Machine Learning 520
Advanced Machine Learning

Lesson 4: Support Vector Machine (SVM)

UNIVERSITY *of* WASHINGTON

Today's Agenda

- Maximum Margin
- Hinge loss
- SVM with linear kernel
- Kernel tricks
- SVM with polynomial kernel
- SVM with radial basis function kernel
- Support Vector Regression

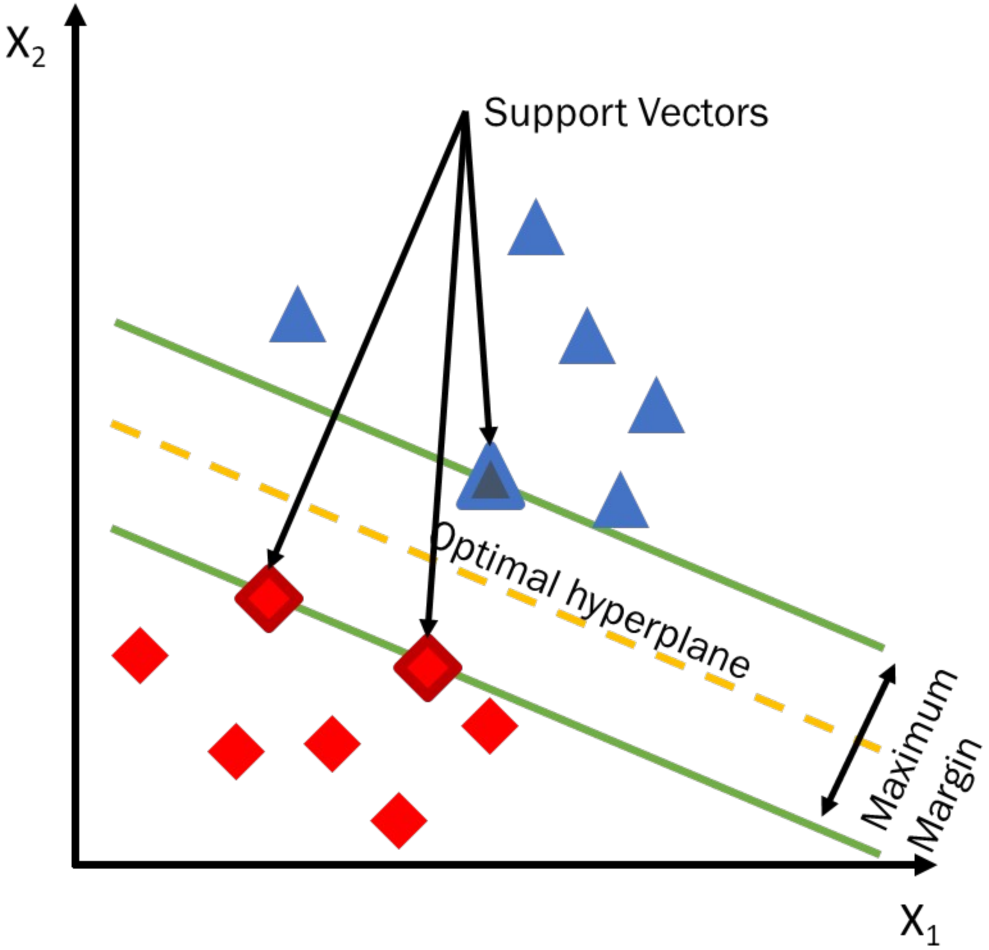
Learning Objectives

By the end of this session, you should be able to:

- Describe the intuition behind maximum margin.
- Differentiate hinge loss from other losses.
- Define kernel tricks and how to effect implicit feature transformation using kernels.
- Describe how similarity is computed using the polynomial kernel.
- Produce a support vector machine classification model based on polynomial kernel and radial basis function kernel.
- Produce a support vector machine regression model with a statistically significant improvement over the null model.

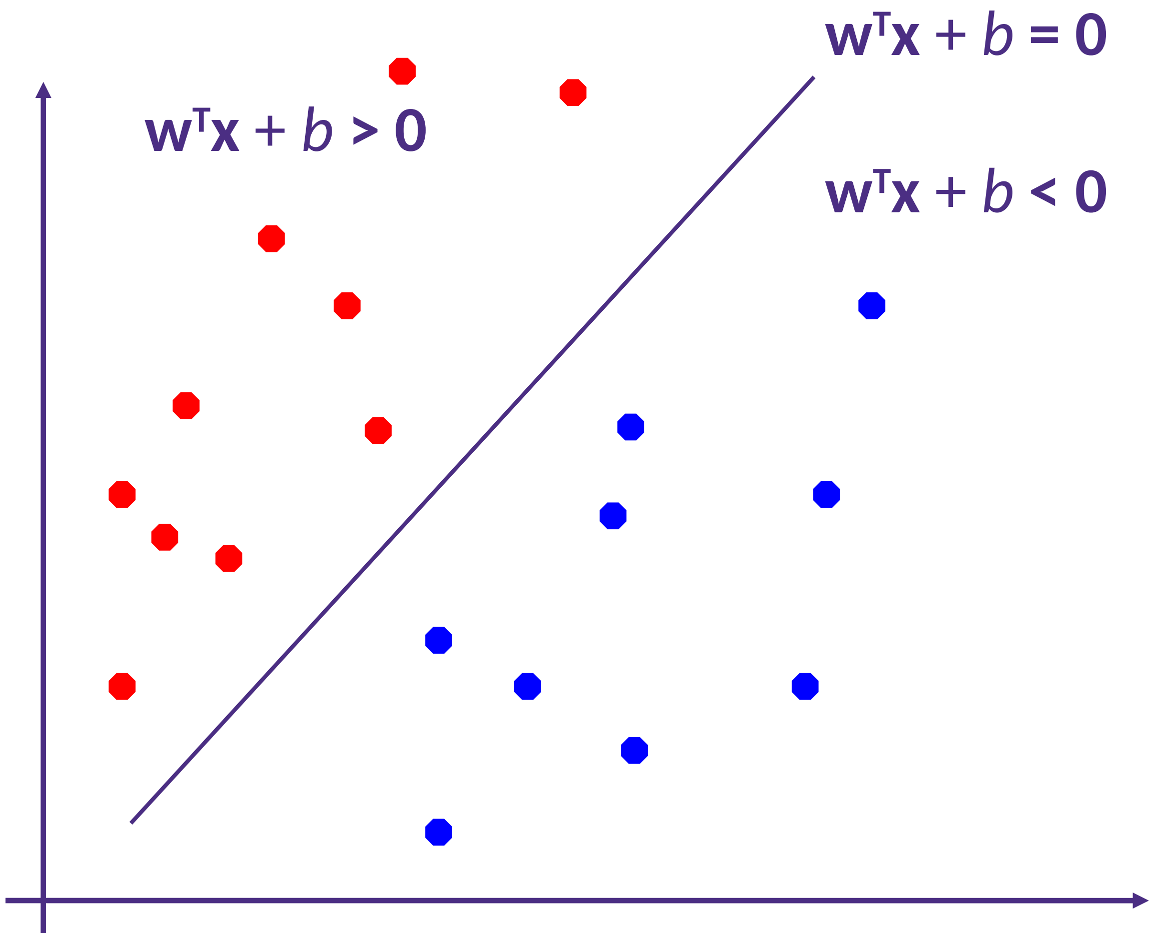
SVMs in a Nutshell…

- SVM views the input data points as two sets of vectors in an n-dimensional space (where n is the number of features)
- It constructs **two vectors that maximize the margin (distance) between the** inner most training data points based on their “similarity”
- The optimal solution boundary is an equidistant line in between the two margins called a hyperplane



Linear Model

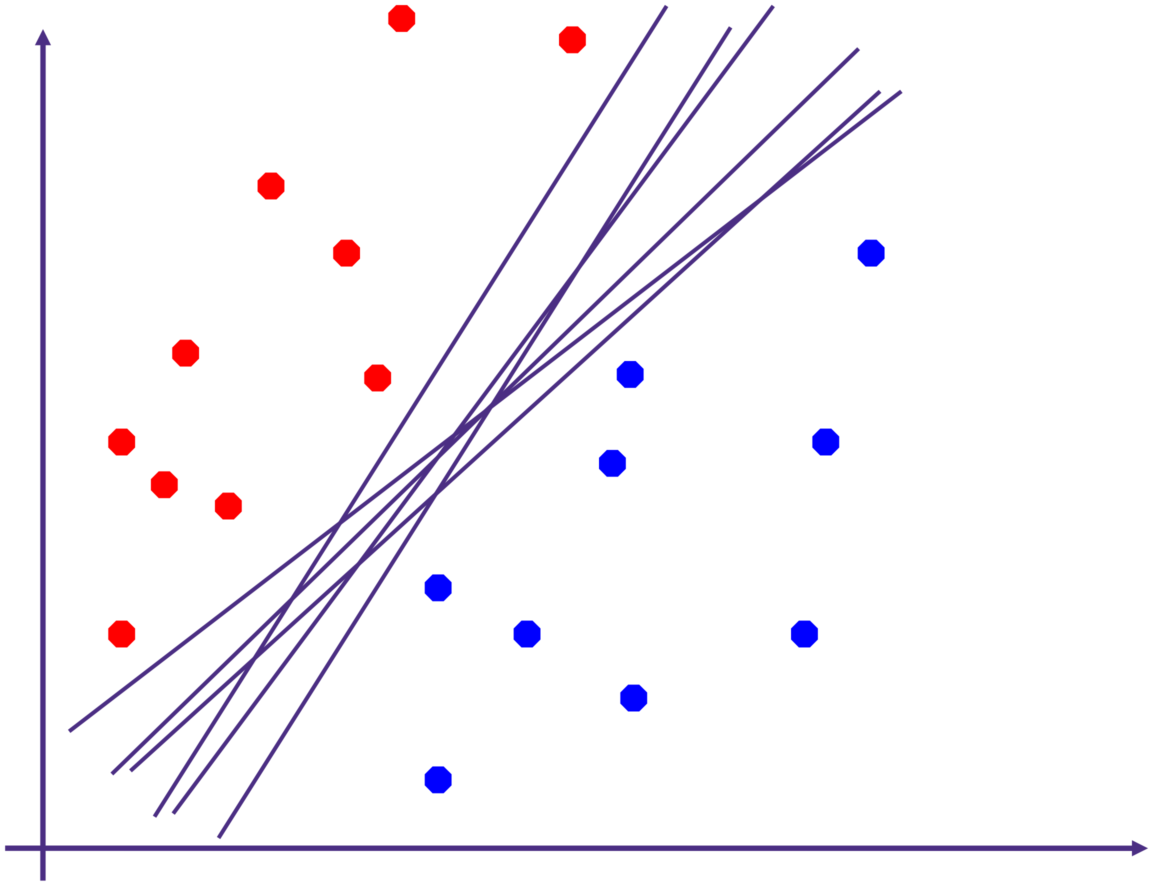
- Binary classification can be viewed as the task of separating classes in feature space



