CSE 417T: Introduction to Machine Learning

Lecture 20: Support Vector Machines

Henry Chai

11/08/18

Recall: kNN

- Classify a point as the most common label among the labels of the k nearest training points
- If we have a binary classification problem and k is odd:

$$g(\vec{x}) = \mathrm{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- k controls the complexity of the hypothesis set $\Longrightarrow$ k affects how well the learned hypothesis will generalize
  - $k = 3$
  - $k = \lfloor\sqrt{n}\rfloor$
  - Cross-validation

kNN Pros and Cons

- Pros:
  - Intuitive / explainable
  - No training / retraining
  - Self-regularizes
  - Provably near-optimal in terms of E_{out}
- Cons:
  - Computationally expensive
    - Always needs to store all data: $O(nD)$
    - Computing $g(\vec{x})$ requires computing $d(\vec{x}, \vec{x}') \; \forall \, \vec{x}' \in \mathcal{D}$ and finding the k closest points: $O(nD + n\log(k))$
  - Suffers from the "curse of dimensionality"

kNN

- kNN only considers some points and weights them equally
- What if we considered all points but weighted them unequally?
- Intuition: all points are useful but some points are more useful than others!
- Bonus: no need to choose k

Radial Basis Functions (RBF)

- Suppose we have a binary classification problem
- Recall for kNN (when k is odd):

$$g(\vec{x}) = \text{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- For RBF:

$$g(\vec{x}) = \text{sign}\left(\sum_{i=1}^{n} \phi(\vec{x}, \overrightarrow{x_i}) y_i\right)$$

Radial Basis Functions (RBF)

- Suppose we have a binary classification problem
- Recall for kNN (when k is odd):

$$g(\vec{x}) = \operatorname{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- For RBF:

$$g(\vec{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \left(\frac{\phi(\vec{x}, \vec{x_i})}{\sum_{i=1}^{n} \phi(\vec{x}, \vec{x_i})}\right) y_i\right)$$

Radial Basis Functions (RBF)

- Suppose we have a binary classification problem
- Recall for kNN (when k is odd):

$$g(\vec{x}) = \operatorname{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- For RBF:

$$g(\vec{x}) = \operatorname{sign}\left(\sum_{i=1}^{n}\left(\frac{\phi(\|\vec{x}-\vec{x_i}\|)}{\sum_{i=1}^{n}\phi(\|\vec{x}-\vec{x_i}\|)}\right)y_i\right)$$

Radial Basis Functions (RBF)

- Suppose we have a binary classification problem
- Recall for kNN (when k is odd):

$$g(\vec{x}) = \mathrm{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- For RBF:

$$g(\vec{x}) = \mathrm{sign}\left(\sum_{i=1}^{n}\left(\frac{\phi\left(\frac{\|\vec{x}-\vec{x_i}\|}{r}\right)}{\sum_{i=1}^{n}\phi\left(\frac{\|\vec{x}-\vec{x_i}\|}{r}\right)}\right)y_i\right)$$

Setting r

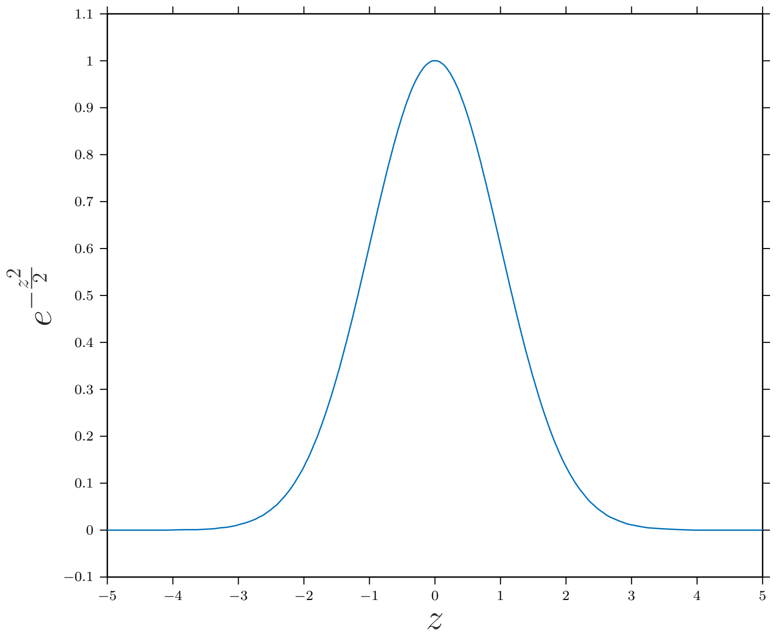
- If r is really small, then $\frac{\|\vec{x}-\vec{x_i}\|}{r}$ will always be large (unless $\|\vec{x}-\vec{x_i}\|$ is small)
- If $\frac{\|\vec{x}-\vec{x_i}\|}{r}$ is large, then $\phi\left(\frac{\|\vec{x}-\vec{x_i}\|}{r}\right)$ will always be small (unless $\|\vec{x}-\vec{x_i}\|$ is small)
- The smaller r is, the closer $\vec{x}$ has to be to $\vec{x_i}$ in order for $\phi\left(\frac{\|\vec{x}-\vec{x_i}\|}{r}\right)$ to be non-zero
- As $r \to 0$, the RBF hypothesis approaches the nearest-neighbor hypothesis
- Set r using cross-validation

