PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Machine Learning Techniques

DATASCI 420

Lesson 07 Support Vector Machines

Support Vector Machine--History

- The mathematical idea of an SVM has been around since the 60’s (V. Vapnik, 1963) the first robust application was published in 1992 by Boser, Guyon and Vapnik
- SVMs are considered one of the best “off the shelf” machine learning algorithms
  - They are less likely to overfit the data
  - Can be used for both classification and regression
  - Applications range of applications
- They attempt to “regulate” the hypothesis space to ensure maximum accuracy

Applications in Literature

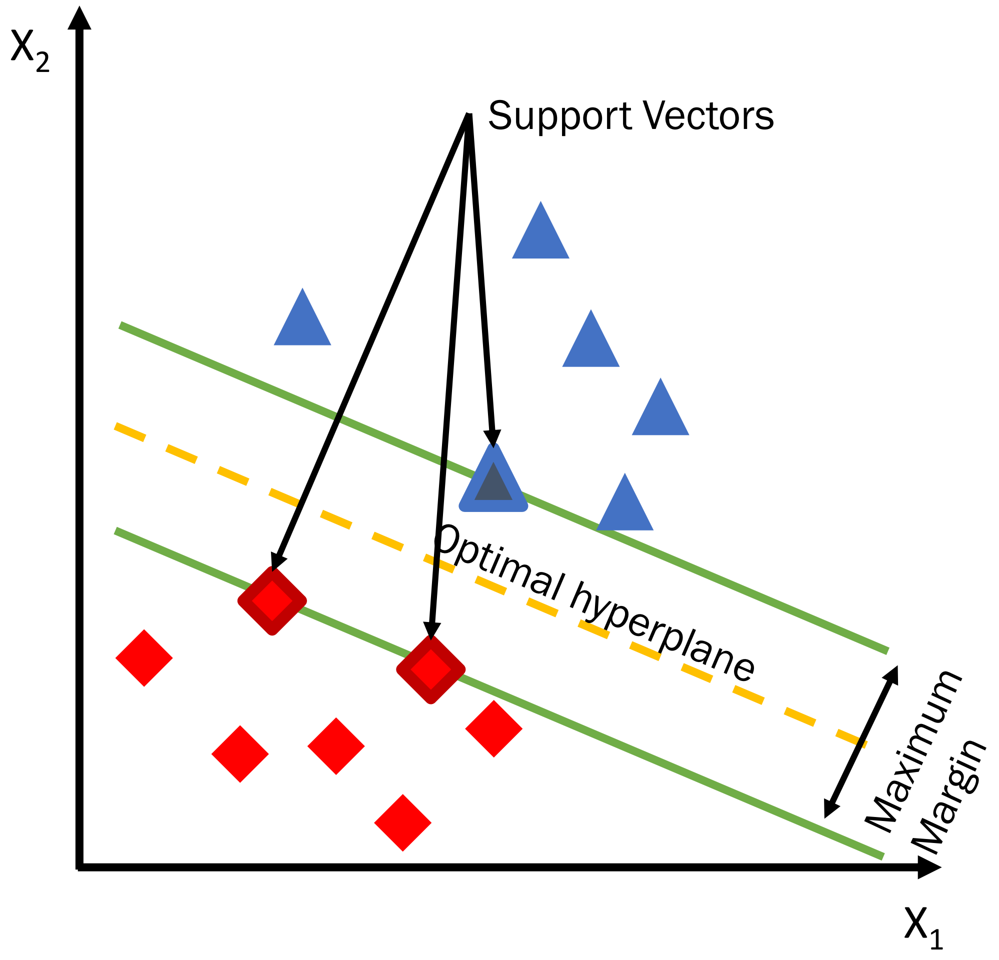
- Medical imaging classification
- Face recognition
- Emotion classification
- Air quality analysis
- Page ranking algorithms in online search
- Time series prediction
- Outlier identification (potentially good as a filter mechanism for other types of machine learning methods)

SVM in a Nutshell

- Similar to linear regression these algorithms are used to find a hyperplane that separates data points into two classes
- Are robust binary classifiers
- The model is a representation of these points in “space” which is why we consider them **vectors**—they have a value and a location
- They are divided by a clear gap (**margin**)—as wide as possible given all known points—known as a **hyperplane**
- The **margin** is the space *between* the closest individual data points (**support vectors**).