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

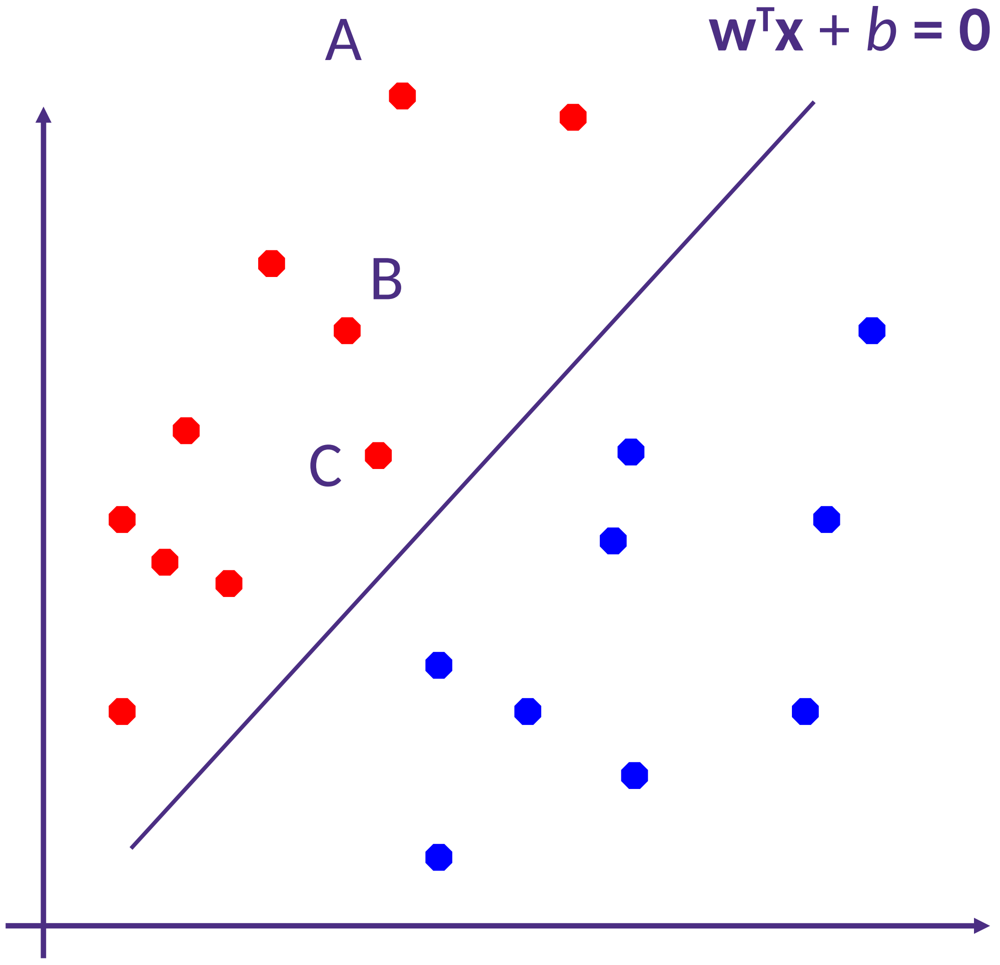
Wide Margin Intuition

> Which of the linear decision boundary is optimal?



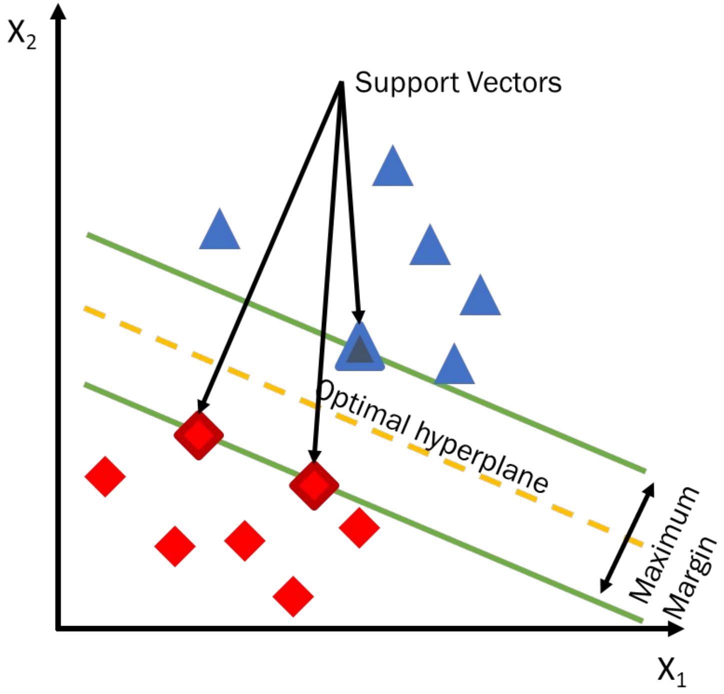
Intuition of Margin

- Consider points A, B, and C.
- We are quite confident in our prediction for A because it's far from the decision boundary.
- In contrast, we are not so confident in our prediction for C because a slight change in the decision boundary may flip the decision.



Goal of Learning

Given a training set, we would like to make all our predictions correct and confident! This can be captured by the concept of margin.



Functional Margin

$$\hat{\gamma}^i = y^i(\mathbf{w} \cdot \mathbf{x}^i + b)$$

Note that $\hat{\gamma}^i > 0$ if classified correctly

> We define this as the functional margin of the linear classifier with respect to the training example (x^i, y^i)

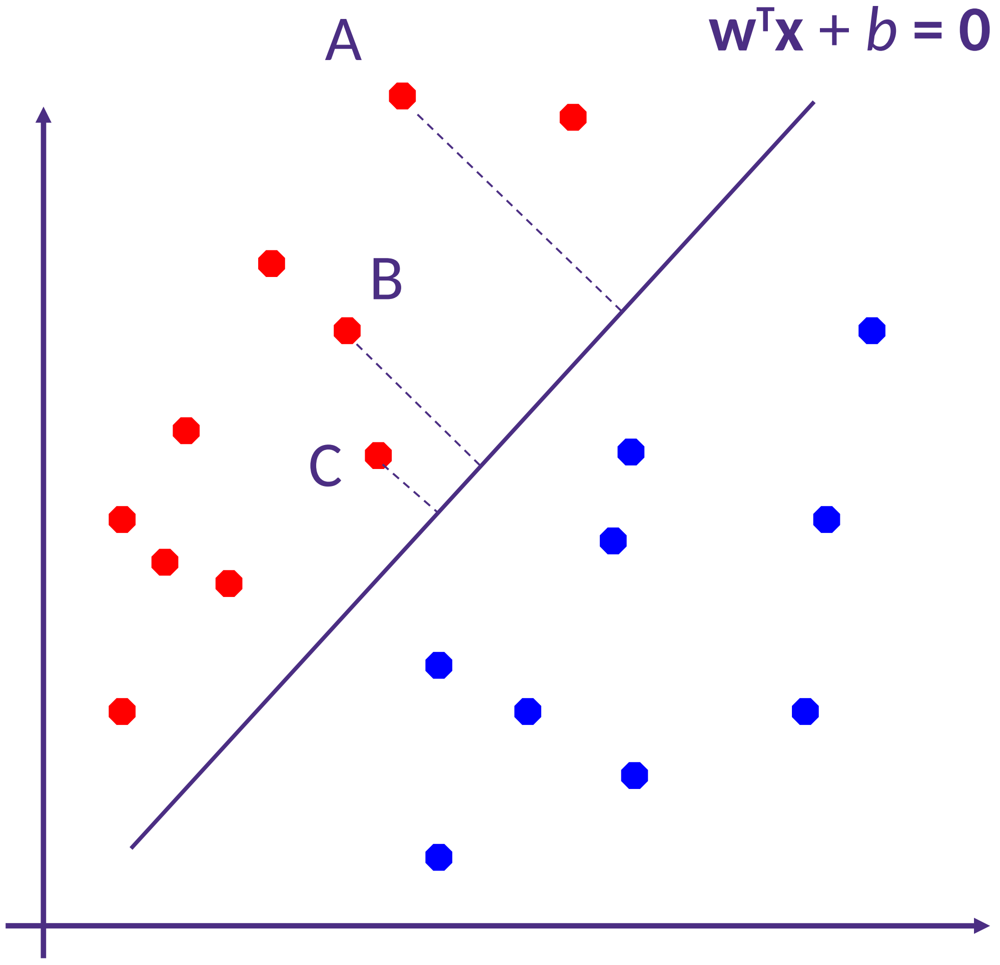
> The large the value, the better?

> What if we rescale (W, b) by a factor of α?

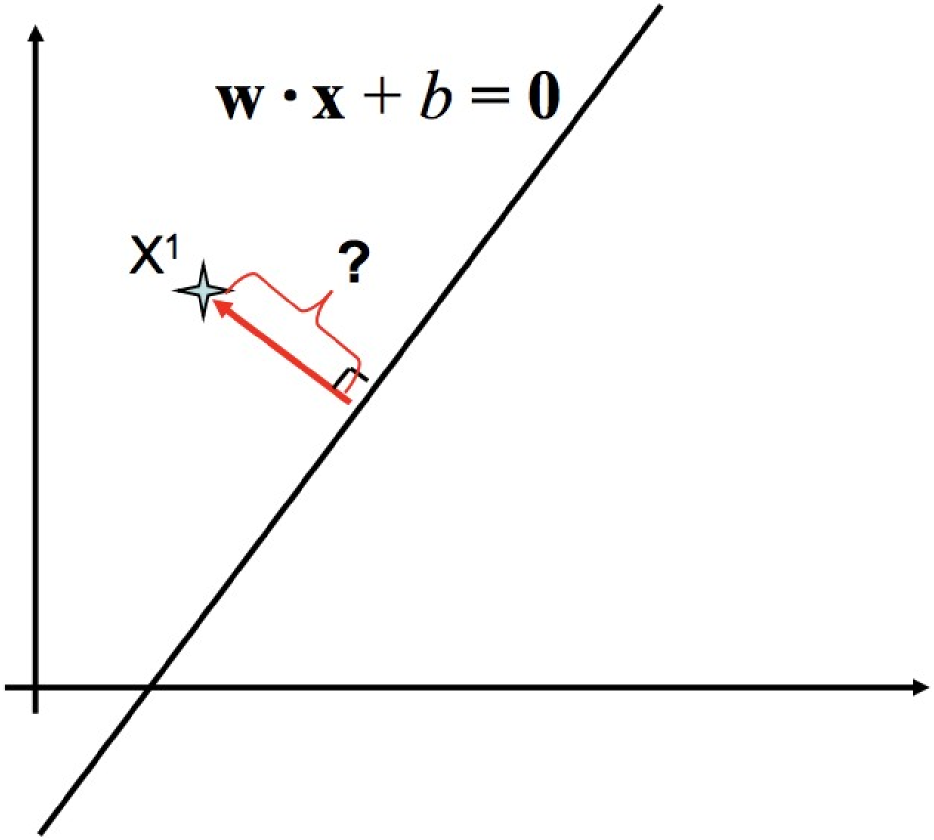
- Decision boundary remains the same.
- Yet, functional margin gets multiplied by α.

Geometric Margin

- What we really want is the distances between the examples and the decision boundary to be large.
- This distance is called **geometric margin**.
- How do we compute the geometric margin of a data point with respect to a particular line parameterized by W and b.



