Machine Learning

SVM

Outline

- Support Vector Machines (SVM)
  - Intuition
  - Important tricks

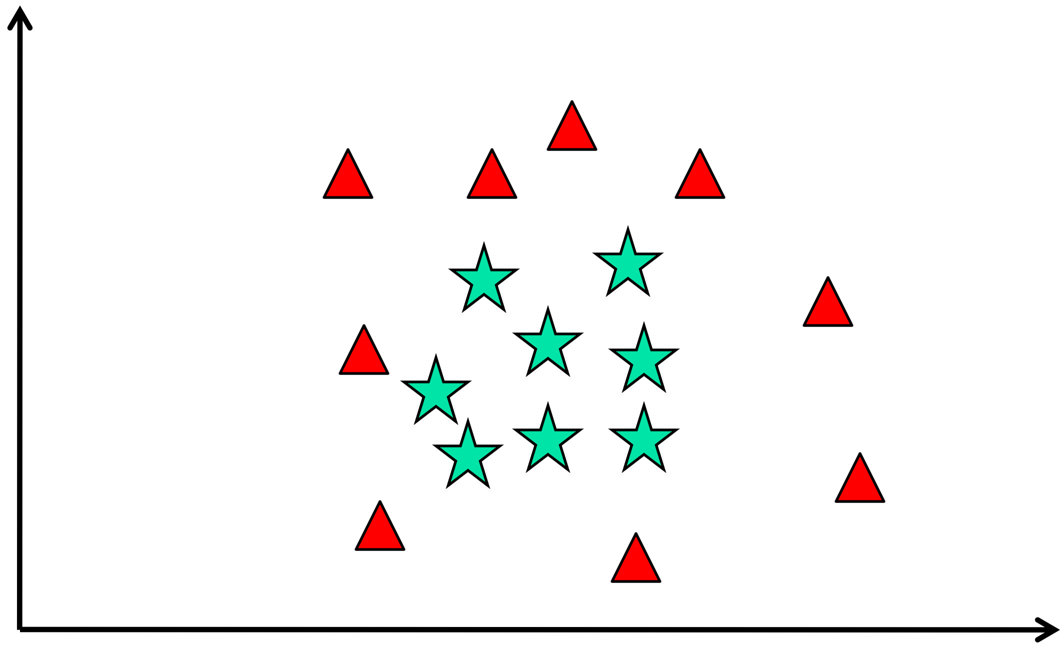
Past classes

- We have seen how the basic idea of linear regression
  - Can be used for classification
    - Perceptron
    - Logistic regression
  - Can be used to approximate non linear functions
    - Transfroming the independent variables
    - Adding attributes
    - Connecting many components to form a network