Choosing ϕ

- Intuitively, we want points close to $\vec{x}$ to have large weights and points far from $\vec{x}$ to have small weights
  - $\phi(z)$ is maximized when $z = 0$
  - $\phi(z) \to 0$ as $z \to \infty$

- Most common choice is the Gaussian kernel:

$$\phi(z) = e^{-\frac{z^2}{2}}$$



Radial Basis Functions (RBF)

- Suppose we have a binary classification problem
- Recall for kNN (when k is odd):

$$g(\vec{x}) = \text{sign}\left(\sum_{i=1}^{k} y_{[i]}(\vec{x})\right)$$

- For RBF:

$$g(\vec{x}) = \text{sign}\left(\sum_{i=1}^{n}\left(\frac{e^{-\frac{\|\vec{x}-\vec{x_i}\|^2}{2r^2}}}{\sum_{i=1}^{n} e^{-\frac{\|\vec{x}-\vec{x_i}\|^2}{2r^2}}}\right) y_i\right)$$

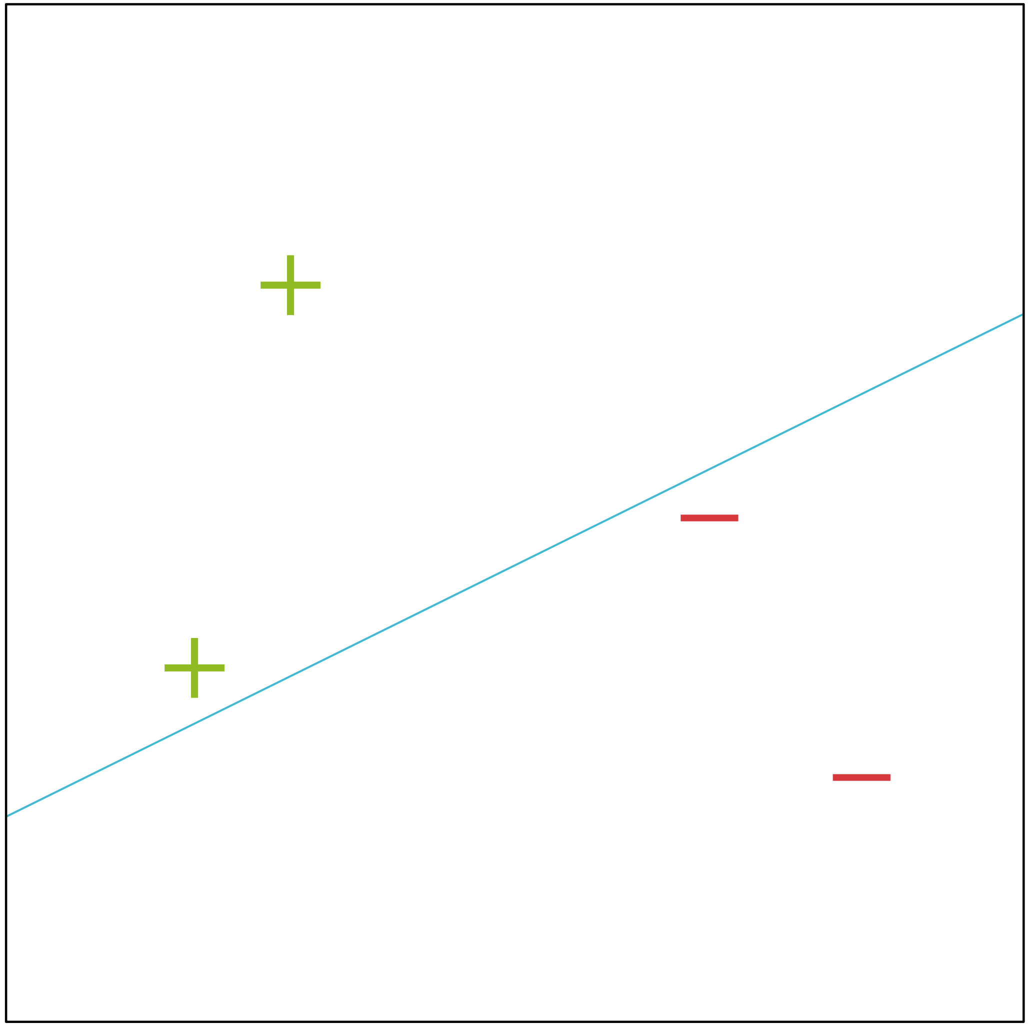
Recall: Perceptron

- A linear model for classification:

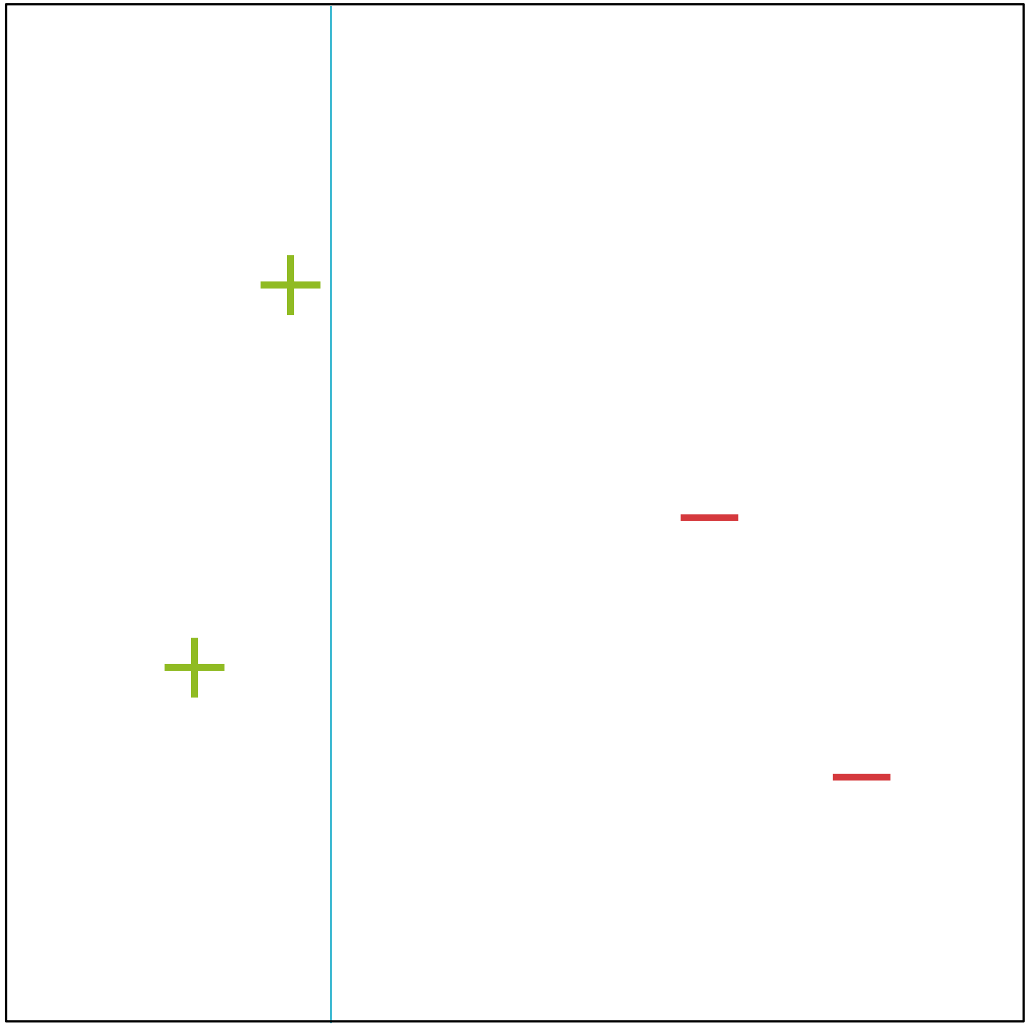
$$h(\vec{x}) = \operatorname{sign}\left(\sum_{i=0}^{D} w_i x_i\right) = \operatorname{sign}\left(\vec{w}^{\mathrm{T}}\vec{x}\right)$$

- Perceptron Learning Algorithm (PLA) finds **a** linear separator in finite time, **if the data is linearly separable**
  - Input: $\mathcal{D} = \{(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)\}$
  - Initialize $\vec{w}$ to all zeros or (small) random numbers
  - While ∃ a misclassified training point
    - Pick a misclassified training point, $(\vec{x}, y)$
    - Update $\vec{w}$: $\vec{w} = \vec{w} + y\vec{x}$