How to calculate the distance of a point to a line?



$$\boxed{\frac{w \cdot x^1 + b}{\|w\|}}$$

$$\|\boldsymbol{z}\| := \sqrt{|z_1|^2 + \cdots + |z_n|^2}$$

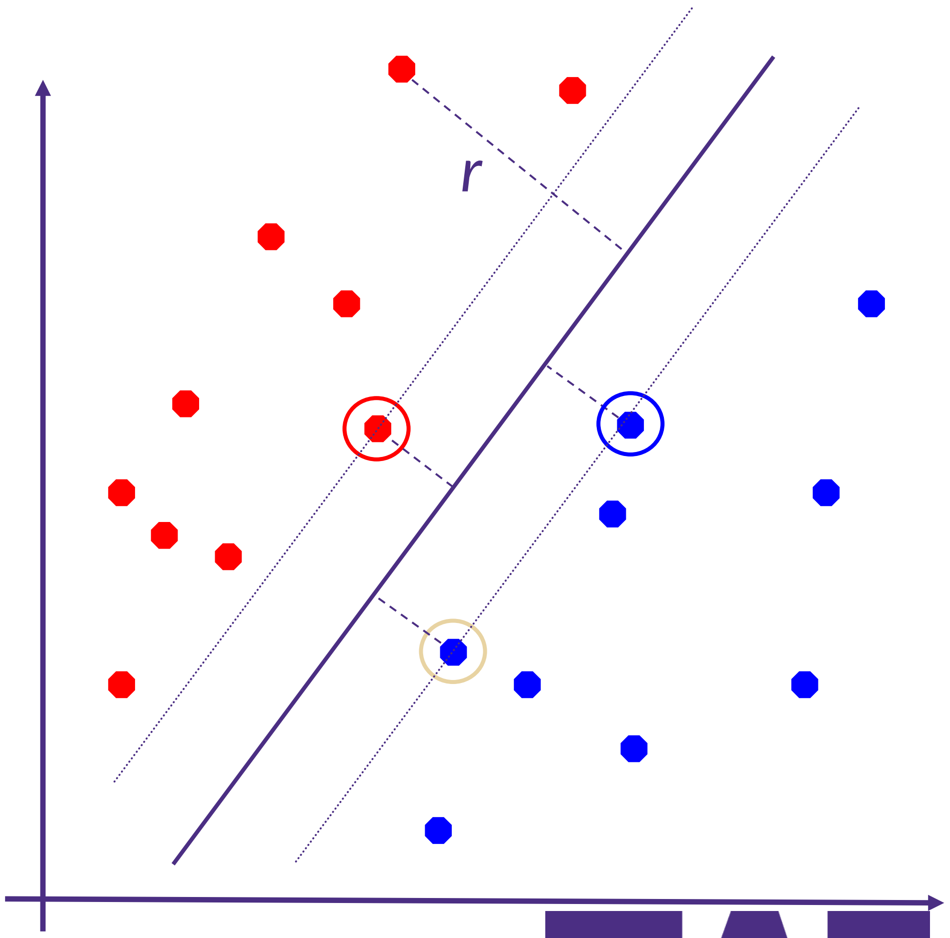
Maximize the Geometric Margin

- The geometric margin of x^i with respect to (W, b) is the distance from x^i to the decision surface.

- This distance can be computed as

$$\gamma^i = \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|}$$

- Examples closest to the hyperplane are ***support vectors***.

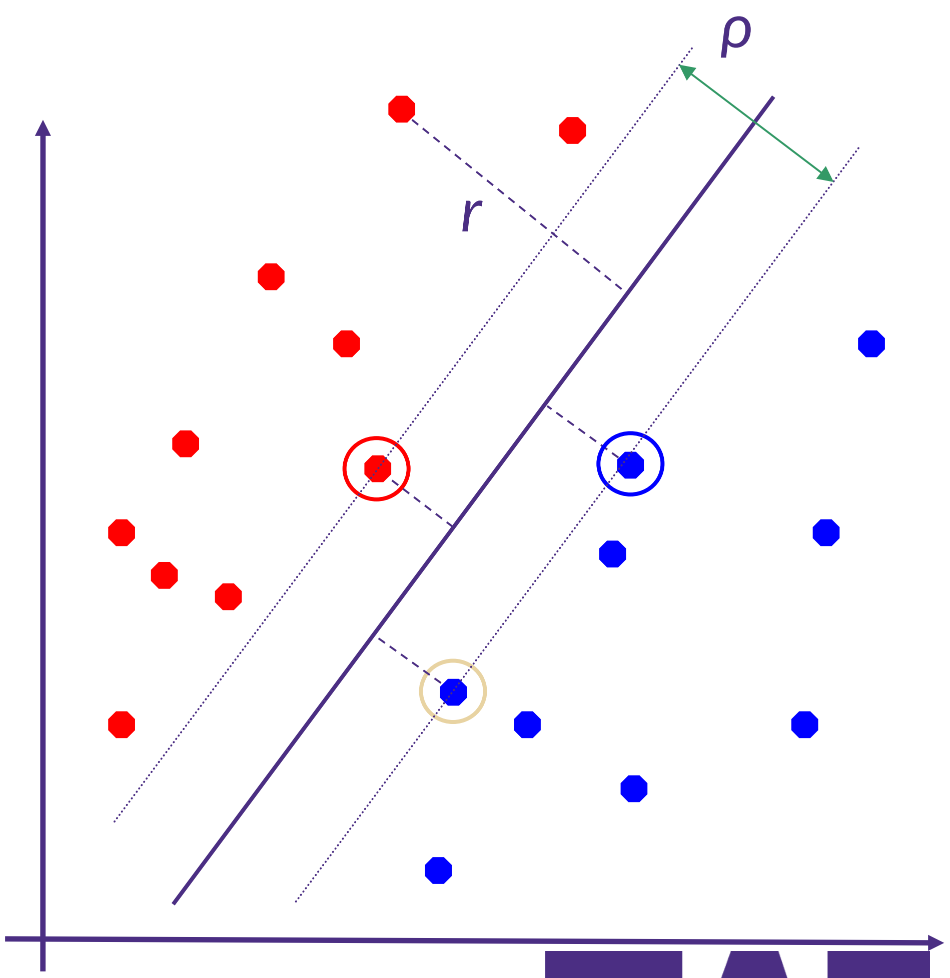


Maximize the Geometric Margin

- Given a training set, the geometric margin of the classifier with respect to this training set is:

$$\gamma = \min_{i=1\cdots N} \gamma^{(i)}$$

- ***Margin*** ρ of the separator is the distance between support vectors.



Geometric Margin vs. Functional Margin

> The **Geometric Margin** of a training example *EQUALS* the **Functional Margin** normalized by the magnitude magnitude of w.

$$\gamma^i = \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|}$$

Functional margin

Quick Recap

> We want to find a linear decision boundary whose margin is the largest.

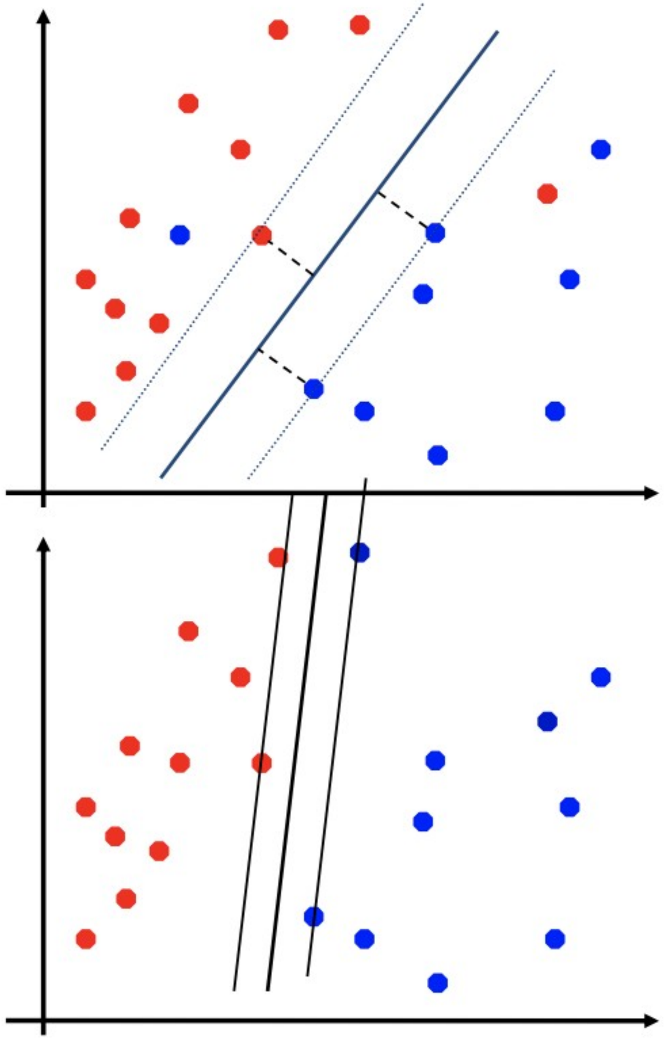
> We know how to measure the margin of a linear decision boundary.

> We have a new learning objective. Given a linearly separable training set, we would like to find a linear classifier with maximum margin.

Non-separable Data and Noise

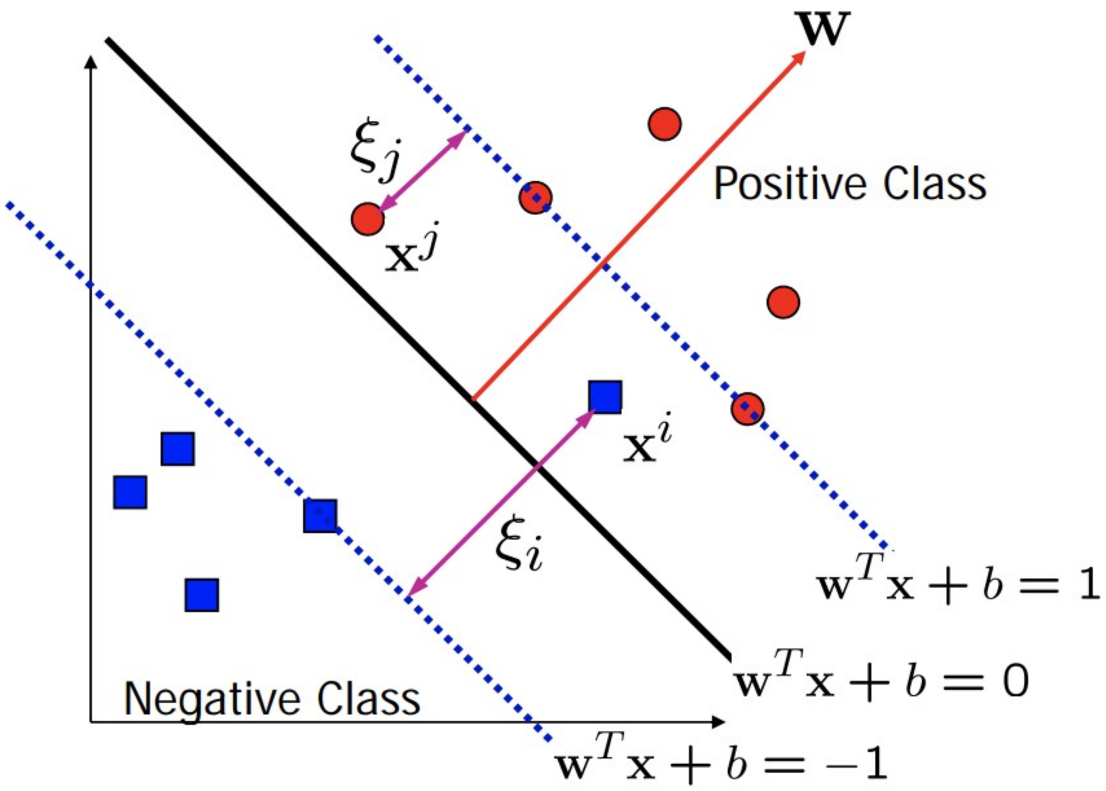
- What if the data is not linearly separable?

- We may have noise in data, and maximum margin classifier is not robust to noise!



Soft Margin

- Allow functional margins to be less than 1.
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft margin*.