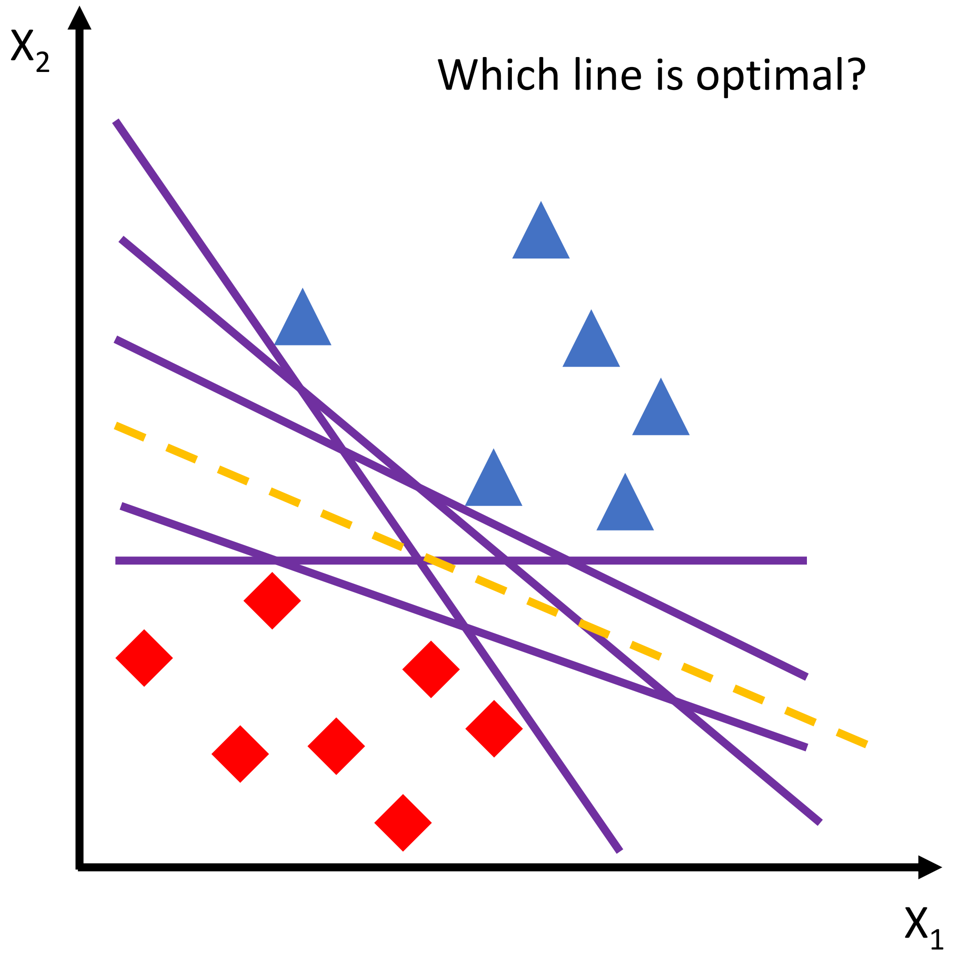
SVMs in a Nutshell…

- 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



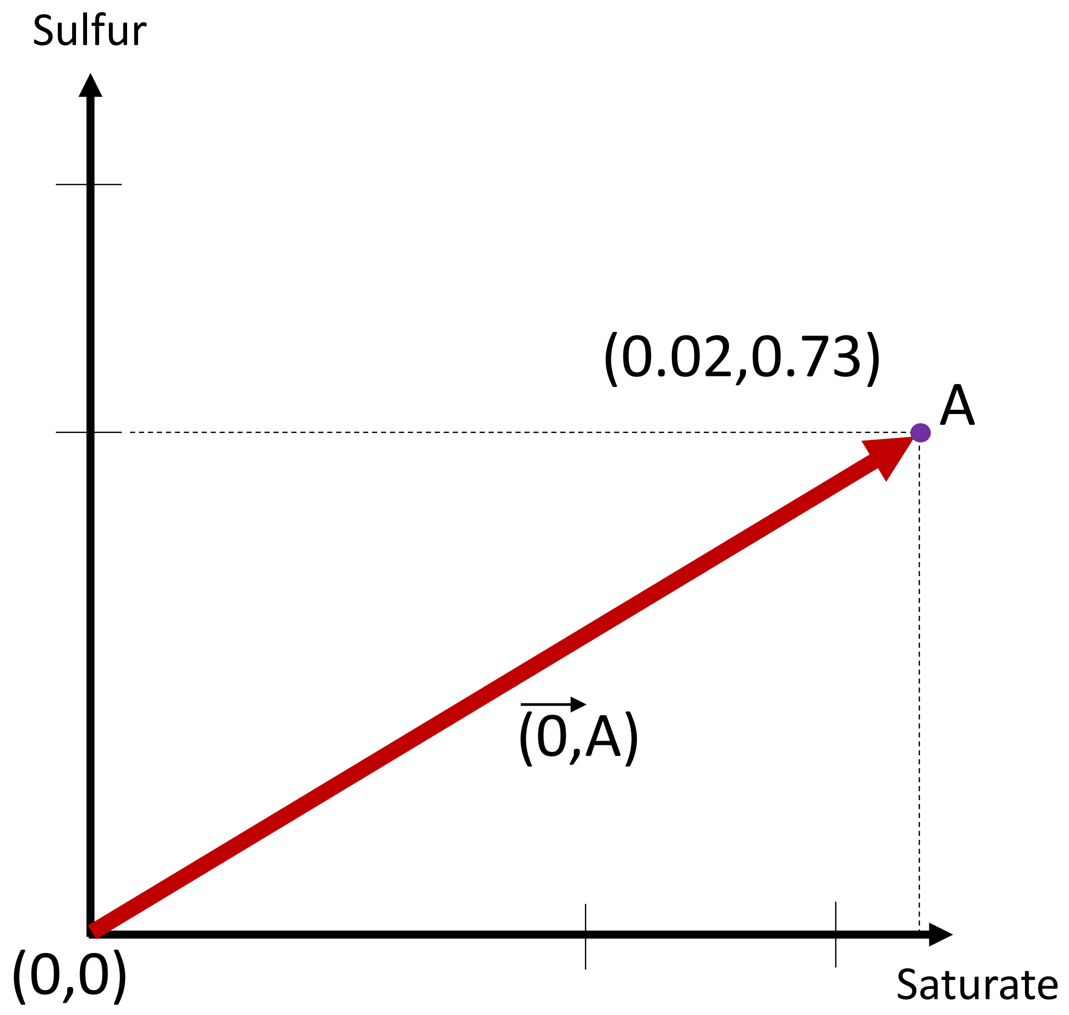
Linear Regression vs. Linear SVMs

- Similar to Linear Regression, SVMs, are a supervised ML algorithms for identifying a hyperplane to linearly separate a set of data points
- The problem with linear regression is that it may identify several possible hyperplanes with the same data, of which none are optimal



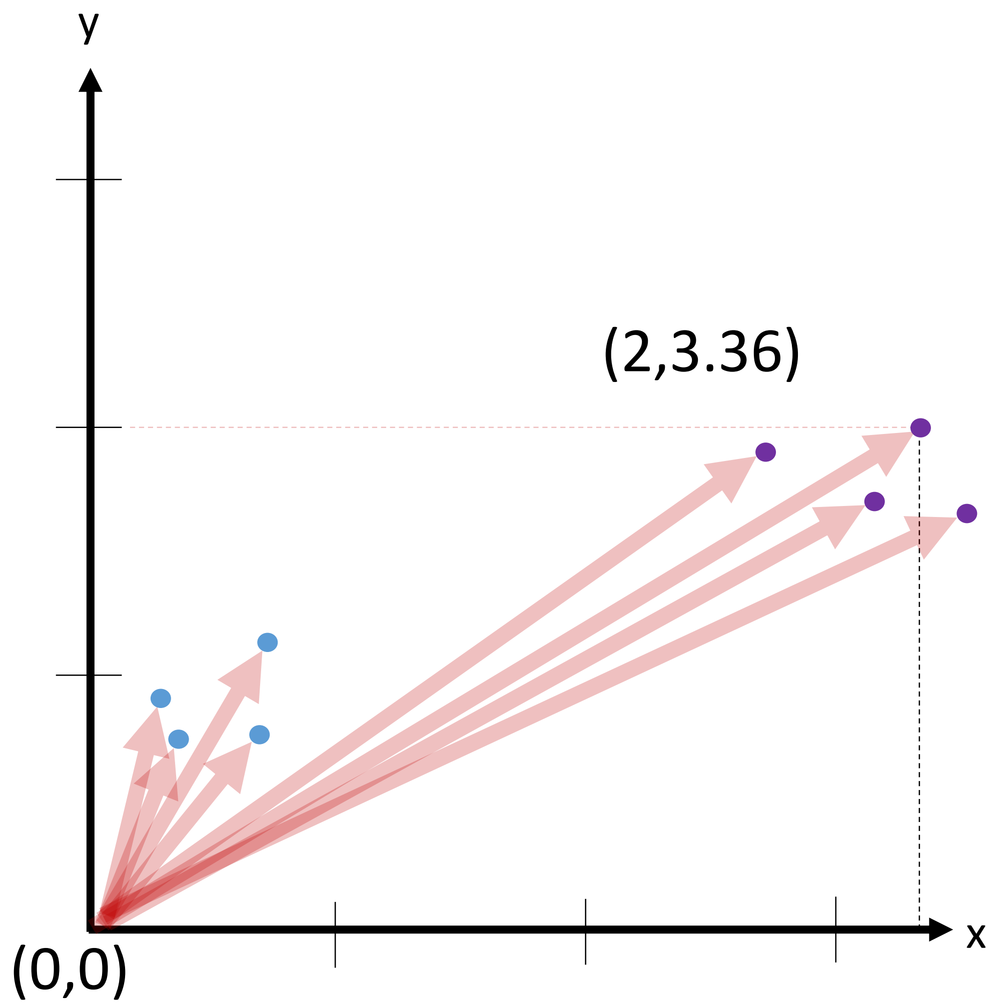
Representation of Samples Geometrically

- Assume that a subject (e.g., synthetic or petroleum-based motor) is described by *n* characteristics (features)
- Representation: every oil tested has a vector in an n-dimensional space
    - Tail at point with 0 (zero) coordinates
    - Arrow-head defined by feature values
    - Direction is + or - value away from the origin
- E.g.: a oil can be represented by saturate level and sulfur.
- 0,A is the distance of the vector or the hypotenuse of a triangle



Representation of Samples Geometrically

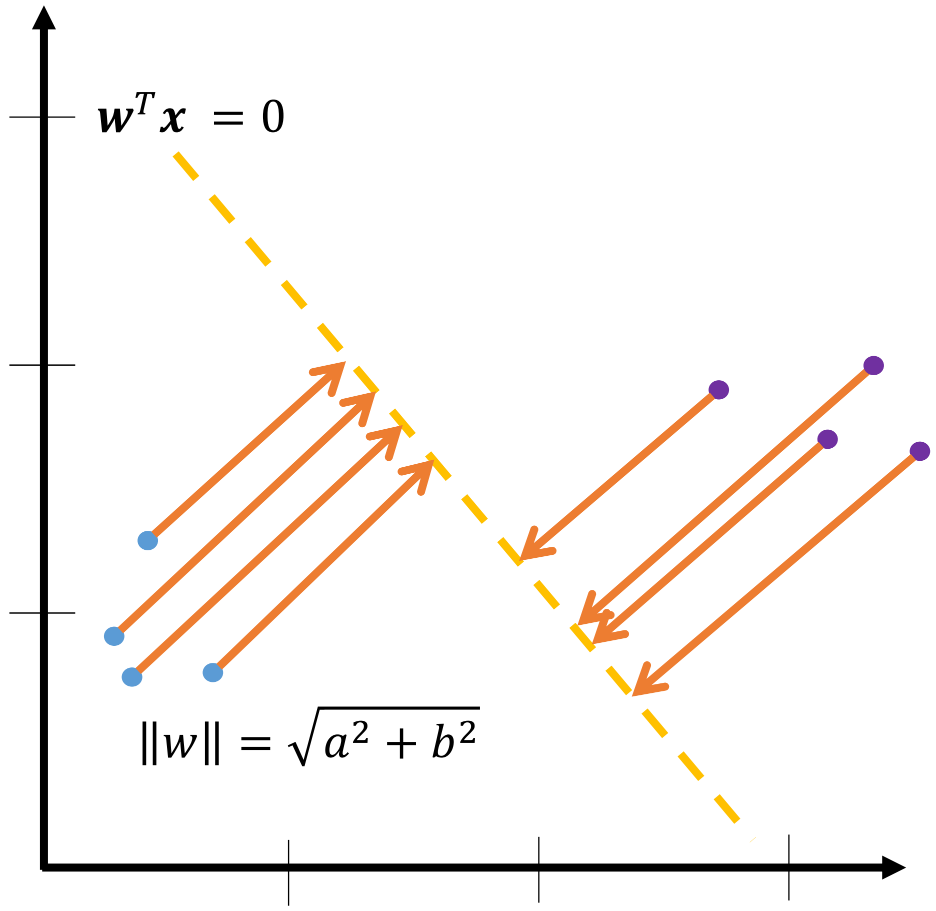
- More samples populate the n-dim space (R^n)
- New features refine the datapoint's location (positive or negative) in the feature space
- Works for large feature sets
- Once all of the vectors are plotted in R^n determine the best boundary between them