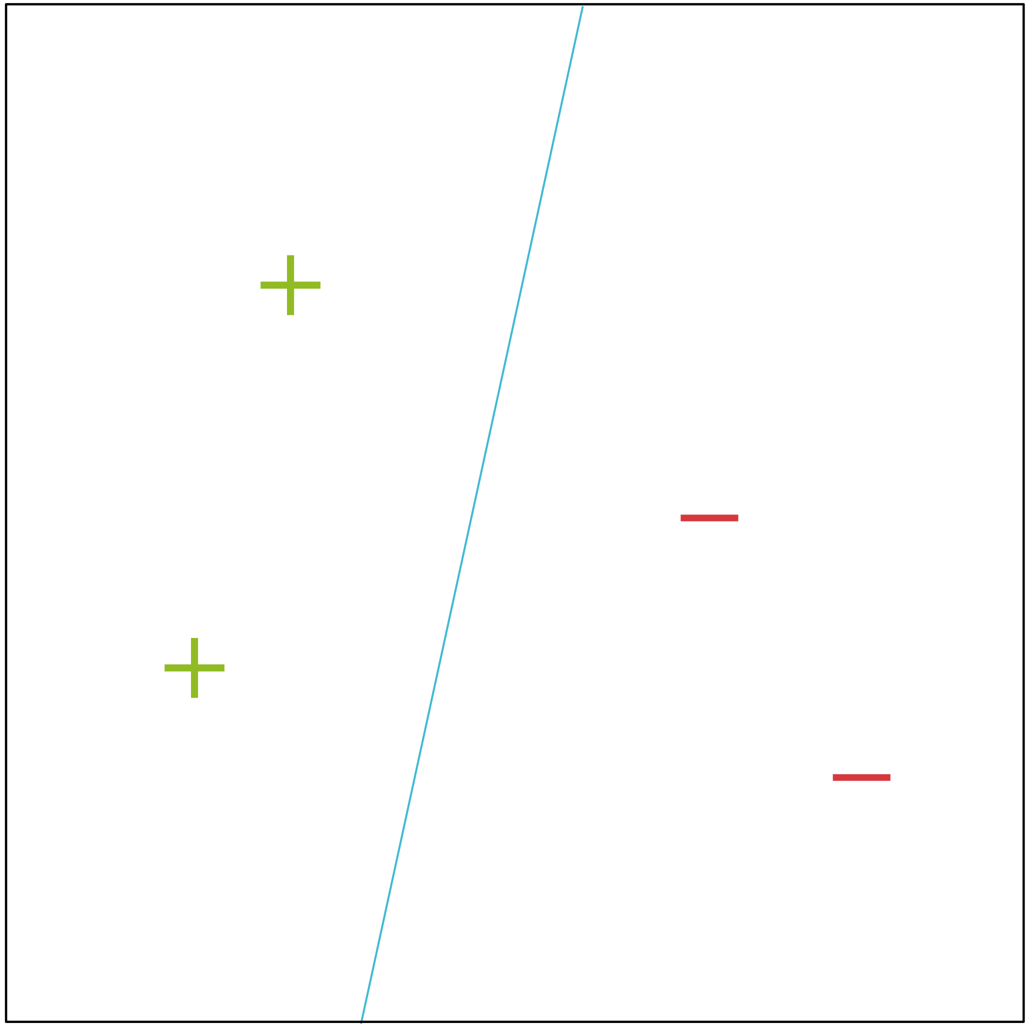
Linearly Separable Data

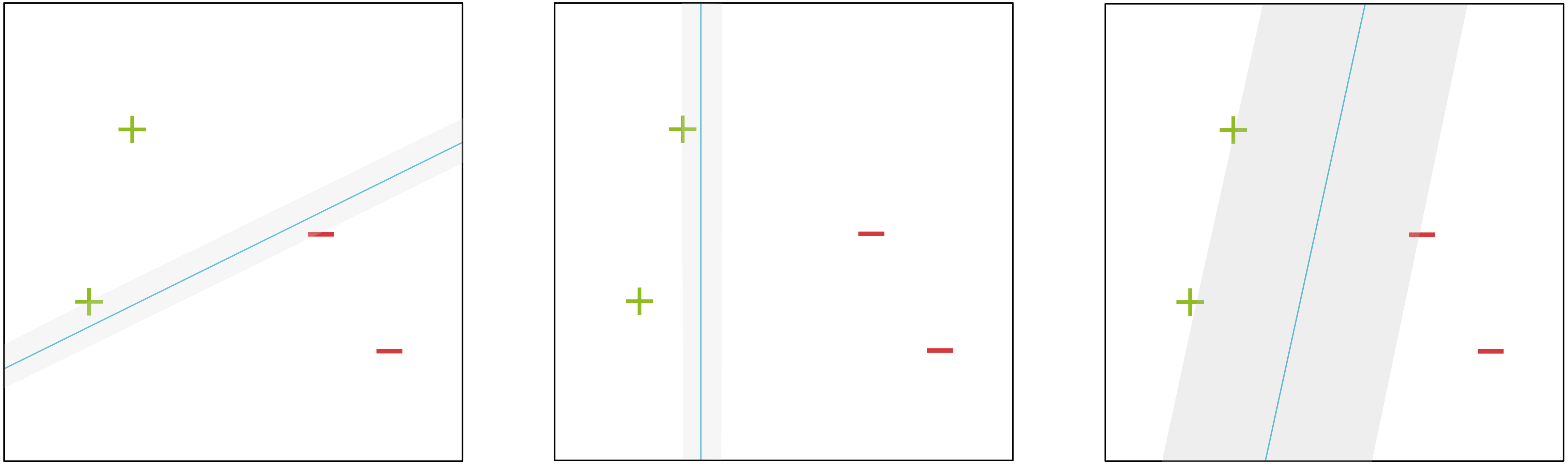


Linearly Separable Data



Linearly Separable Data





Which linear separator is best?