Originally functional margins need to satisfy:

$$y^i(w \cdot x^i + b) \geq 1$$

Now we allow it to be less than 1:

$$y^i(w \cdot x^i + b) \geq 1 - \xi_i$$

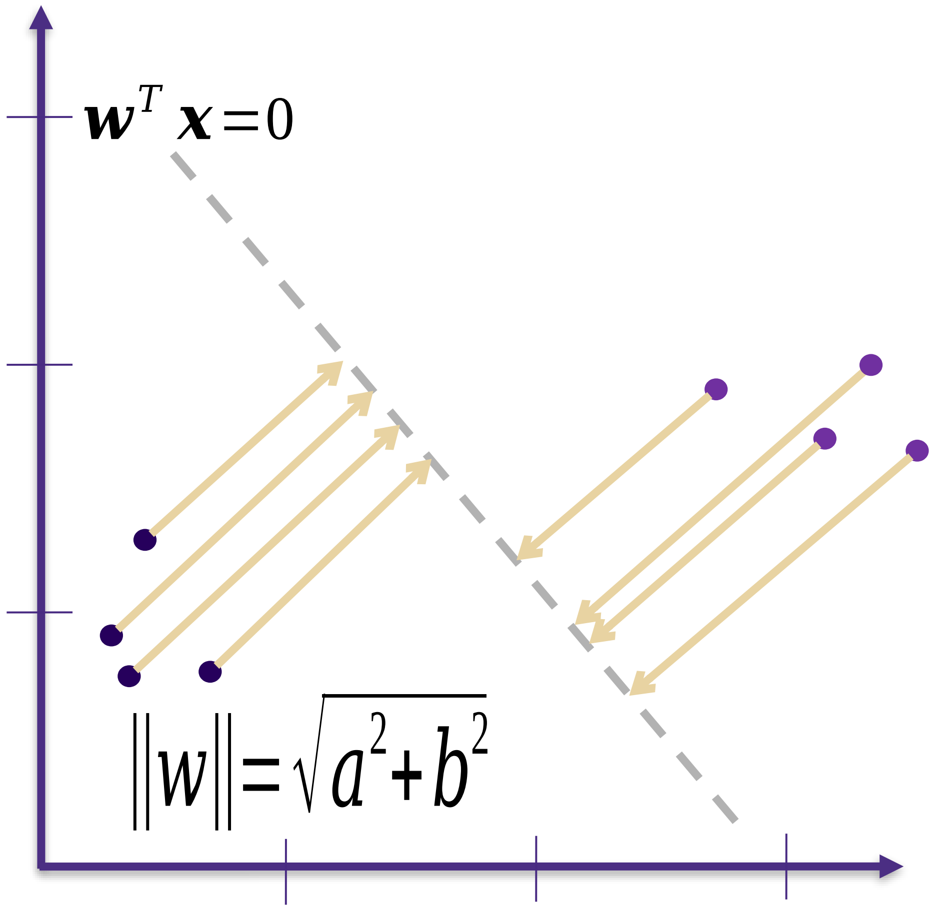
The objective ftn also change to:

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2 + c\sum_{i=1}^{N} \xi_i$$

Find an Optimum Decision Boundary

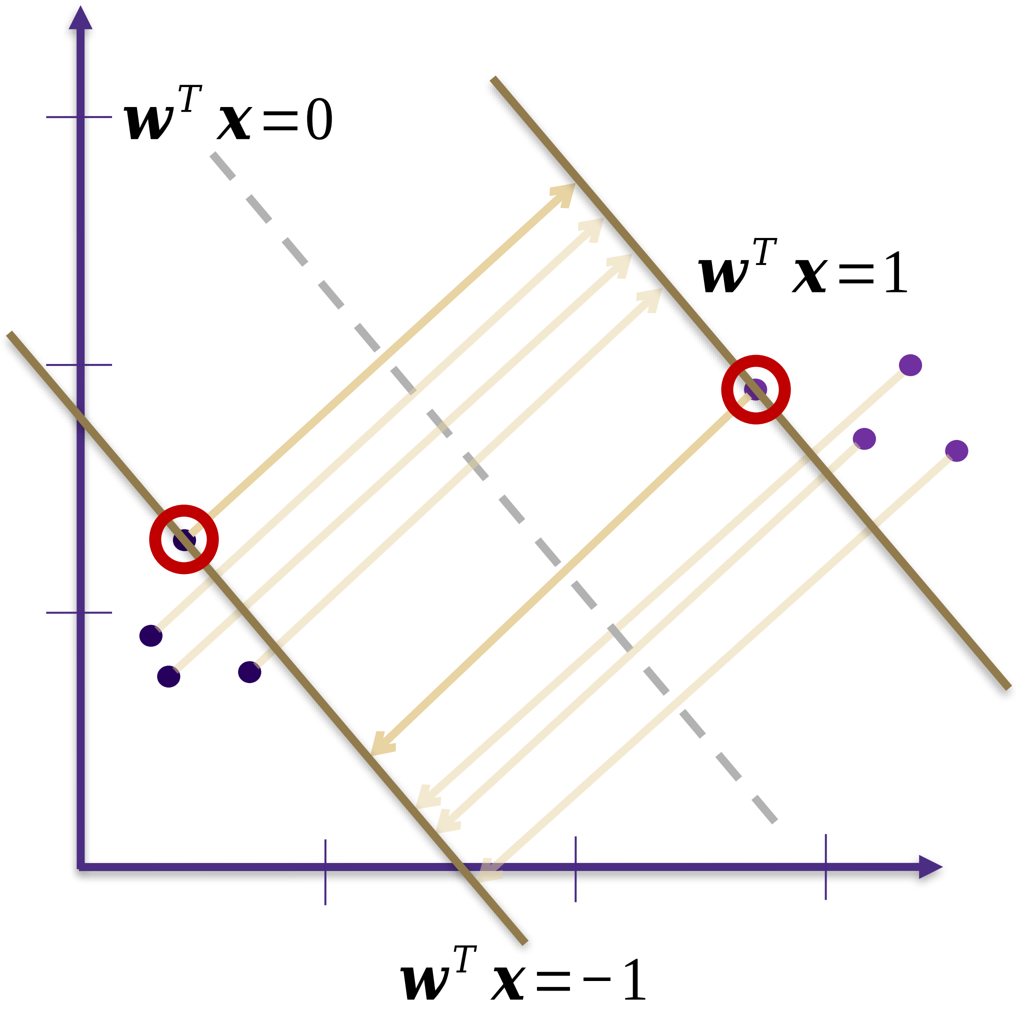
- Decision boundaries classify all the data points correctly
- Several hyperplanes may satisfy this requirement
- For SVMs, we are looking for the Euclidean dot product calculated as follows:

$$\sum_{t=1}^{d} w_1 x_1 = w^T x$$

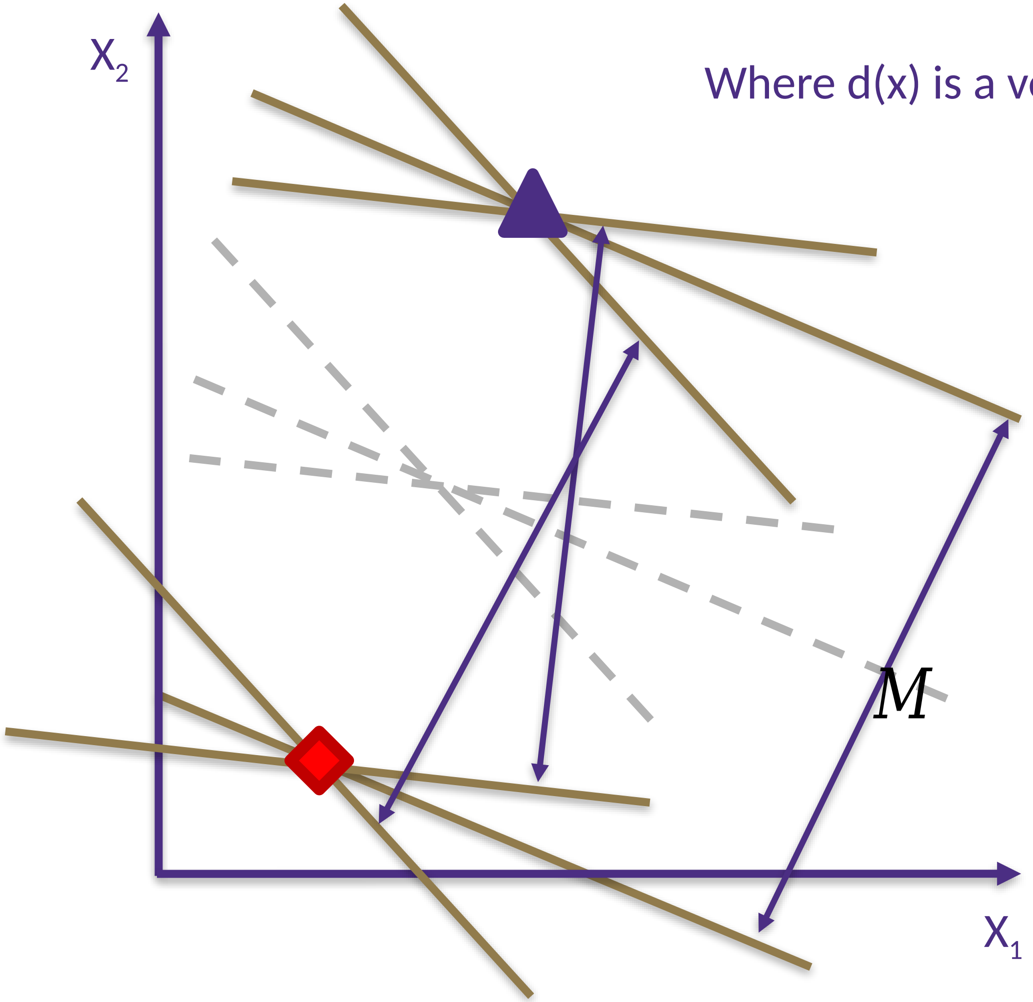


Find the Maximum Margin

- Calculate the distances from each data vector
- Maximum distance between any two points and denote that as 2
- We define as midway distance between the two closest points
- Therefore, the distance between the margins are two parallel vectors to the hyperplane 2 distance apart



Find the Optimal Hyperplane



Where d(x) is a vector point in the set D and w is weight

The optimal hyperplane is the orthogonal projection of a perpendicular line that is the maximum distance from **<u>all</u>** of the vectors