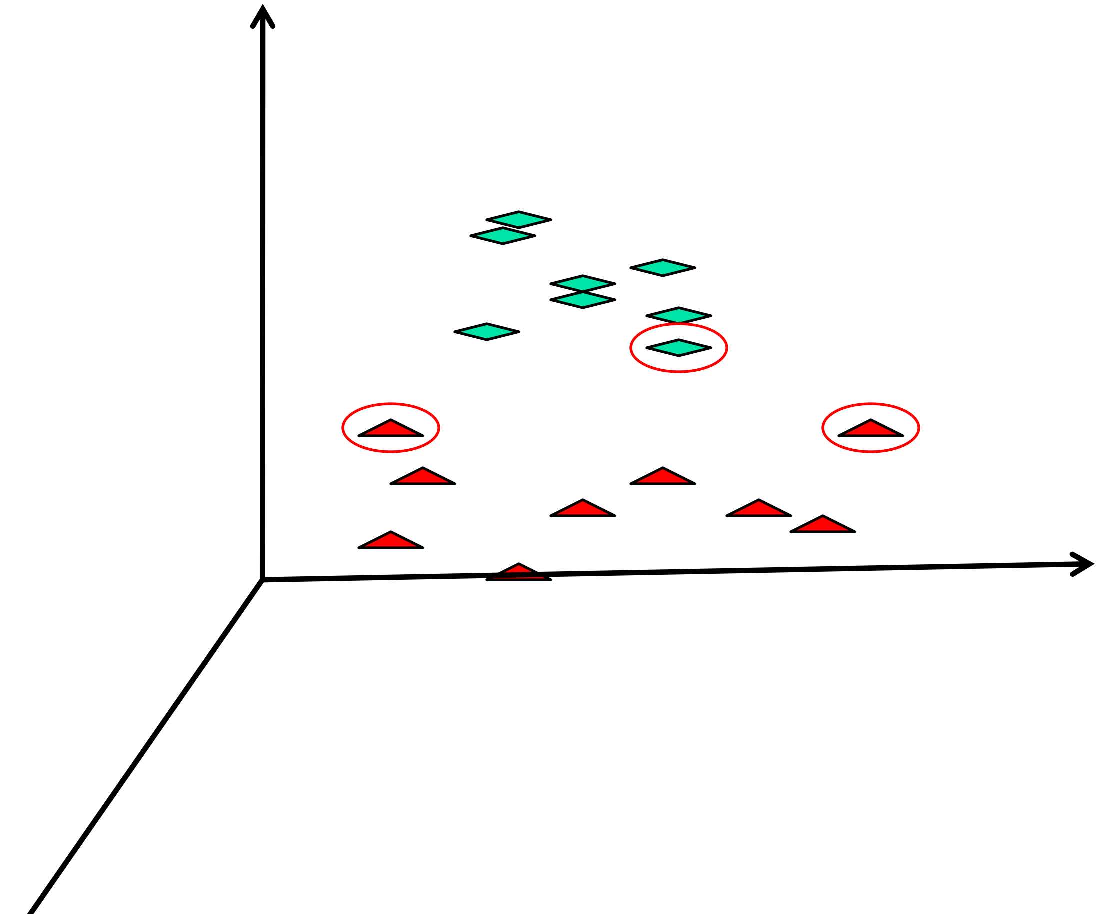
General Idea

- We want to keep on using the idea of separating classes with hyperplanes
  - We know how to solve this optimally
  - Problem, of course, is that not all problems are linearly separable
- Idea: Maybe not linearly separable in their current space, but perhaps they will be in a higher dimension
  - Create additional dimensions so that the categories become linearly separable
  - Similiar in spirit to what we did adding attributes