Maximal Margin Linear Separators

- The margin of a separating hyperplane is the distance between the hyperplane and the nearest training point

- Questions:
  - How can we efficiently find a maximal-margin linear separator?
  - Why are linear separators with larger margins better?
  - What can we do if the data is not linearly separable?

Hyperplanes

- For linear models, hypotheses are D-dimensional hyperplanes defined by a weight vector, $[w_0, \vec{w}]$

$$[w_0, \vec{w}]^T [1, \vec{x}] = 0 \;\rightarrow\; \vec{w}^T \vec{x} + w_0 = 0$$

- Problem: there are infinitely many weight vectors that describe the same hyperplane
  - $x_1 + 2x_2 + 2 = 0$ is the same line as $2x_1 + 4x_2 + 4 = 0$, which is the same line as $1000000x_1 + 2000000x_2 + 2000000 = 0$
- Solution: normalize weight vectors

Normalizing Hyperplanes

- Given a dataset $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_n}, y_n)\}$ where $\mathcal{Y} = \{-1, +1\}$, $[w_0, \vec{w}]$ is a linear separator if
$$y_i(\vec{w}^T\overrightarrow{x_i} + w_0) > 0 \;\forall\; (\overrightarrow{x_i}, y_i) \in \mathcal{D}$$
- $h(\vec{x}) = \vec{w}^T\vec{x} + w_0 = 0$ is a separating hyperplane if
$$\min_{(\overrightarrow{x_i}, y_i) \in \mathcal{D}} y_i(\vec{w}^T\overrightarrow{x_i} + w_0) = 1$$
- If $[w_0, \vec{w}]$ is a linear separator, then $h(\vec{x}) = \frac{\vec{w}^T}{\rho}\vec{x} + \frac{w_0}{\rho} = 0$ is a separating hyperplane where
$$\rho = \min_{(\overrightarrow{x_i}, y_i) \in \mathcal{D}} y_i(\vec{w}^T\overrightarrow{x_i} + w_0)$$

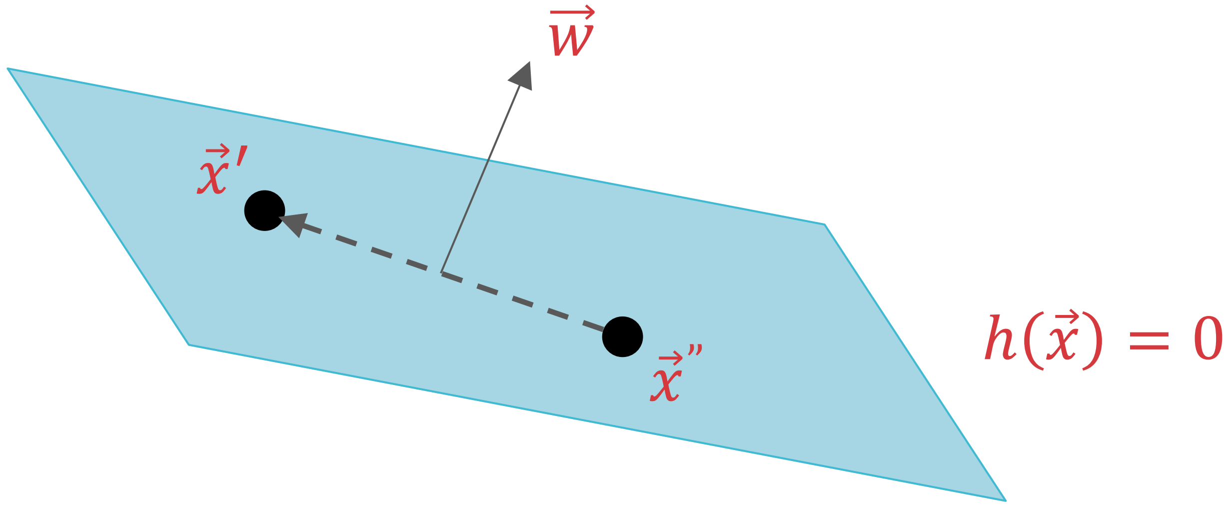
Computing the Margin

- Claim: the vector $\vec{w}$ is orthogonal to the hyperplane $h(\vec{x}) = \vec{w}^T\vec{x} + w_0 = 0$
- Proof:
  - A vector is orthogonal to a hyperplane if it is orthogonal to every vector in that hyperplane
  - Vectors $\vec{a}$ and $\vec{b}$ are orthogonal if $\vec{a}^T\vec{b} = 0$
  - Let $\vec{x}'$ and $\vec{x}''$ be two arbitrary points on $h(\vec{x}) = 0$
    - $\vec{x}' - \vec{x}''$ is a vector on $h(\vec{x}) = 0$
    - $\vec{w}^T\vec{x}' + w_0 = 0 \rightarrow \vec{w}^T\vec{x}' = -w_0$
  - $\vec{w}^T(\vec{x}' - \vec{x}'') = \vec{w}^T\vec{x}' - \vec{w}^T\vec{x}'' = -w_0 + w_0 = 0$ ■