Find an Optimum Decision Boundary

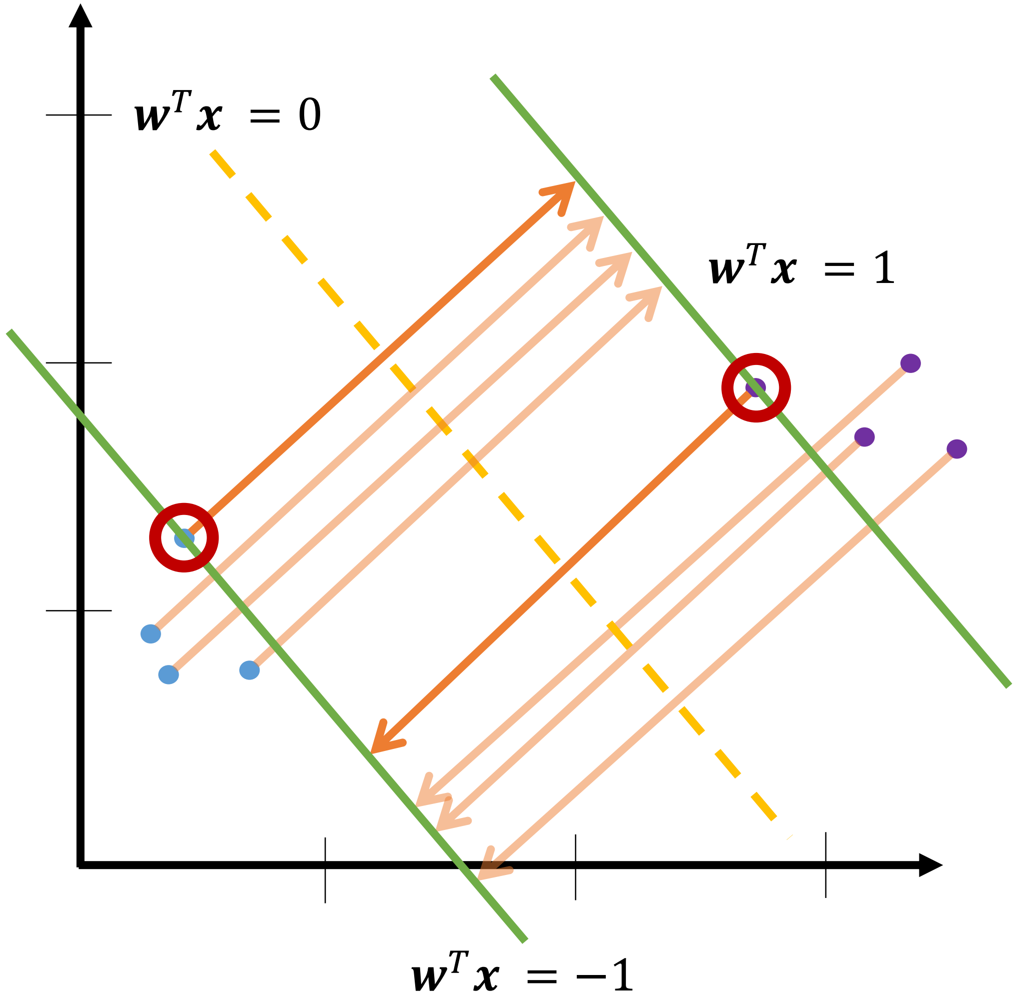
- Decision boundaries classify all the data points correctly
- Several hyperplane may satisfy this requirement
- For SVMs, we are looking for the Euclidean dot product calculated as follows:

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


$\boldsymbol{w}^T\boldsymbol{x} = 0$
$\|w\| = \sqrt{a^2 + b^2}$

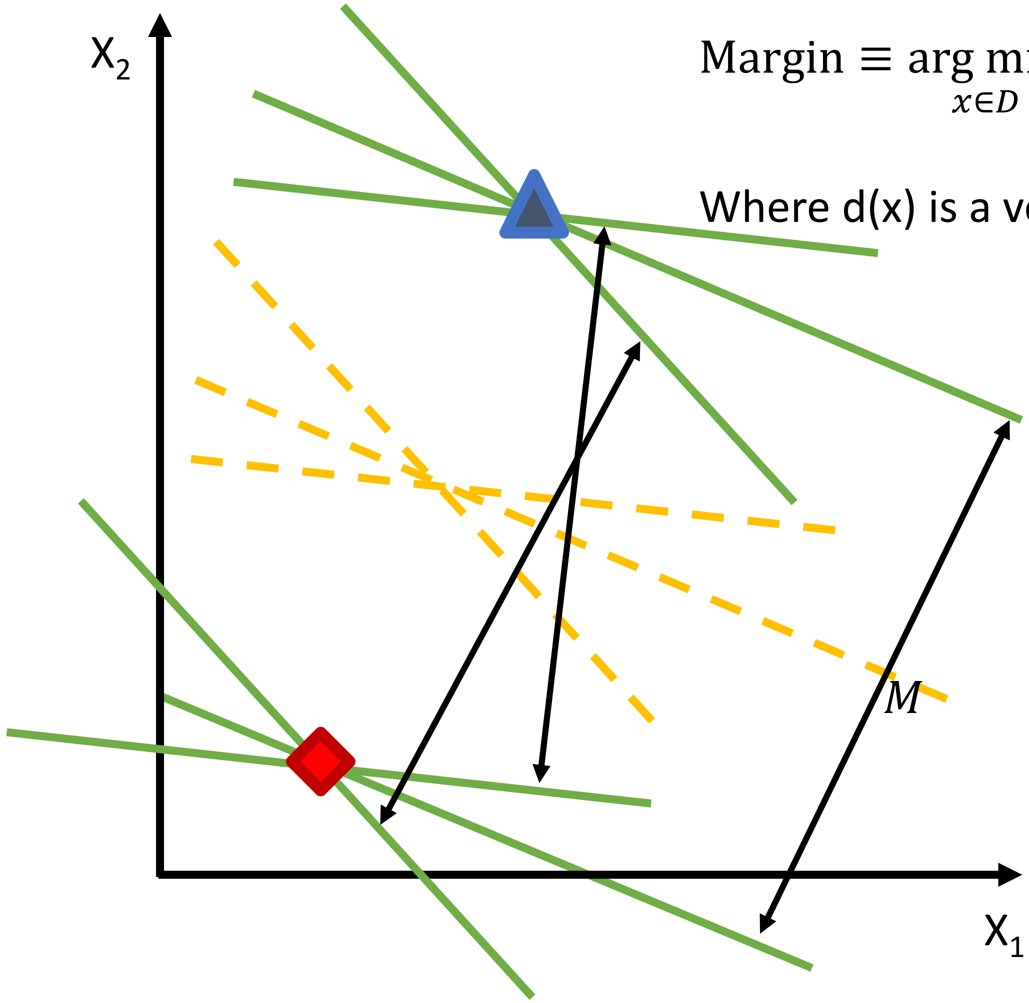
Find the Maximum Margin

- Calculate the distances from each data vector
- Maximum distance between any two points is $||p||$
- And we know that $||p||$ is midway between the two closest points
- Therefore, the distance between the margins are two parallel vectors to the hyperplane $2||p||$ distance apart


$\boldsymbol{w}^T\boldsymbol{x} = 0$
$\boldsymbol{w}^T\boldsymbol{x} = 1$
$\boldsymbol{w}^T\boldsymbol{x} = -1$

Find the Optimal Hyperplane

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



$$\text{Margin} \equiv \underset{x \in D}{\arg\min}\, d(x) = \underset{x \in D}{\arg\min} \frac{|x \cdot w|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

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

Goal:

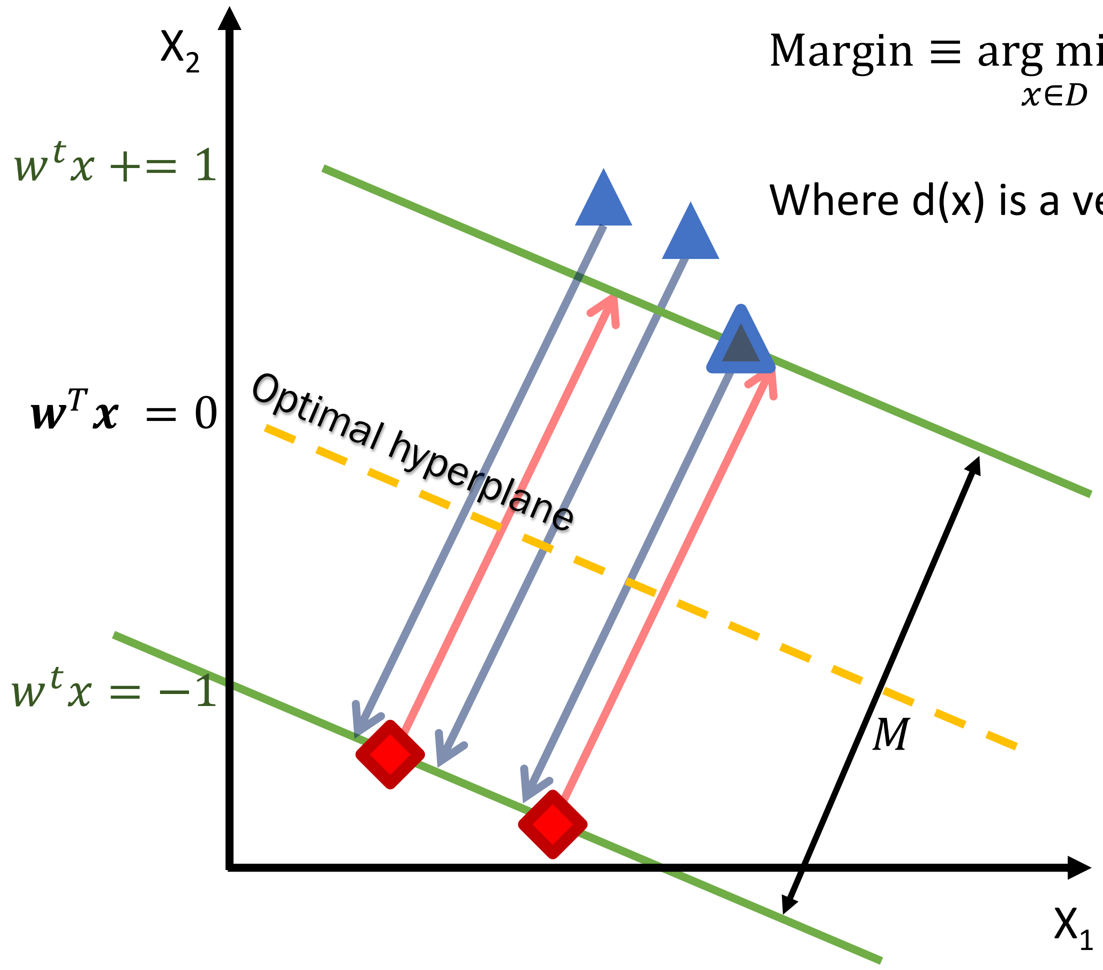
$$\text{Maximum Margin} = M = \frac{2}{\|w\|}$$

Where $\|w\|$ 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



$$\text{Margin} \equiv \arg\min_{x \in D} d(x) = \arg\min_{x \in D} \frac{|x \cdot w|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

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

Goal:

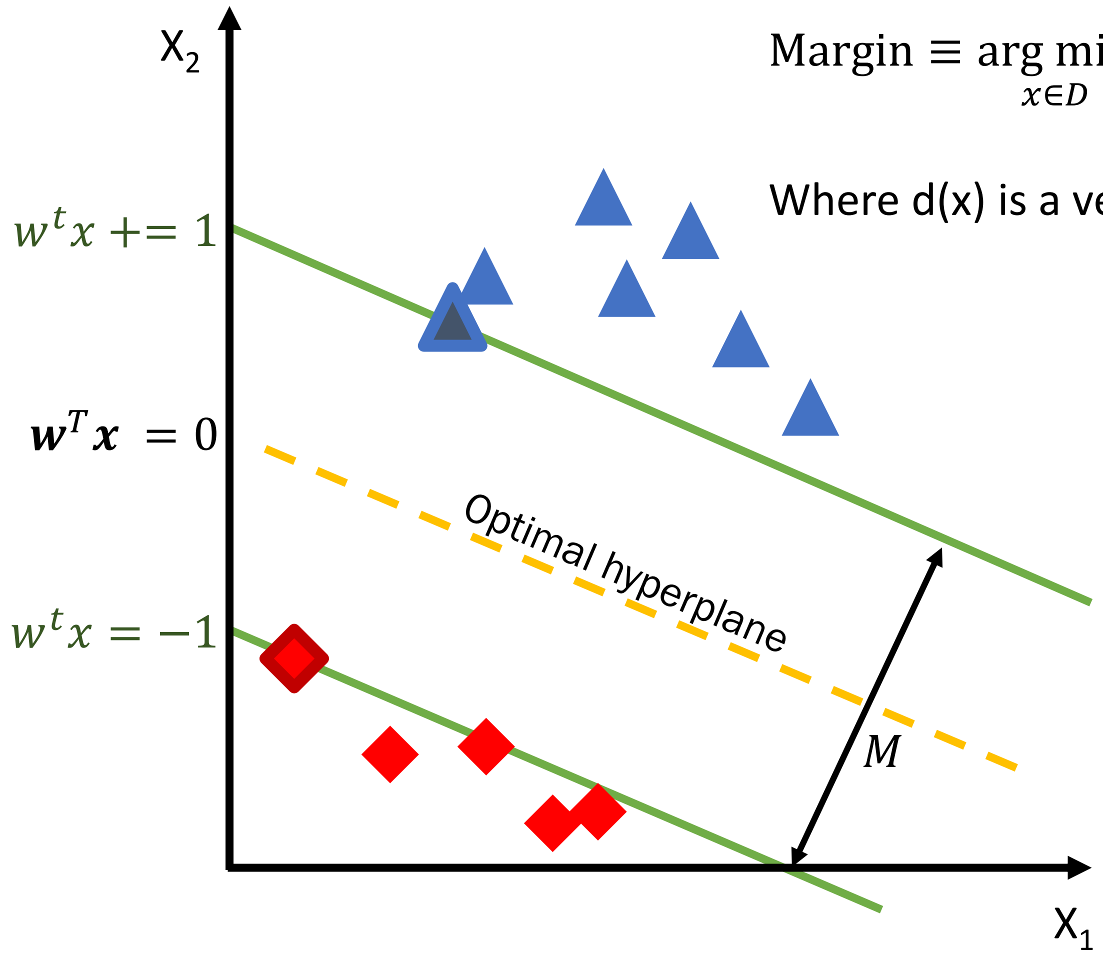
$$\text{Maximum Margin} = M = \frac{2}{\|w\|}$$

Where $\|w\|$ 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



$$\text{Margin} \equiv \underset{x \in D}{\arg\min}\, d(x) = \underset{x \in D}{\arg\min} \frac{|x \cdot w|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