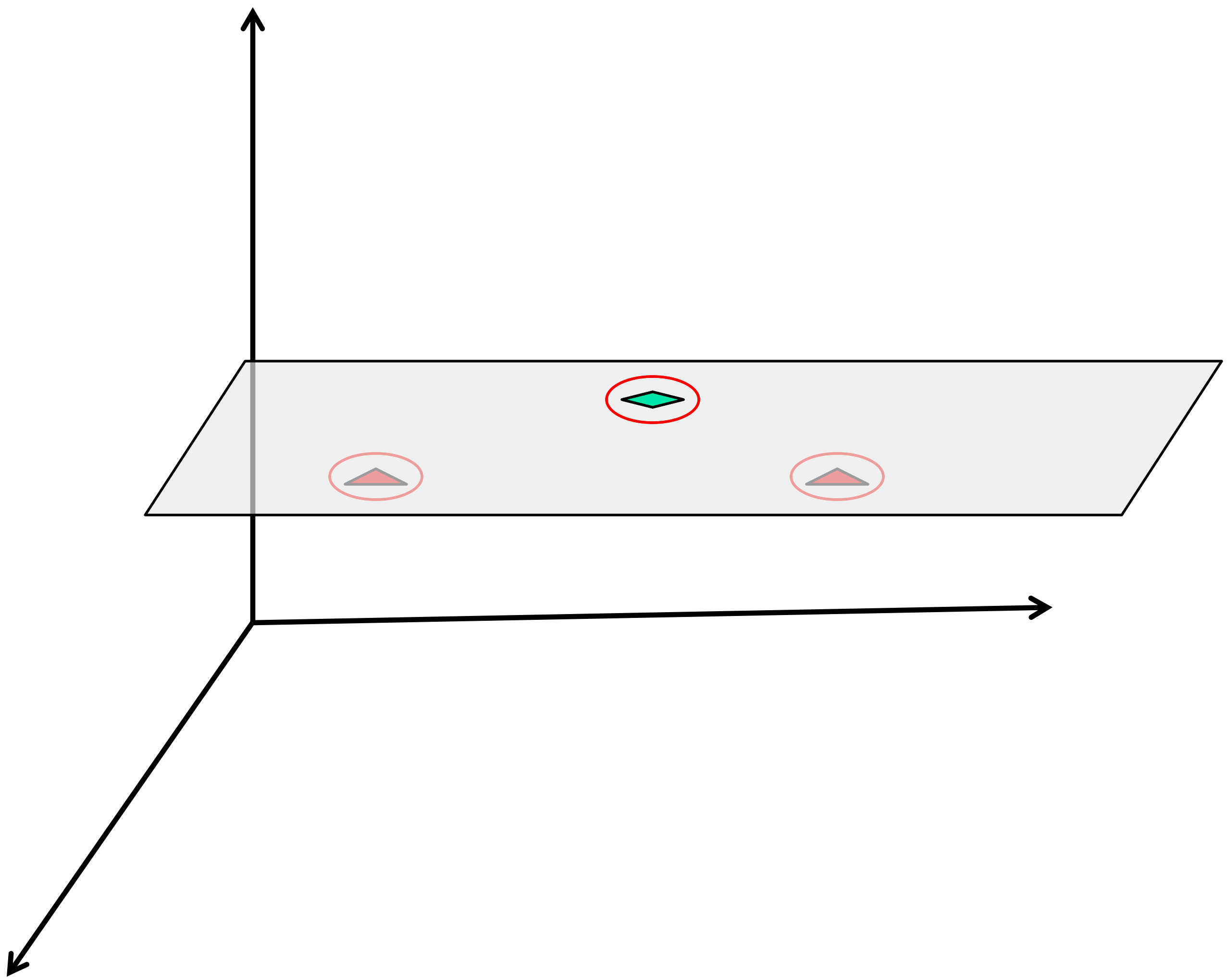
Original Data



Increase in Dimension and Support Vectors



Optimal Margin Classifier



General Idea

- The idea is familiar, so are its problems
  - How do I add dimensions automatically? How many?
  - Adding dimensions is adding new attributes or features to the original data vectors
    - In having more dimensions we increment the sideffects of the curse of dimensionality
    - More sensitivity to noisy data
    - Also, there is an increased risk of added computational burden
      - If the number of extra features is very large just computing them could be prohibitive

SVM

- Support vector machines are a proposal that addresses this issues
- Their development required the use of a a few important ideas
  - We will review them presently