Computing the Margin

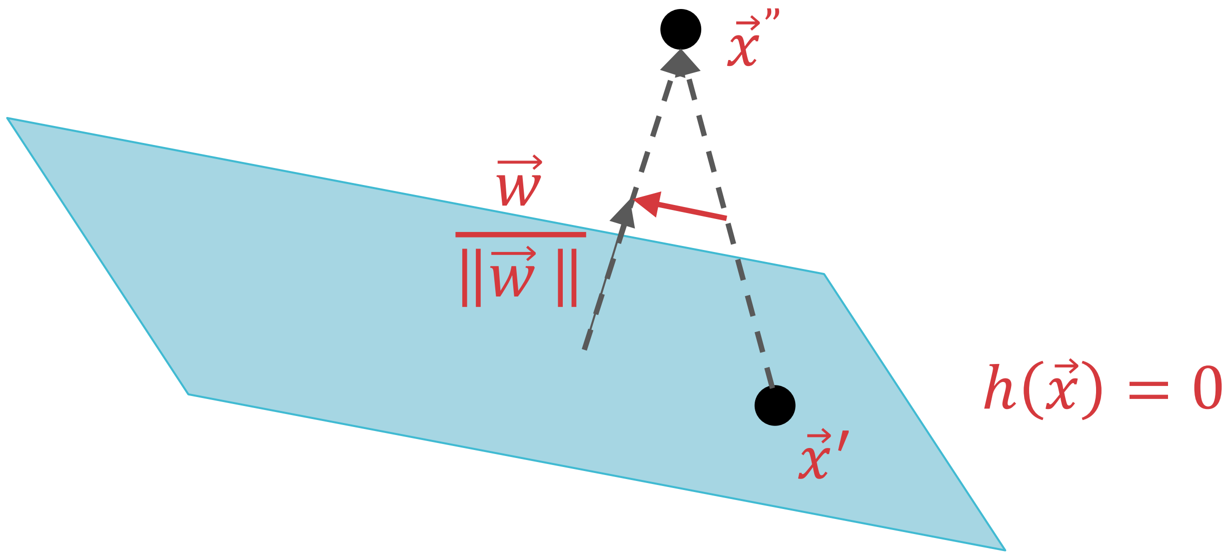
- Claim: the vector $\vec{w}$ is orthogonal to the hyperplane

$$h(\vec{x}) = \vec{w}^T\vec{x} + w_0 = 0$$



Computing the Margin

- Let $\vec{x}'$ be an arbitrary point on the hyperplane $h(\vec{x}) = \vec{w}^T\vec{x} + w_0 = 0$ and let $\vec{x}''$ be an arbitrary point
- The distance between $\vec{x}''$ and $h(\vec{x}) = 0$ is equal to the magnitude of the projection of $\vec{x}'' - \vec{x}'$ onto $\frac{\vec{w}}{\|\vec{w}\|}$, the unit vector orthogonal to $h(\vec{x}) = 0$



Computing the Margin

- Let $\vec{x}'$ be an arbitrary point on the hyperplane $h(\vec{x}) = \vec{w}^T\vec{x} + w_0 = 0$ and let $\vec{x}''$ be an arbitrary point
- The distance between $\vec{x}''$ and $h(\vec{x}) = 0$ is equal to the magnitude of the projection of $\vec{x}'' - \vec{x}'$ onto $\frac{\vec{w}}{||\vec{w}||}$, the unit vector orthogonal to $h(\vec{x}) = 0$

$$d(\vec{x}'', h) = \left|\frac{\vec{w}^T(\vec{x}'' - \vec{x}')}{||\vec{w}||}\right| = \frac{|\vec{w}^T\vec{x}'' - \vec{w}^T\vec{x}'|}{||\vec{w}||} = \frac{|\vec{w}^T\vec{x}'' + w_0|}{||\vec{w}||}$$

Computing the Margin

- The margin of a separating hyperplane is the distance between the hyperplane and the nearest training point

$$\min_{(\vec{x_i}, y_i) \in \mathcal{D}} d(\vec{x_i}, h) = \min_{(\vec{x_i}, y_i) \in \mathcal{D}} \frac{|\vec{w}^T \vec{x_i} + w_0|}{||\vec{w}||}$$

$$= \frac{1}{||\vec{w}||} \min_{(\vec{x_i}, y_i) \in \mathcal{D}} |\vec{w}^T \vec{x_i} + w_0|$$

$$= \frac{1}{||\vec{w}||} \min_{(\vec{x_i}, y_i) \in \mathcal{D}} y_i(\vec{w}^T \vec{x_i} + w_0)$$

$$= \frac{1}{||\vec{w}||} = \frac{1}{\sqrt{\vec{w}^T \vec{w}}}$$

Maximizing the Margin

$$\text{maximize} \quad \frac{1}{\sqrt{\vec{w}^T\vec{w}}}$$

$$\text{subject to} \quad \min_{(\vec{x}_i, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x}_i + w_0) = 1$$

$$\Downarrow$$

$$\text{maximize} \quad \frac{1}{\vec{w}^T\vec{w}}$$

$$\text{subject to} \quad \min_{(\vec{x}_i, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x}_i + w_0) = 1$$

$$\Downarrow$$

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

$$\text{subject to} \quad \min_{(\vec{x}_i, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x}_i + w_0) = 1$$

$$\Downarrow$$

$$\text{minimize} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$\text{subject to} \quad \min_{(\vec{x}_i, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x}_i + w_0) = 1$$