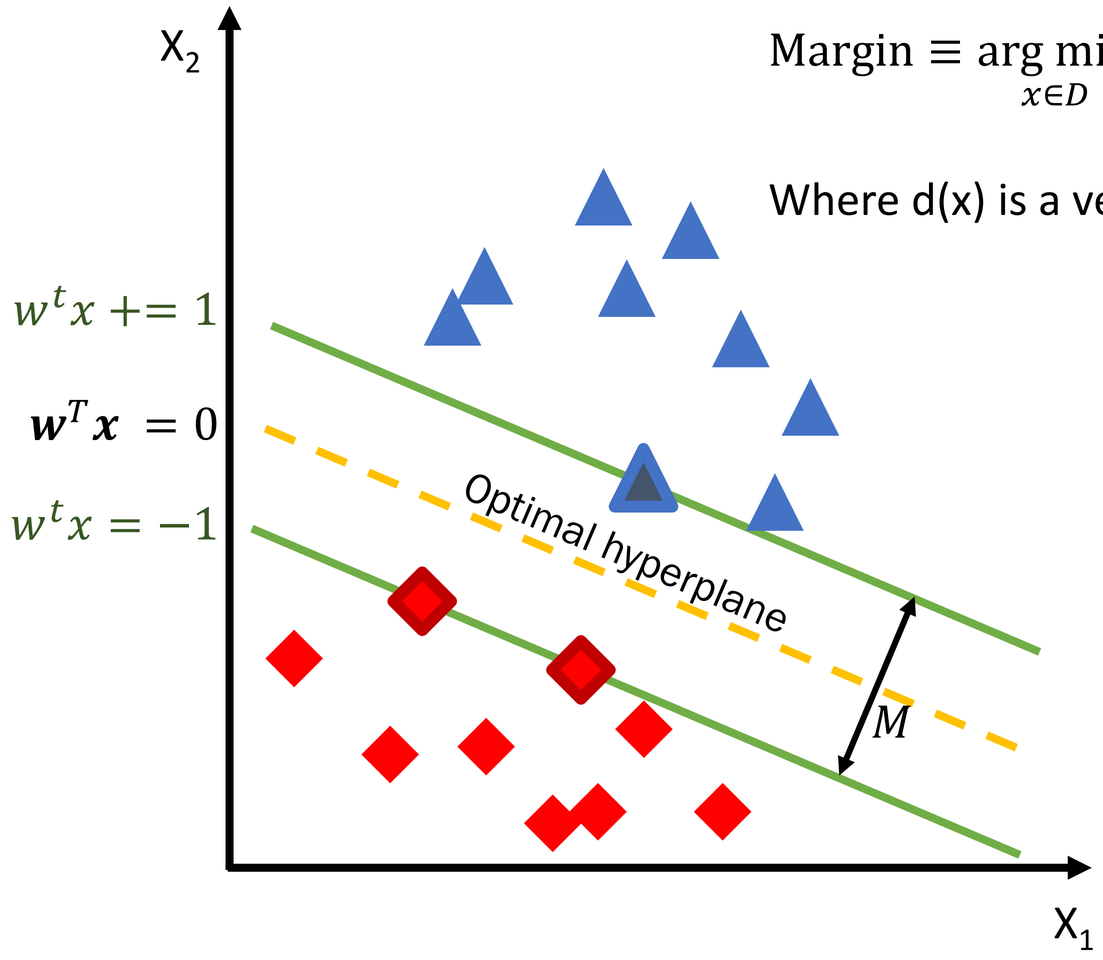
Where d(x) is a vector point in the set D and w is weight

Goal:

$$\text{Maximum Margin} = M = \frac{2}{\|w\|}$$

Where $\|w\|$ is the length of the vector of weights

Find the Optimal Hyperplane



$$\text{Margin} \equiv \underset{x\in D}{\arg\min}\, d(x) = \underset{x\in D}{\arg\min} \frac{|x \cdot w|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

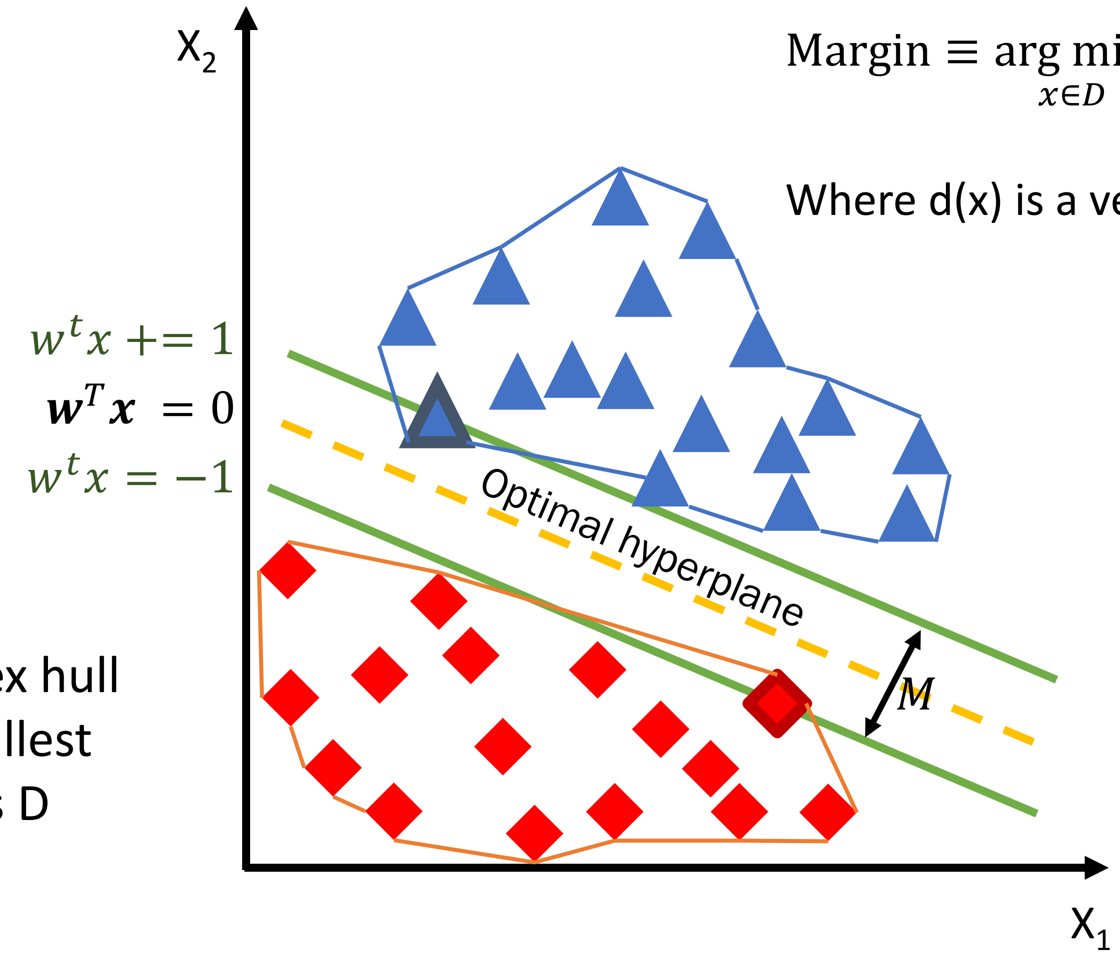
Where d(x) is a vector point in the set D and w is weight

Goal:

$$\text{Maximum Margin} = M = \frac{2}{\|w\|}$$

Where $\|w\|$ is the length of the vector of weights

Find the Optimal Hyperplane



$$\text{Margin} \equiv \underset{x \in D}{\arg\min}\, d(x) = \underset{x \in D}{\arg\min} \frac{|x \cdot w + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

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:

$$\text{Maximum Margin} = M = \frac{2}{\|w\|}$$

Where $\|w\|$ is the length of the vector of weights

Modified SVM with Slack Variables

- Also known as “Soft Margin” or “Hard Margin”
- Lower ζ_i relaxes constraints to allow the SVM to generalize better on “unseen” data points.