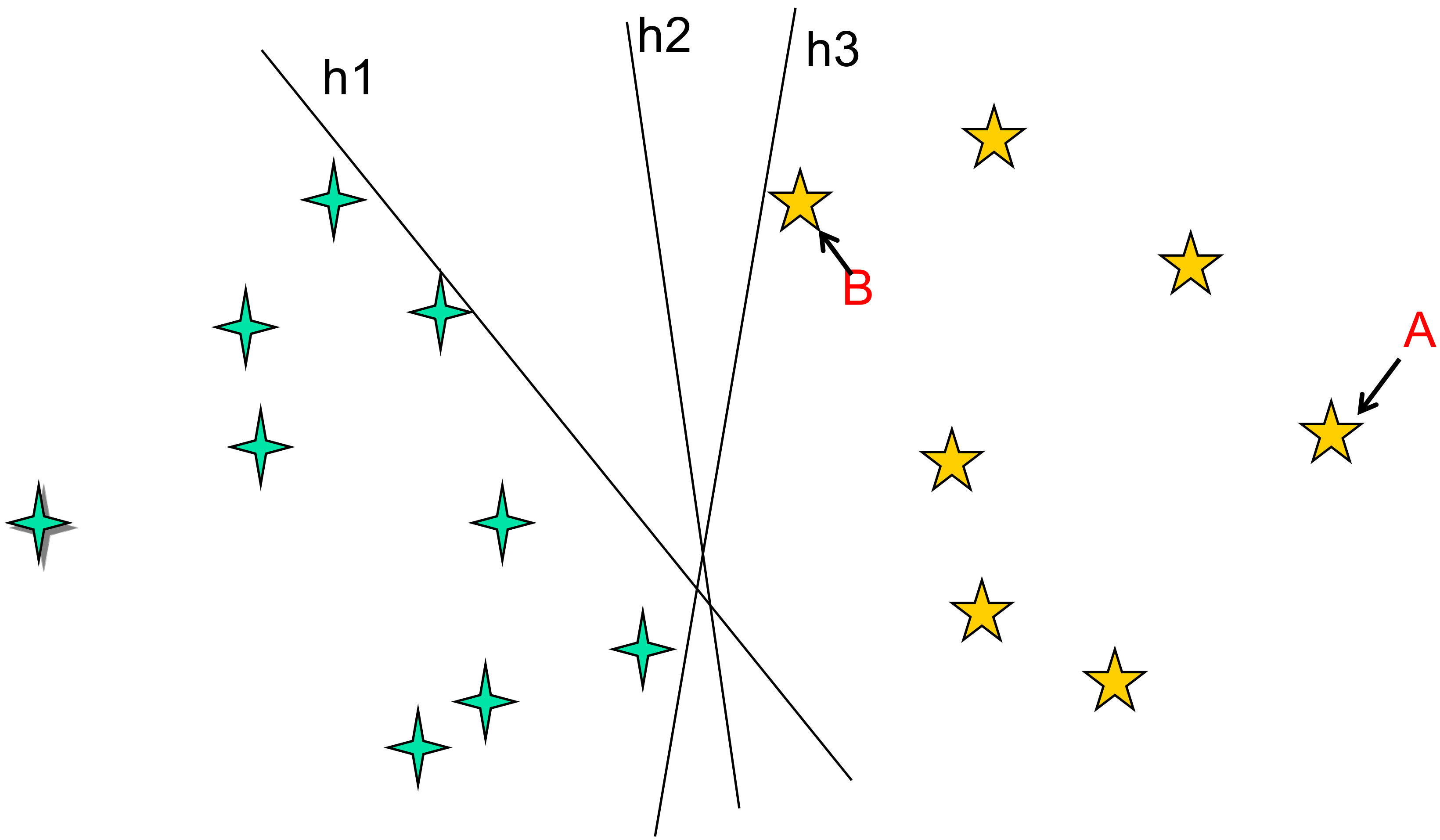
SVM
Main insights

- Optimal margin classifier
  - Reduces sensitivity to noise
  - Speeds up computations
  - Diminishes the curse of dimensionality
- Use of kernels
  - Adds dimensions to the data (arbitrarily large)
    - The kernel trick lets us pay a small computational price
- Soft margin
  - Slightly relax the requirement of lineal separability
    - Reduce sensitivity to noisy data

Optimal Margin Classifier

- Remember our lineal classifier?
  - In general, there are infinitely many hyperplanes to separate two classes
  - For what follows let us assume the classes are linearly separable in the current space
    - We will address this assumption later with the use of kernels and the soft margin

Binary Classificatio



¿Cuál separación es mejor, h1 h2 o h3?

Optimal Margin Classifier

- It is reasonable to assume that our prediccion is more reliable the further away the data point is from the separating hyperplane
  - We are more confident that A belongs to the Star class than B
  - A small change in h3 would change a change in classification for B
- h2 generalizes better

Optimal Margin Classifier Intuition

- If we think of our logistic classifier, the further away an instance is from the decision boundary, the closer the value of $g(w^Tx+w_0)$ will be to 0 or 1 (depending on the class)
- If we think of the perceptron that has only two output values the same result is given to instances on the same side of the boundary
- The optimal margin combines both this ideas
  - We want instances within a band to satisfy the first observation
  - We want instances outside the band to satisfy the second; to not contribute to the error and permit the value of w^Tx+w_0 to change without impact (as long as they remain on the proper side of the boundary)
  - In this way we can focus only on the instances that are on the border between classes

The Optimization Problem: the optimal margin classifier

- The cost function used will maximize this band
- The step transfer function establishes:
  - $V\hat{}(x) = 1$ if $w^T x >= 0$ else -1
- What we now want
  - $V\hat{}(x) = 1$ if $w^T x >= 1$ and
  - $V\hat{}(x) = -1$ if $w^T x <= -1$
  - Otherwise we want the function to vary smoothly between -1 and 1