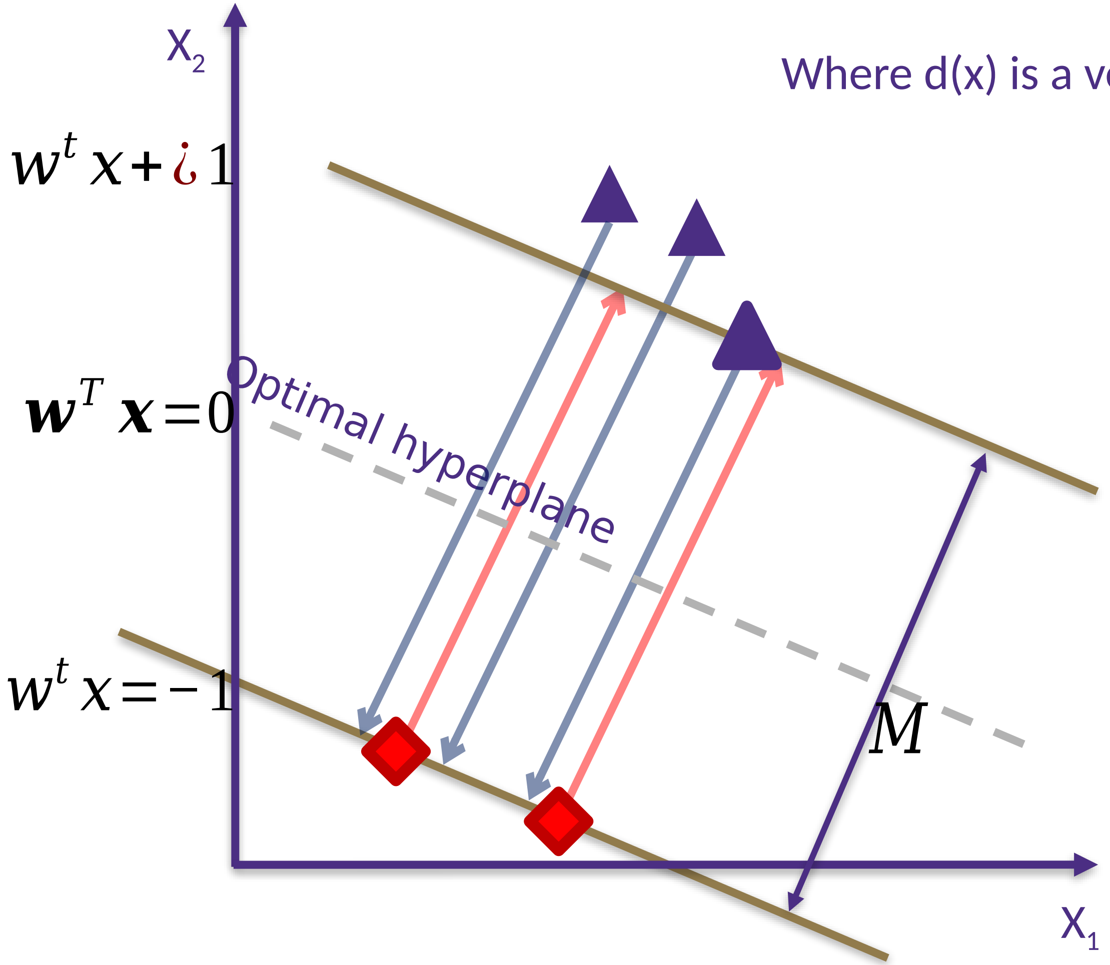
Goal:

Where is the length of the vector of weights

Find the Optimal Hyperplane

At each new datapoint

1. Select two hyperplanes which separate the data point with no points between them
2. maximize their distance (the margin)
3. Half the distance is the optimal hyperplane



Where d(x) is a vector point in the set D and w is weight

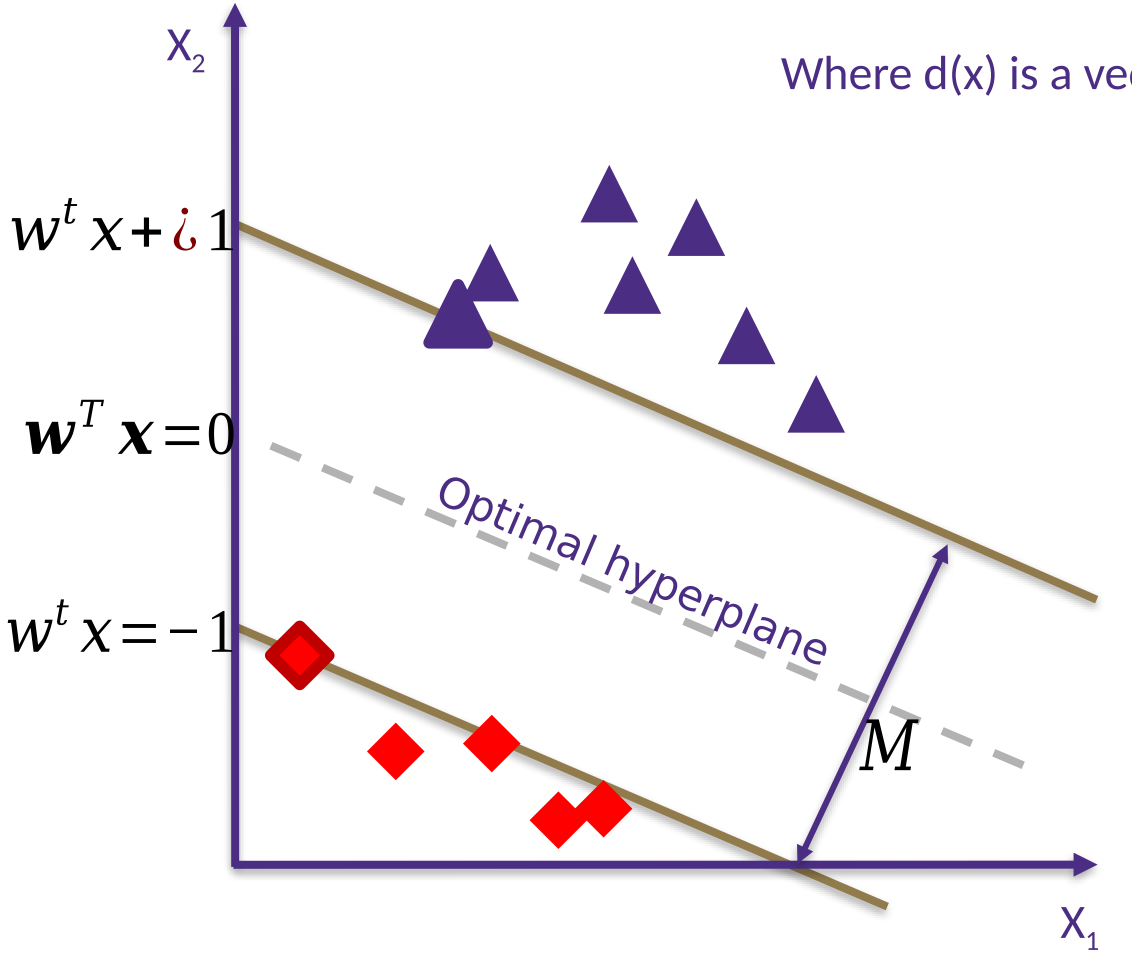
Goal:

Where is the length of the vector of weights

Find the Optimal Hyperplane

At each new datapoint

1. Select two hyperplanes which separate the datapoint with no points between them
2. maximize their distance (the margin)
3. Half the distance is the optimal hyperplane



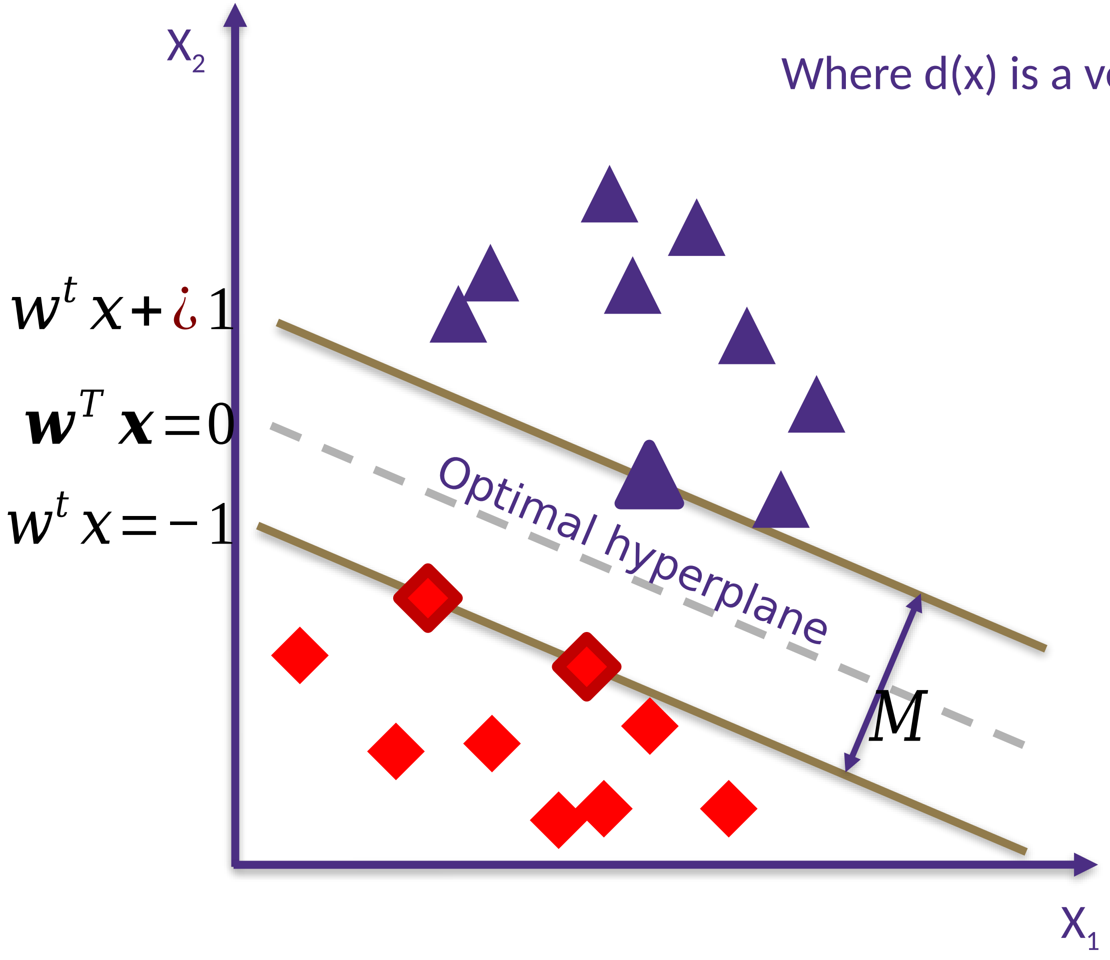
Where d(x) is a vector point in the set D and w is weight

Goal:

Where is the length of the vector of weights

Find the Optimal Hyperplane

Where d(x) is a vector point in the set D and w is weight

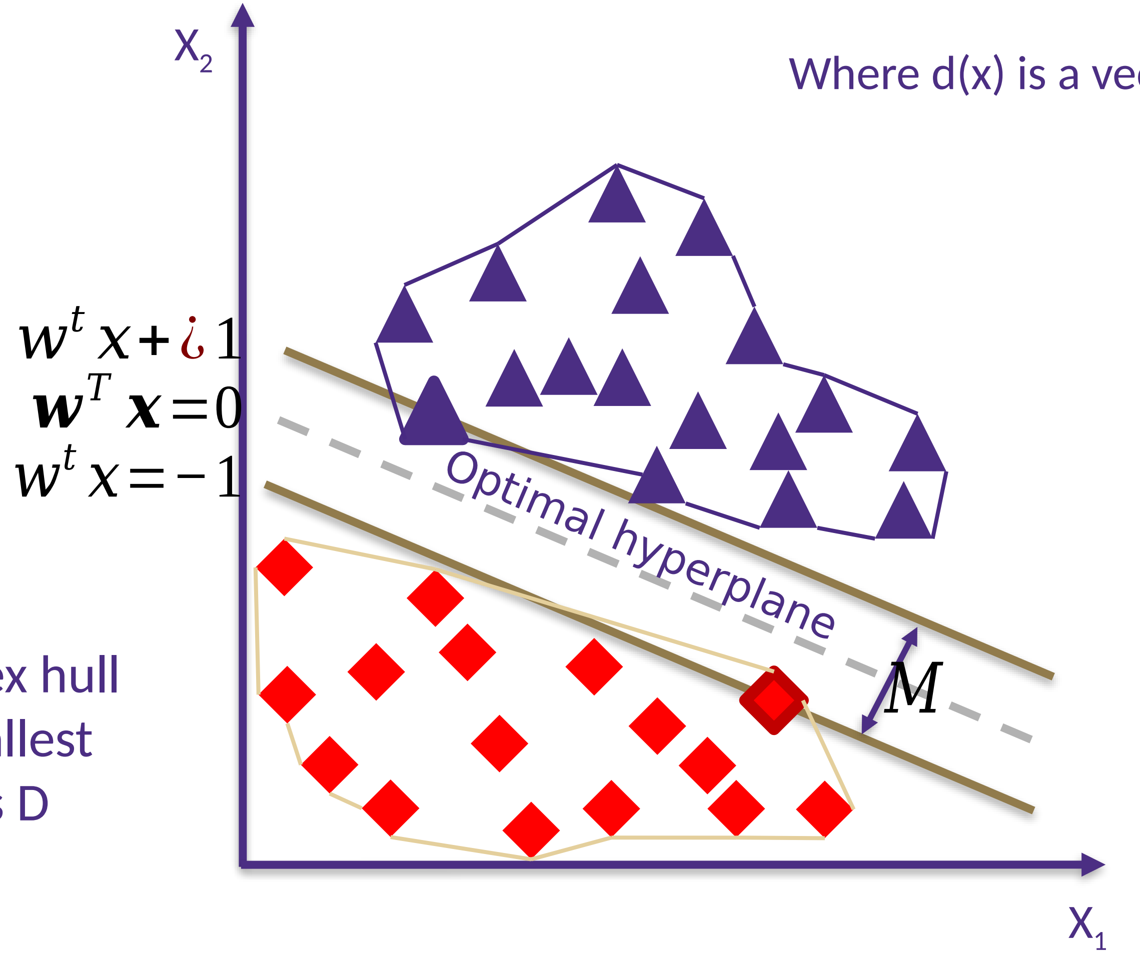


Goal:

Where is the length of the vector of weights

Find the Optimal Hyperplane

Where d(x) is a vector point in the set D and w is weight



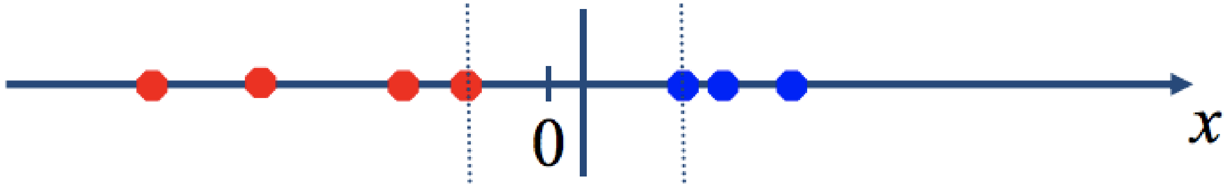
SVMs identify the convex hull of each group… the smallest convex set that contains D

Goal:

Where is the length of the vector of weights

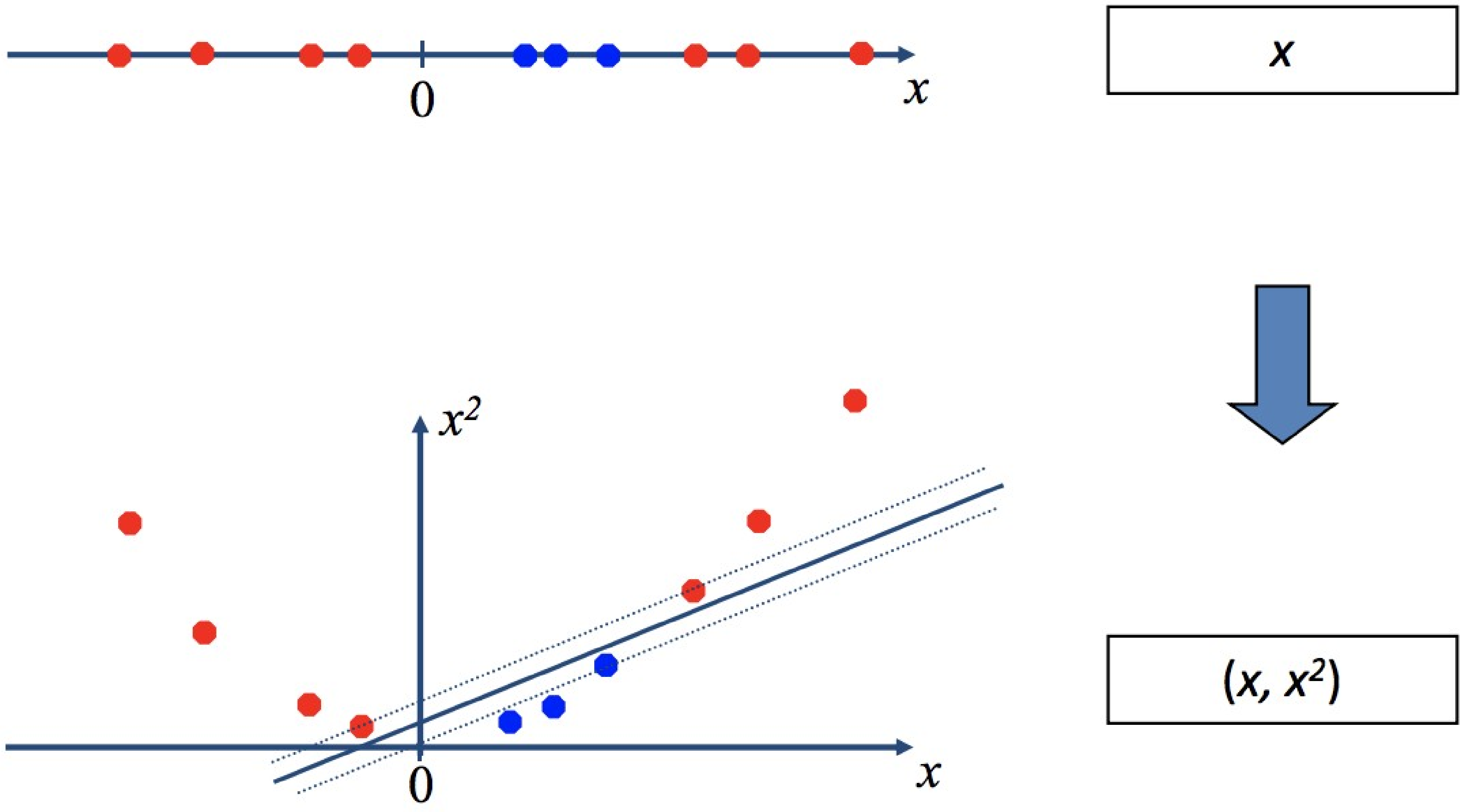
Non-linear SVM

> Datasets that are linearly separable with some noise work out great.



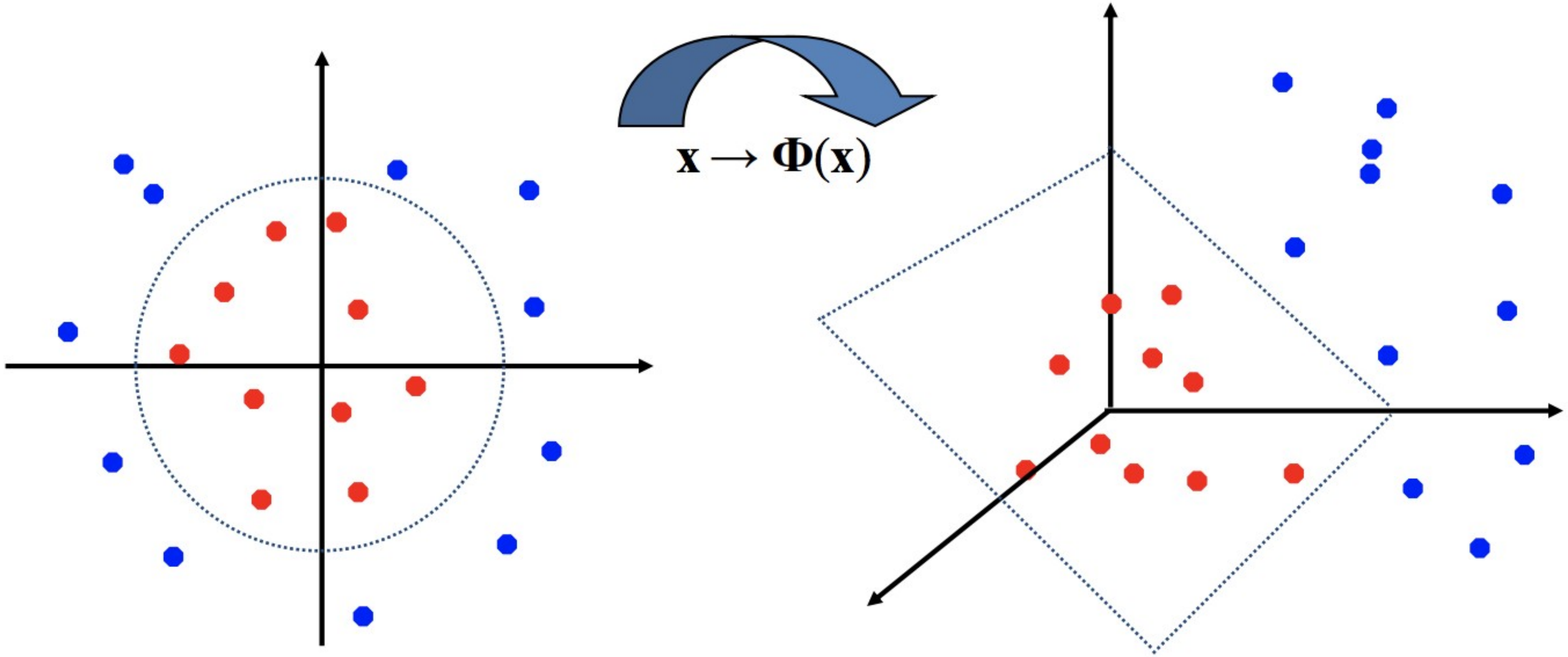
> But what are we going to do if the dataset is just too hard?

Map data into a higher dimensional space

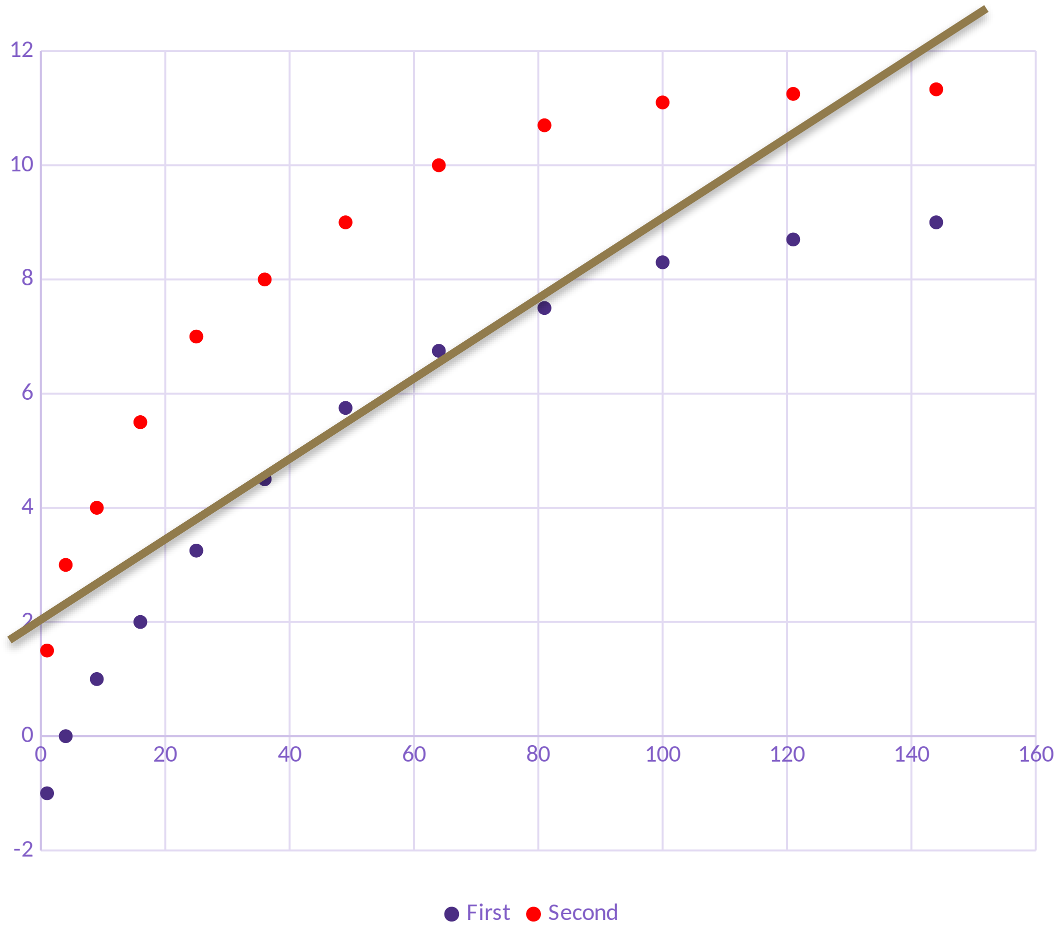


Non-linear SVMs: Feature Spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space such that the train data is linearly separable.