Maximizing the Margin

$$\text{minimize} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$\text{subject to} \quad \min_{(\vec{x_i}, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x_i} + w_0) = 1$$

$$\Downarrow$$

$$\text{minimize} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + w_0) \geq 1 \ \forall (\vec{x_i}, y_i) \in \mathcal{D}$$

- If $[w_0^*, \vec{w}^*]$ is the optimal solution, then $\exists$ at least one training point $(\vec{x_i}, y_i) \in \mathcal{D}$ s.t. $y_i\left(\vec{w}^{*T}\vec{x_i} + w_0^*\right) = 1$
  - All points $(\vec{x_i}, y_i) \in \mathcal{D}$ where $y_i\left(\vec{w}^{*T}\vec{x_i} + w_0^*\right) = 1$ are known as support vectors

Maximizing the Margin

$$\text{minimize} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$\text{subject to} \quad \min_{(\vec{x_i}, y_i) \in \mathcal{D}} y_i(\vec{w}^T\vec{x_i} + w_0) = 1$$

$$\Downarrow$$

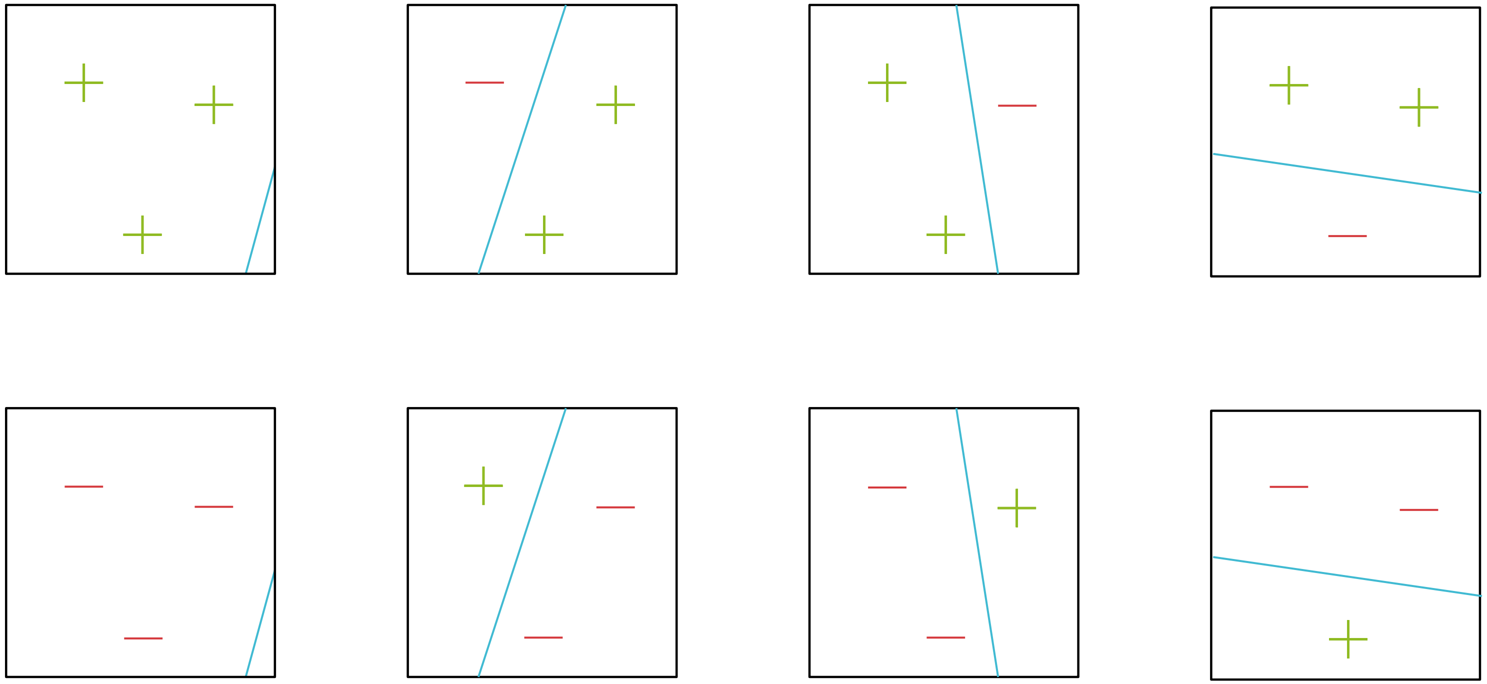
$$\text{minimize} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + w_0) \geq 1 \; \forall (\vec{x_i}, y_i) \in \mathcal{D}$$

