of the \vec{z} space grows

- There exists a corresponding dual formulation (more next lecture)

$$\begin{array}{ll} \text{maximize}_{\vec{\alpha}} & \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \vec{z}_n^T \vec{z}_m \\ \text{subject to} & \sum_{n=1}^N \alpha_n y_n = 0 \\ & \alpha_n \geq 0, \forall n \end{array}$$

The only difference for the nonlinear transformation is from calculating $\vec{x}_n^T \vec{x}_m$ to $\vec{z}_n^T \vec{z}_m$

- Why dual
 - The optimal primal is the same as the optimal dual
 - We can infer the optimal primal solutions from the optimal dual solutions

Next Lecture

- Primal-dual formulations using Lagrangian

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

$$\begin{aligned} &\text{maximize}_{\vec{\alpha}} \quad \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \vec{z}_n^T \vec{z}_m \\ &\text{subject to} \quad \sum_{n=1}^N \alpha_n y_n = 0 \\ &\quad \alpha_n \geq 0, \forall n \end{aligned}$$

- Kernel tricks
 - Intuition: If we can find an efficient way to calculate $\vec{z}_n^T \vec{z}_m$, we can efficiently derive the optimal dual to infer the optimal primal.
 - Doing nonlinear transform without sacrificing much about computation.