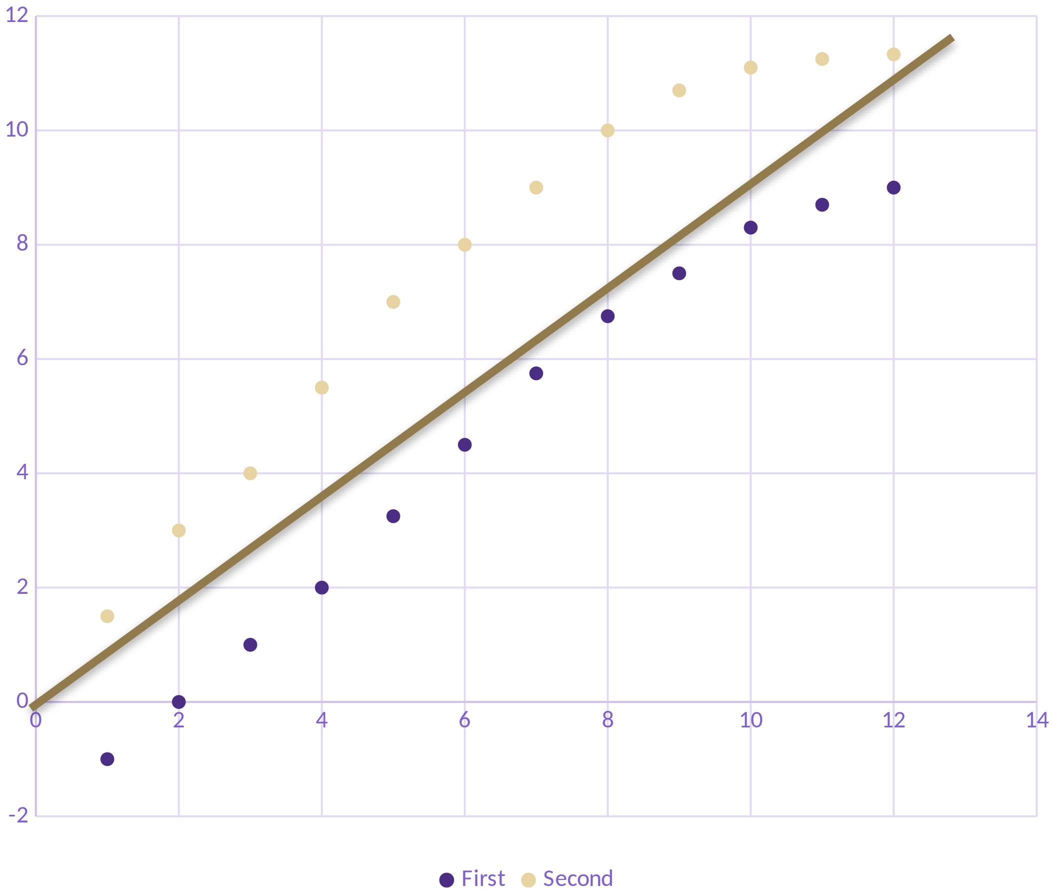


Simple Non-linear Example



	First	Second
1	-1	1.5
4	0	3
9	1	4
16	2	5.5
25	3.3	7
36	4.5	8
49	5.8	9
64	6.8	10
81	7.5	10.7
100	8.3	11.1
121	8.7	11.25
144	9	11.33

Simple Non-Linear Example

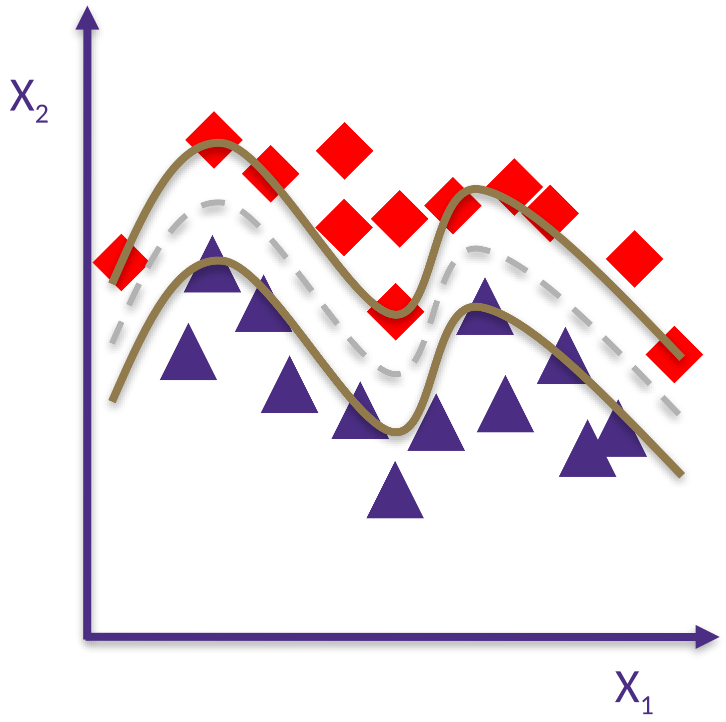


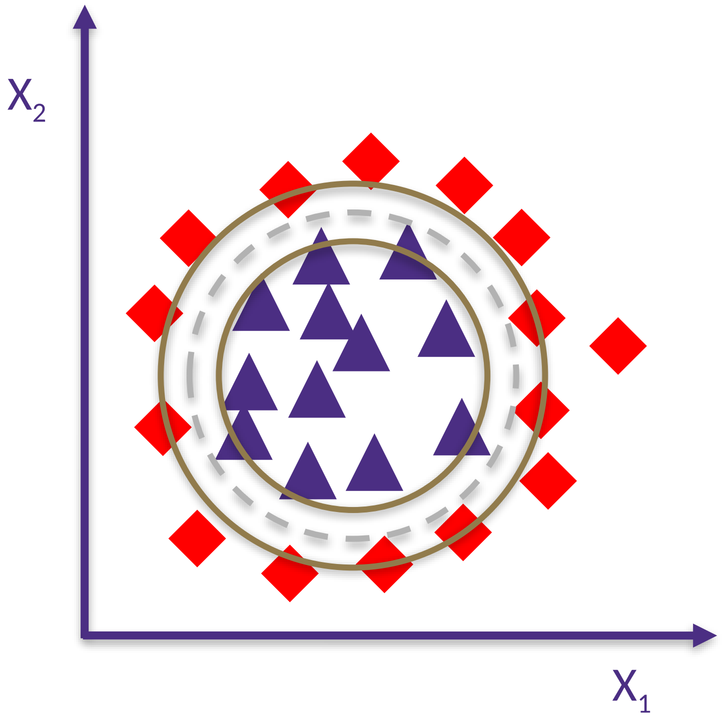
	First	Second
1	-1	1.5
2	0	3
3	1	4
4	2	5.5
5	3.3	7
6	4.5	8
7	5.8	9
8	6.8	10
9	7.5	10.7
10	8.3	11.1
11	8.7	11.25
12	9	11.33

Nonlinear Support Vector Machines

- Good for smaller data sets
- No assumption of probability distribution
- Converts 2d to multidimensional space
- Common methods:
  - Polynomial kernel
  - Radial Basis Function
  - Sigmoid kernel
  - Gaussian kernel
  - Exponential kernel
  - Among others… *
- Choosing the correct kernel is a non-trivial task

Other Non-linearly Separability Data Sets





Separation in non-linear data sets is accomplished using a kernel function, of which there are several

Non-Linear SVM Summary

- Map the input space to a high dimensional feature space and learn a linear decision boundary in the feature space
- The decision boundary will be nonlinear in the original input space
- Many possible choices of kernel functions
  - How to choose? Most frequently used method: cross-validation

Kernel Trick: Advantages and Caveats

- Useful in high-dimensional spaces – can work even when the number of dimensions is greater than examples (Caveat: predictive capability may be poor)
- Features are non-parametric (Caveat: computational cost)
  - Not constricted to a “distribution”
  - In theory, infinite, thus are “assumption free” model
  - Reduced chances of the ‘curse of dimensionality’
- Kernel functions can be added together (be ensembles) to create even more complex hyperplanes (Caveat: computational cost)
- Give a highly optimal hyperplane (Caveat: no probability functions)

Parameters in Support Vector Classification

sklearn.svm.svc

- Important Hyperparameters:
  - **Kernel** – can be linear, **rbf**, poly, sigmoid,
  - **C** (cost) hyperparameter – higher value adds a higher cost for misclassifications (hard margin) and lower value allows for more leeway (soft margin) – softer margin allow for more generalizability and lower sensitivity to noise. Default is **1.0**
  - **Gamma** – hyperparameter for rbf, poly and sigmoid kernels to configure model sensitivity to feature differences. It defines the distance of influence for a single training example. Low values meaning ‘far’ and high values meaning ‘close’. Default is **1/n** (each input vector has a 1/n influence)
  - **Degree** – hyperparameter for polynomial/exponential kernels, specifies the largest possible exponent. Default is

In general, cost and gamma are way to tune the model for softer or harder margins

Notebook Time

Appendix

Maximum Margin Classifier

> This can be represented as a constrained optimization problem.

$$\max_{\mathbf{w},b} \gamma$$
$$\text{subject to}: \; y^{(i)} \frac{(\mathbf{w}\cdot\mathbf{x}^{(i)}+b)}{\|\mathbf{w}\|} \geq \gamma, \quad i=1,\cdots,N$$

> This optimization problem is in a nasty form, so we need to do some rewriting.