- Converting the non-linear constraint (involving the min function) to n linear constraints allows the optimization problem to be solved (approximately) using quadratic programming (QP) in $O(D^3)$ time

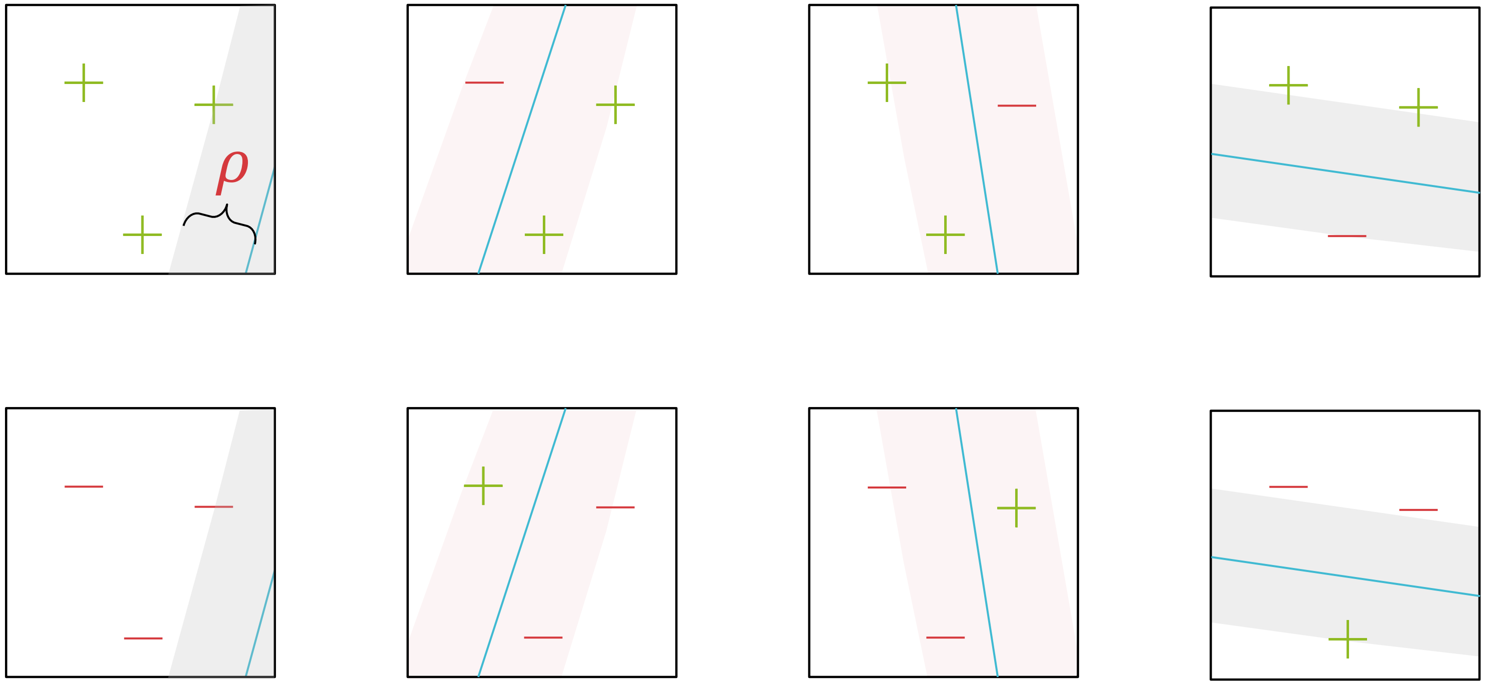
Why Maximal Margins?

- Consider three points in a **bounded** input space
- $\mathcal{H}$ = linear separators can shatter any such three points



Why Maximal Margins?

- Consider three points in a **bounded** input space
- $\mathcal{H}_\rho$ = linear separators with minimum margin ρ cannot always shatter all sets of three points



VC-dimension of $\mathcal{H}_\rho$

- If the input space, $\mathcal{X}$, is a D-dimensional sphere of radius R, then:

$$d_{VC}(\mathcal{H}_\rho) \leq \min\left(D, \left\lceil \frac{R^2}{\rho^2} \right\rceil\right) + 1$$

Regularization

$$\text{minimize} \quad E_{in}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{w}^T x_i - y_i)^2$$

$$\text{subject to} \quad \sum_{j=0}^{d} w_j^2 \le C$$

	SVM	Regularization
minimize	$\frac{1}{2}\vec{w}^T\vec{w}$	$E_{in}(\vec{w})$
subject to	$E_{in}(\vec{w}) = 0$	$\vec{w}^T\vec{w} \le C$