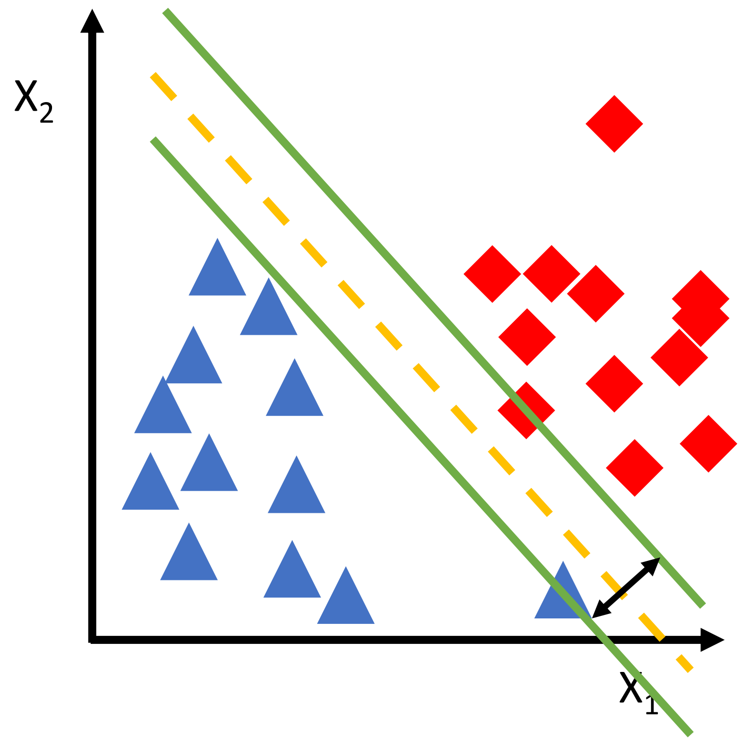
$$w^T x + b \geq 1$$

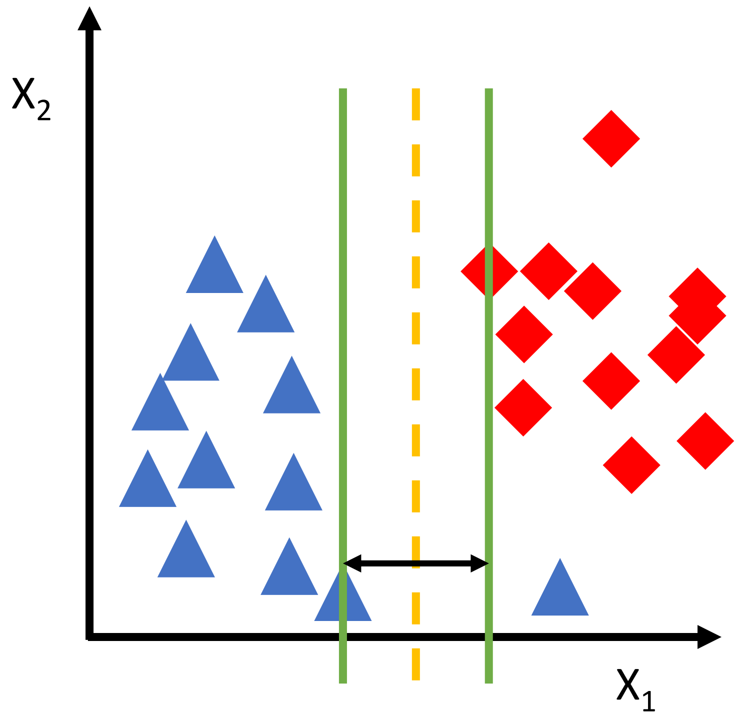
Becomes:

$$w^T x + b \geq 1 - \zeta_i$$

- Where ζ_i is an error or “cost” function that can be tightened or relaxed.
- Relaxing cost allows for mapping a data point when it is too close from the hyperplane, or it is not on the correct side of the hyperplane.

Slack Variables



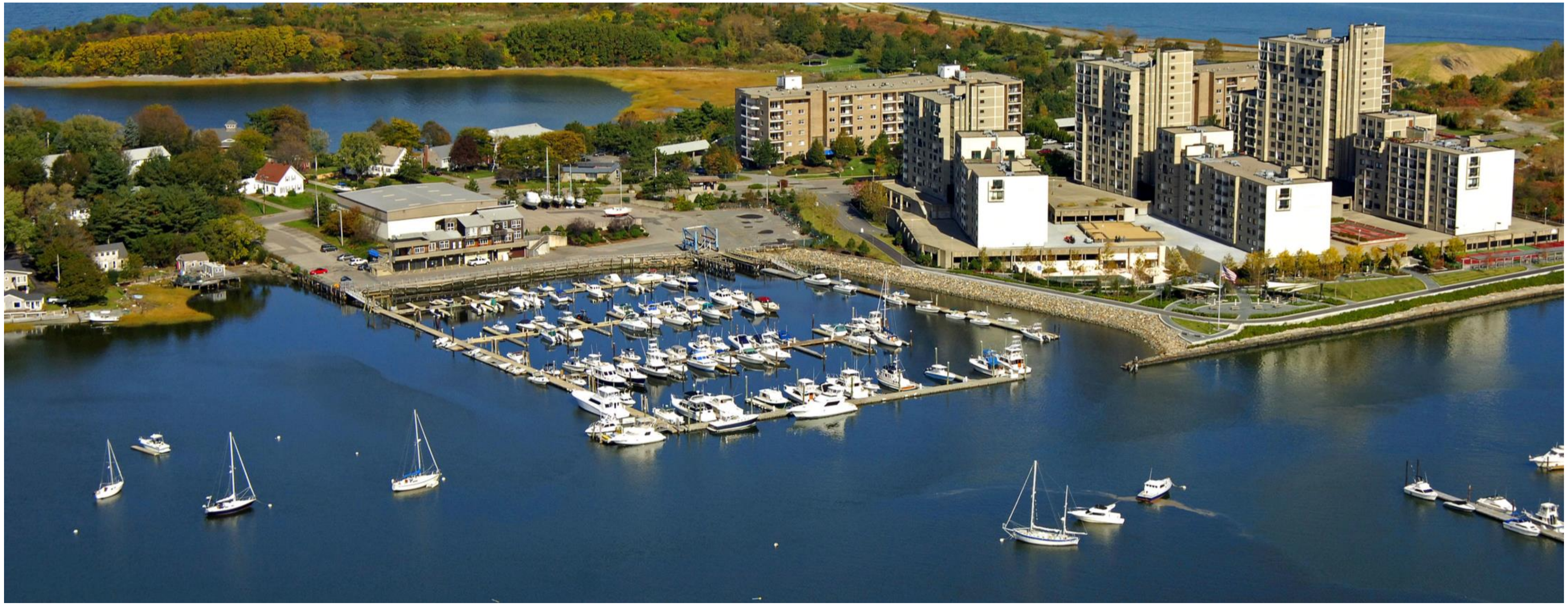


Slack variables relax the constraints to give a broader and less overfitted prediction boundary

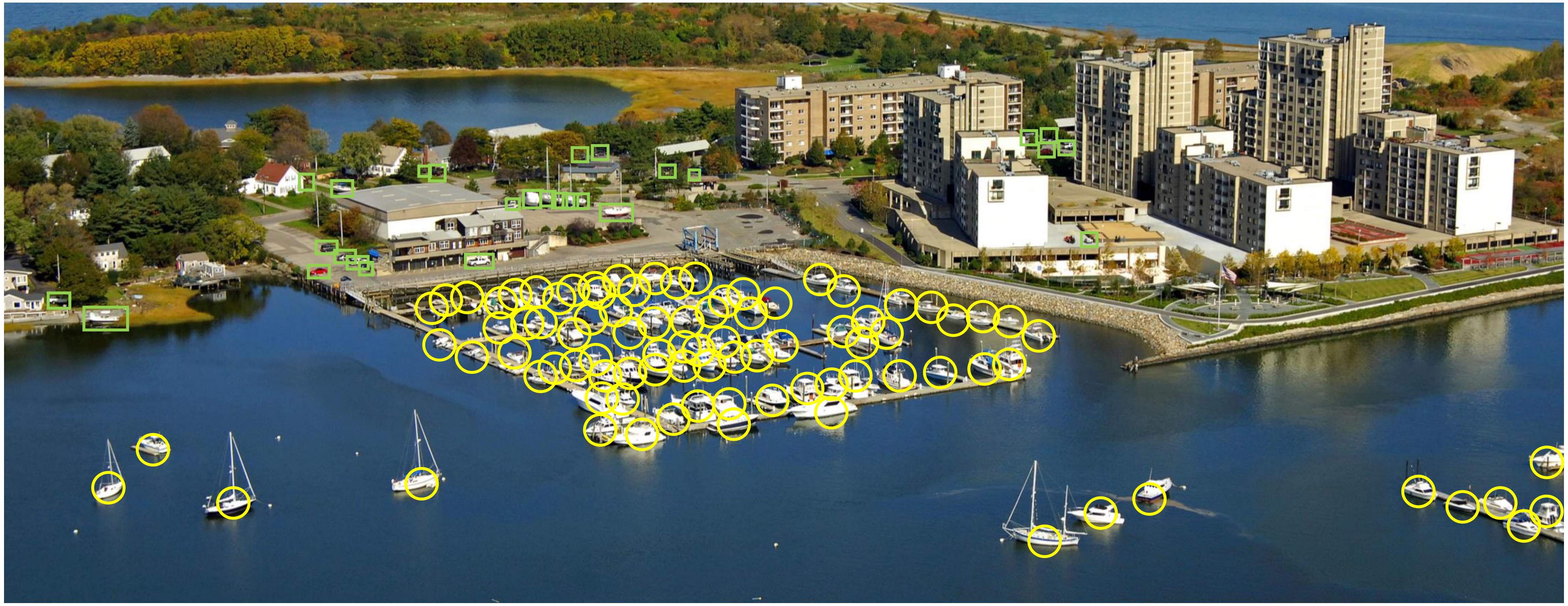
Downsides of LSVM

- LSVM only works well when you have linear separability
- Each new training data point can result in the need to regenerate the “support vectors”
- Although, there are multi-class SVMs, the typical implementation is “one vs. all”—which means we’d have to train an SVM model for every class