> Let γ' = γ · ||w|| , we can rewrite this as

$$\max_{\mathbf{w},b} \frac{\gamma'}{\|\mathbf{w}\|}$$
$$\text{subject to}: \; y^{i}(\mathbf{w}\cdot\mathbf{x}^{i}+b) \geq \gamma', \quad i=1,\cdots,N$$

Maximum Margin Classifier con't

- Note that we can arbitrarily rescale w and b to make the functional margin γ' large or small.
- So we can rescale them such that γ' = 1.

$$\max_{\mathbf{w},b} \frac{\gamma'}{\|\mathbf{w}\|}$$

$$\text{subject to}: \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \ge \gamma', \quad i = 1,\cdots,N$$

$$\Downarrow$$

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \quad (\text{or equivalently } \min_{\mathbf{w},b} \|\mathbf{w}\|^2)$$

$$\text{subject to}: \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \ge 1, \quad i = 1,\cdots,N$$

Solve the Optimization Problem

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to: } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1, \quad i = 1,\cdots,N$$

- This is a **quadratic optimization** problem with **linear inequality constraints**.
- This is a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a dual problem where a Lagrange multiplier α_i is associated with every inequality constraint in the primal (original) problem.

Solution

> We can not give you a closed form solution that you can directly plugin in and compute for an arbitrary data sets.

> The solution can always be written in the following form

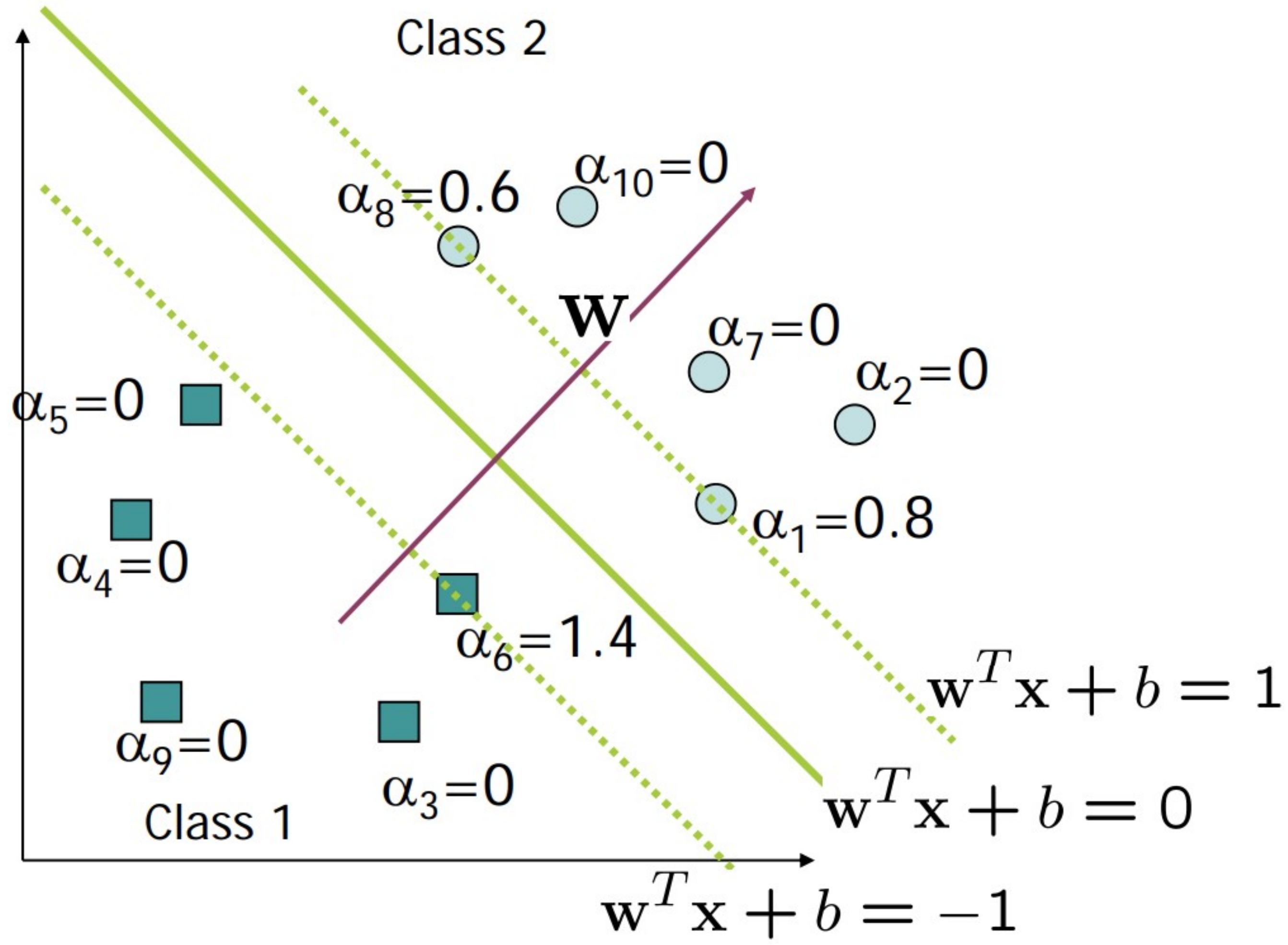
$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

Solution Cont.

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

- The weight vector is a linear combination of all the training examples.
- Importantly, many of the α are zeros. These points that have non-zero α are the **support vectors**.
- Solve the optimization problem involved computing the inner products $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$ between all training points.

A Geometrical Interpretation

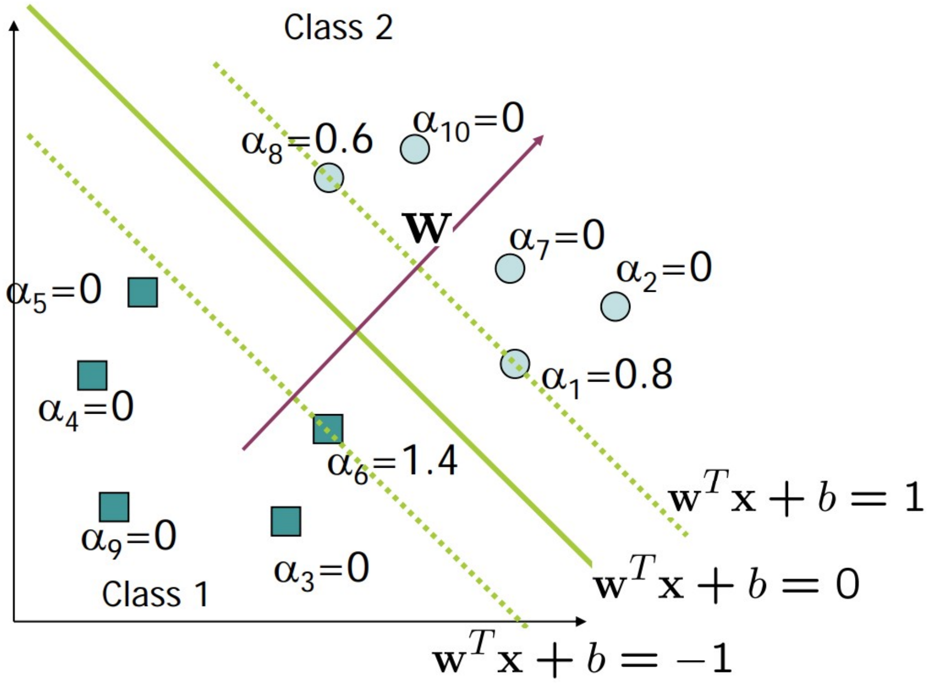


Geometric Interpretation

- $W^T x + b = 0$ gives the decision boundary.
- $W^T x + b = 1$ positive support vectors lie on this line.
- $W^T x + b = -1$ negative support vectors lie on this line.

- The decision boundary can be thought of now as a tube of certain width where no points can be inside the tube.

- Implies that only **support vectors** matter; other training examples are ignorable.



Summarization So Far

> We defined margin (functional, geometric)

> We demonstrated that we prefer to have linear classifiers with large geometric margin.

> We formulated the problem of finding the maximum margin linear classifier as a quadratic optimization problem.

> This problem can be solved using efficient QP algorithms that are available.

> The solution are very nicely formed.

Soft Margin Maximization

> Introduce slack variables ξ_i to allow some examples to have functional margins smaller than 1.

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2$$
$$\text{subject to}: \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1, \quad i = 1, \cdots, N$$

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2 + c\sum_{i=1}^{N} \xi_i$$
$$\text{subject to}: \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, N$$
$$\xi_i \geq 0, \quad i = 1, \cdots, N$$

Hyper-parameter C

> Effect of parameter c:
  - Controls the tradeoff between maximizing the margin and fitting the training examples
  - Large c: slack variables incur large penalty, so the optimal solution will try to avoid them
  - Small c: small cost for slack variables, we can sacrifice a few training examples to ensure that the classifier margin is large

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2 + c\sum_{i=1}^{N} \xi_i$$

$$\text{subject to: } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, N$$

$$\xi_i \geq 0, \quad i = 1, \cdots, N$$

Solution to SVM with Soft Margin

$$\boxed{w = \sum_{i=1}^{N} \alpha_i y^i x^i, \quad \textbf{s.t.} \sum_{i=1}^{N} \alpha_i y^i = 0}$$

No soft margin

$$\boxed{w = \sum_{i=1}^{N} \alpha_i y^i x^i, \quad \textbf{s.t.} \sum_{i=1}^{N} \alpha_i y^i = 0 \textbf{ and } 0 \le \alpha_i \le c}$$

With soft margin

> c controls the tradeoff between maximizing margin and fitting training data.
> It's effect is to put a **box constraint** on α (the weights of the support vectors).
> It limits the influence of individual support vectors (maybe outliers).
> In practice, c can be set by cross-validation.

Make predictions with SVM

For classifying with a new input **z**

Compute

$$w \cdot z + b = \left(\sum_{j=1}^{s} \alpha_{t_j} y^{t_j} x^{t_j}\right) \cdot z + b = \sum_{j=1}^{s} \alpha_{t_j} y^{t_j} (x^{t_j} \cdot z) + b$$

classify **z** as **+** if positive, and - otherwise

Note: **w** need not be formed explicitly, we can classify z by taking inner products with the support vectors

Quadratic Feature Space

Assume m input dimensions

$$\mathbf{x} = (x_1, x_2, \cdots, x_m)$$

Number of quadratic terms:

$1+m+m+m(m-1)/2 \approx m^2$

The number of dimensions increase rapidly!

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

You may be wondering about the $\sqrt{2}$
At least they won't hurt anything!

You will find out why they are there soon!

Dot product in quadratic feature space

$$\Phi(\mathrm{a})\cdot\Phi(\mathrm{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

$$1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

$$\mathbf{\Phi(a)} \bullet \mathbf{\Phi(b)} =$$
$$1 + 2\sum_{i=1}^{m} a_ib_i + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Now let's just look at another interesting function of (a·b):

$$\begin{aligned}
&(a\cdot b+1)^2 \\
&= (a\cdot b)^2 + 2(a\cdot b) + 1 \\
&= (\sum_{i=1}^{m} a_ib_i)^2 + 2\sum_{i=1}^{m} a_ib_i + 1 \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ia_jb_ib_j + 2\sum_{i=1}^{m} a_ib_i + 1 \\
&= \sum_{i=1}^{m} a_i^2b_i^2 + 2\sum_{i=1}^{m}\sum_{j=i+1}^{m} a_ia_jb_ib_j + 2\sum_{i=1}^{m} a_ib_i + 1
\end{aligned}$$

They are the same! And the later only takes O(m) to compute!

Kernel Functions

> If every data point is mapped into high-dimensional space via some transformation $x \rightarrow \varphi(x)$, the inner product that we need to compute for classifying a point x becomes:

$$<\phi(\mathbf{x}^i) \cdot \phi(\mathbf{x})> \text{ for all support vectors } \boldsymbol{x}^i$$

> A **kernel function** is a function that is equivalent to an inner product in some feature space.

$$\boldsymbol{k}(a,b) = <\phi(\mathbf{a}) \cdot \phi(\mathbf{b})>$$

> Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

More Kernel Functions

Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.