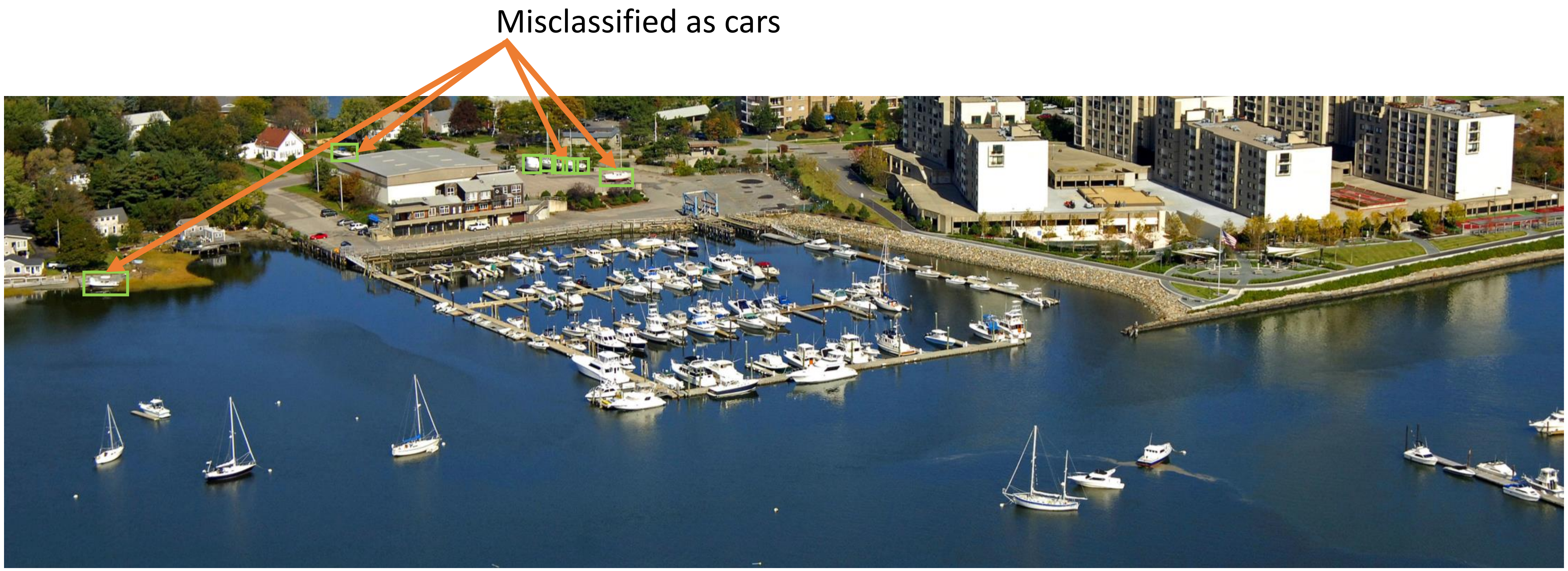
Binary Classification: Cars vs. Boats



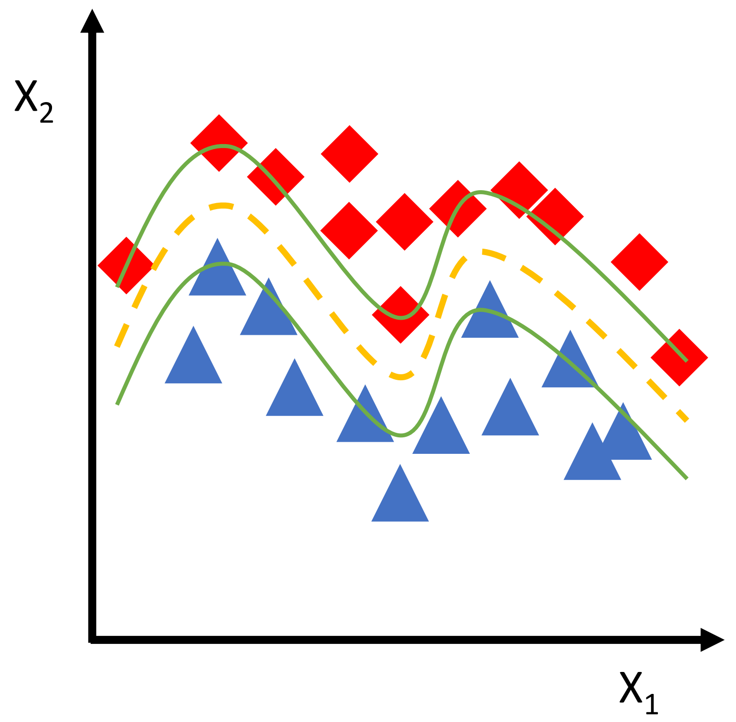
Binary Classification: Building and Boats

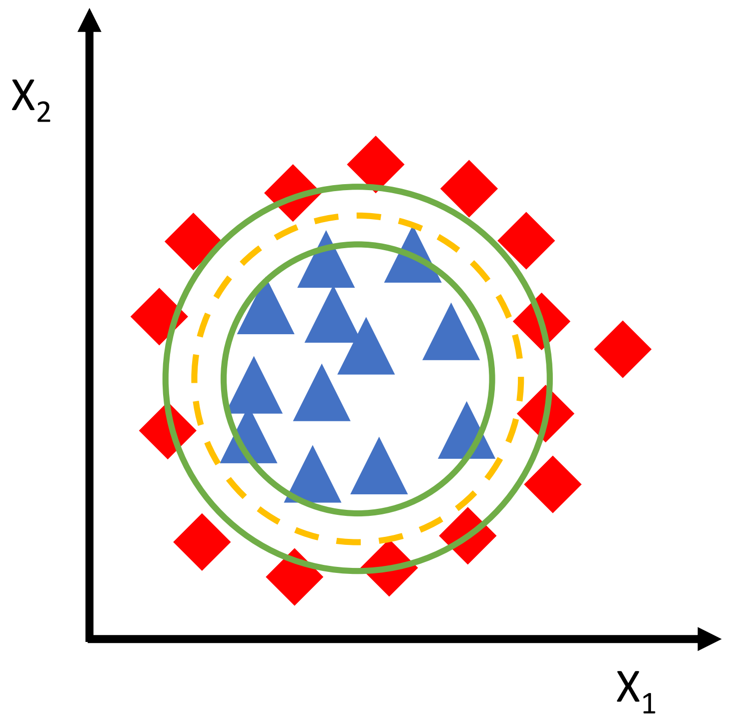


Binary Classification: Cars vs. Boats



Other Non-linearly Separability Data Sets



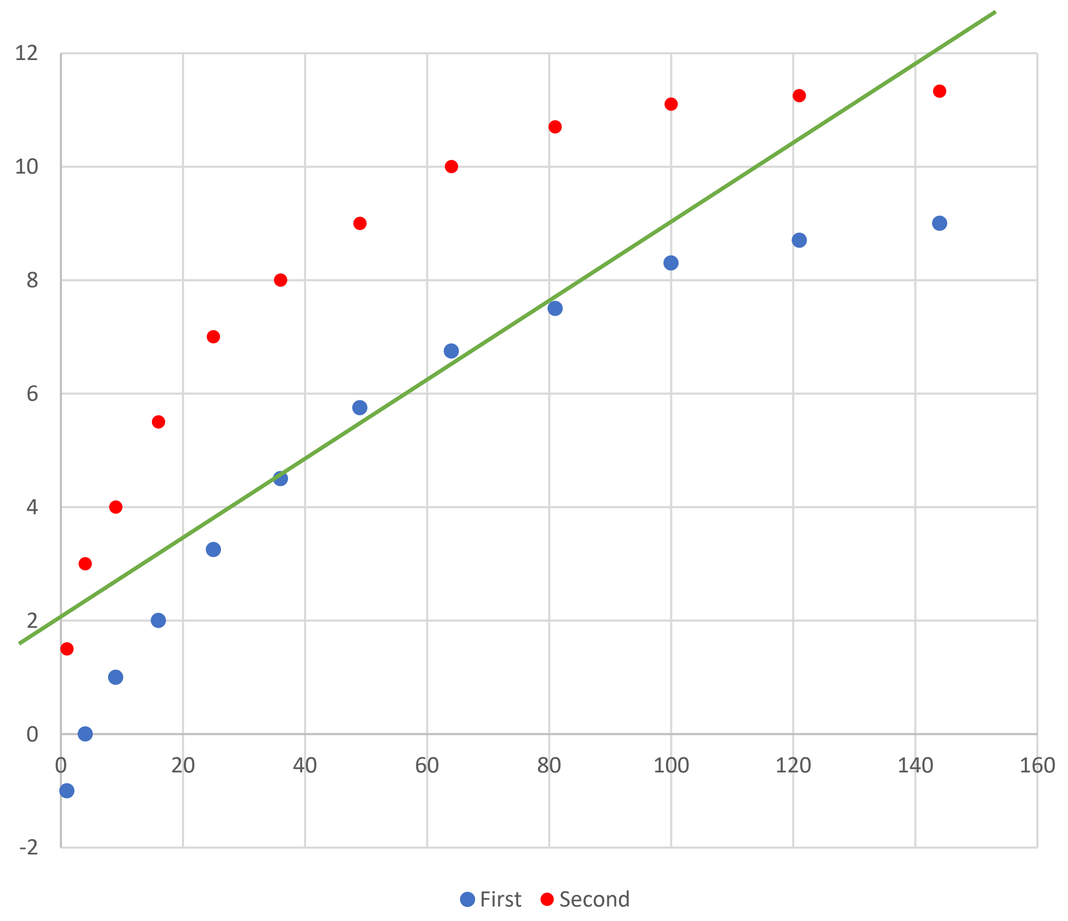


Separation in non-linear data sets is accomplished using a kernel function, of which there are several

The “Kernel Trick”

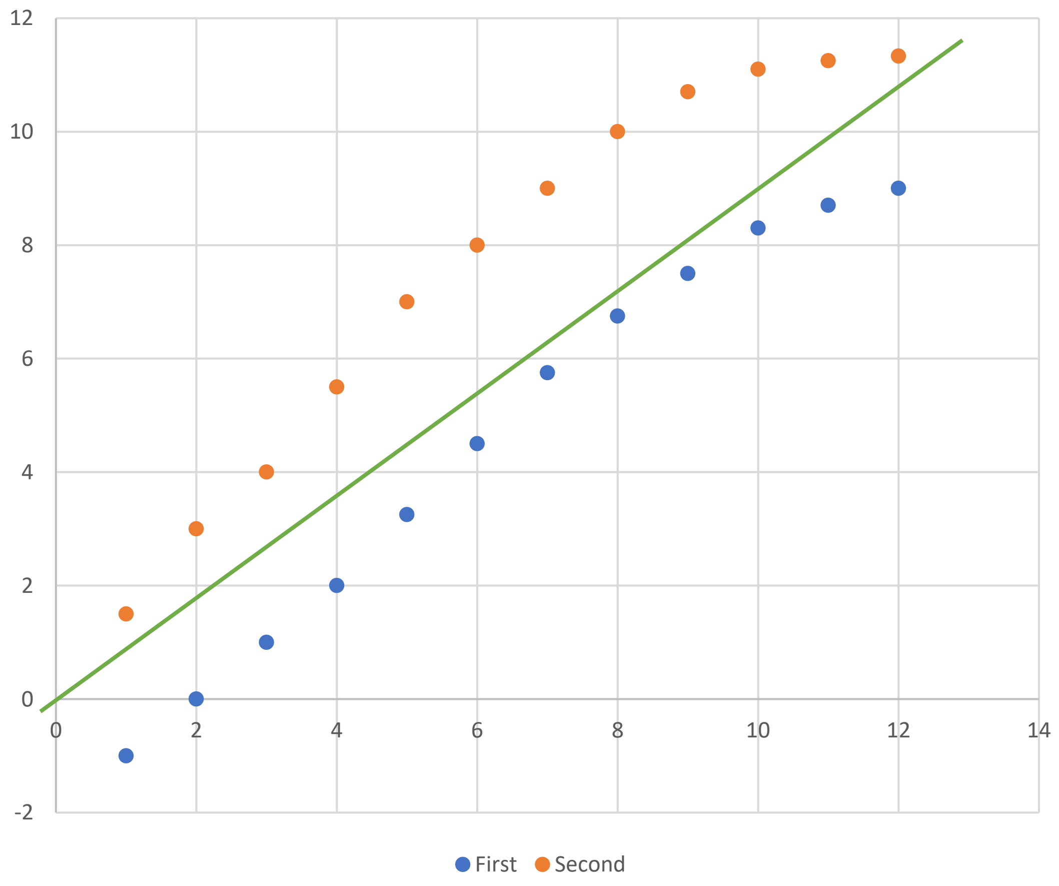
- You have an N-dimensional dataset but for computational reasons you’d rather use a linear machine learning technique to it, but it doesn’t work because the dataset is too non-linear
- You could try finding a non-linear separator to determine higher-order surfaces
- Instead you transform your input variables so that the shape of the dataset becomes more linear (e.g., square one of the variables). You do not need to preserve the dimensionality of the original dataset.
- RBF (Radial basis function) is a good one to start with—the radial distance from a center point

Simple Non-linear Example



$f(x)$	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



$f(\sqrt{x})$	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

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 x^3

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

SVM Python Notebook

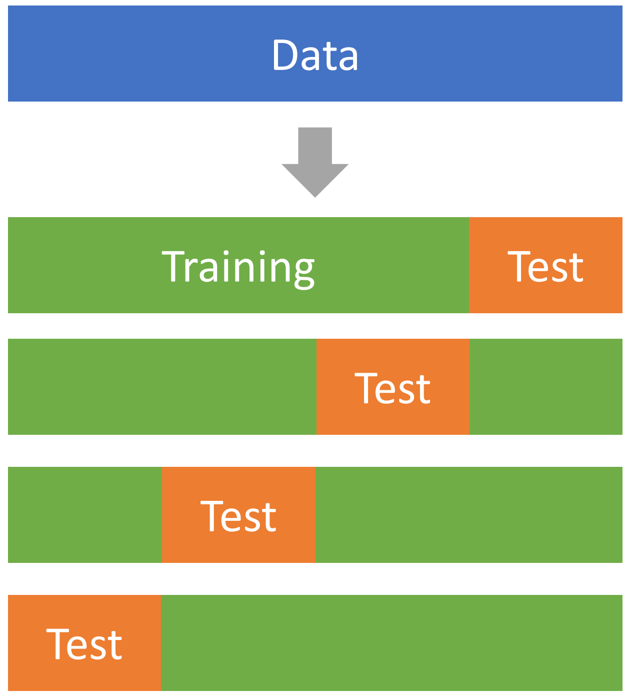
Look at the impacts of kernels on an SVM

Choosing a Kernel Function

- Probably the most tricky part of using SVM : Radial Basis Function kernel (RBF) is a good first option…
- Which is best is dependent on the dataset—try several
- It may help to use a combination of several kernels.
- Keep the same training-testing sets when you try different kernels and parameters.

Cross Validation

- Assessing if result will generalize to an independent data set in practice.
- Involves partitioning a sample of data into complementary subsets, performing the analysis on one subset, and validating the analysis on the other subset.
- Often multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.
- Cross-validation is important when you are short on data and it is hazardous, costly or impossible to collect



Fine Tuning a Model

- Grid Search enables you to avoid ‘twiddling’ hyperparameters
- sklearn.model_selection.GridSearchCV –
  - Where CV is cross validation
- Most important functions are fit, predict
- Performs an exhaustive search over the *specified* parameters and the values
- Parameters are optimized by cross-validated grid-search over a parameter grid

Caveat: Even *more* computationally expensive

SVMs are Good for Sparse Data

- SVM algorithms speed up dramatically if the data is sparse (i.e. many values are zero)
  - Why? To generate margins it requires lots and lots of dot products
  - Sparse data compute dot products very efficiently
- The sparser the data the more capability for large numbers of attributes (10s of thousands)

Things to Consider When SVM Model Building

- Start with linear regression before you move onto LSVM or try both and compare
  - If results are good enough for one or the other, stop
  - Else try SVM with a multidimensional kernel (e.g., RBF)
- SVMs require vector of real numbers
  - Categorical variables → numeric data {R,G,B} → {0,0,1},…{1,0,0}
  - May require scaling to the range [-1, +1] or [0,1]
- Use of kernel functions:
  - RBF is a reasonable first choice
  - Grid search to identify best values for parameters
  - Use cross validation to ensure good performance on test data

References

- http://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html
- Hofmann, T, Schölkopf, B, and Smola, AJ (2008) Kernel Methods in Machine Learning, Annals of Statistics, 36:1171-1220.
- Ben-Hur, A, Ong, C, Sonnenburg, S, Schölkopf, B, and Rätsch, G (2008) Support Vector Machines and Kernels for Computational Biology, PLoS Computational Biology, 4.
- Chen, P, Lin, C, and Schölkopf, B (2003) : http://www.csie.ntu.edu.tw/~cjlin/papers/nusvmtutorial.pdf
- Schölkopf, B (2000) The Kernel Trick for Distances, Microsoft Research, TR MSR(2000-51), Redmond, WA.
- Schölkopf, B (2000) Statistical Learning and Kernel Methods, Microsoft Research, MSR-TR(2000-23).
- Burges, CJ (1998) A Tutorial on Support Vector Machines for Pattern Recognition, Knowledge Discovery and Data Mining, 2(2).