The Optimization Problem: the optimal margin classifier

$$Min \frac{1}{2} w^T w$$

$$tal \quad que$$

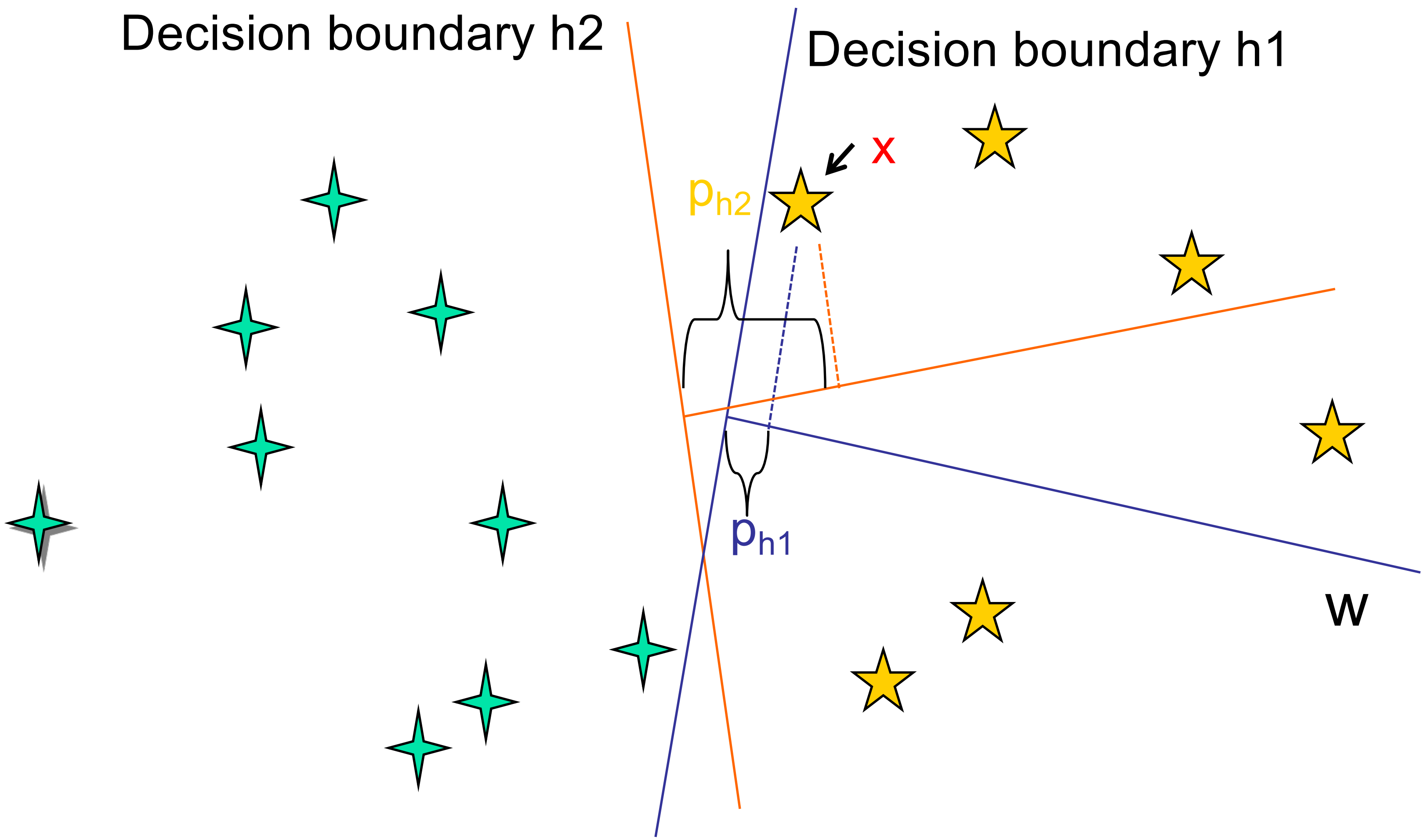
$$y_i(w^T x_i + w_0) \geq 1 \quad para \quad i = 1, \ldots, M$$

With y_i 1 o -1

Geometric Interpretation

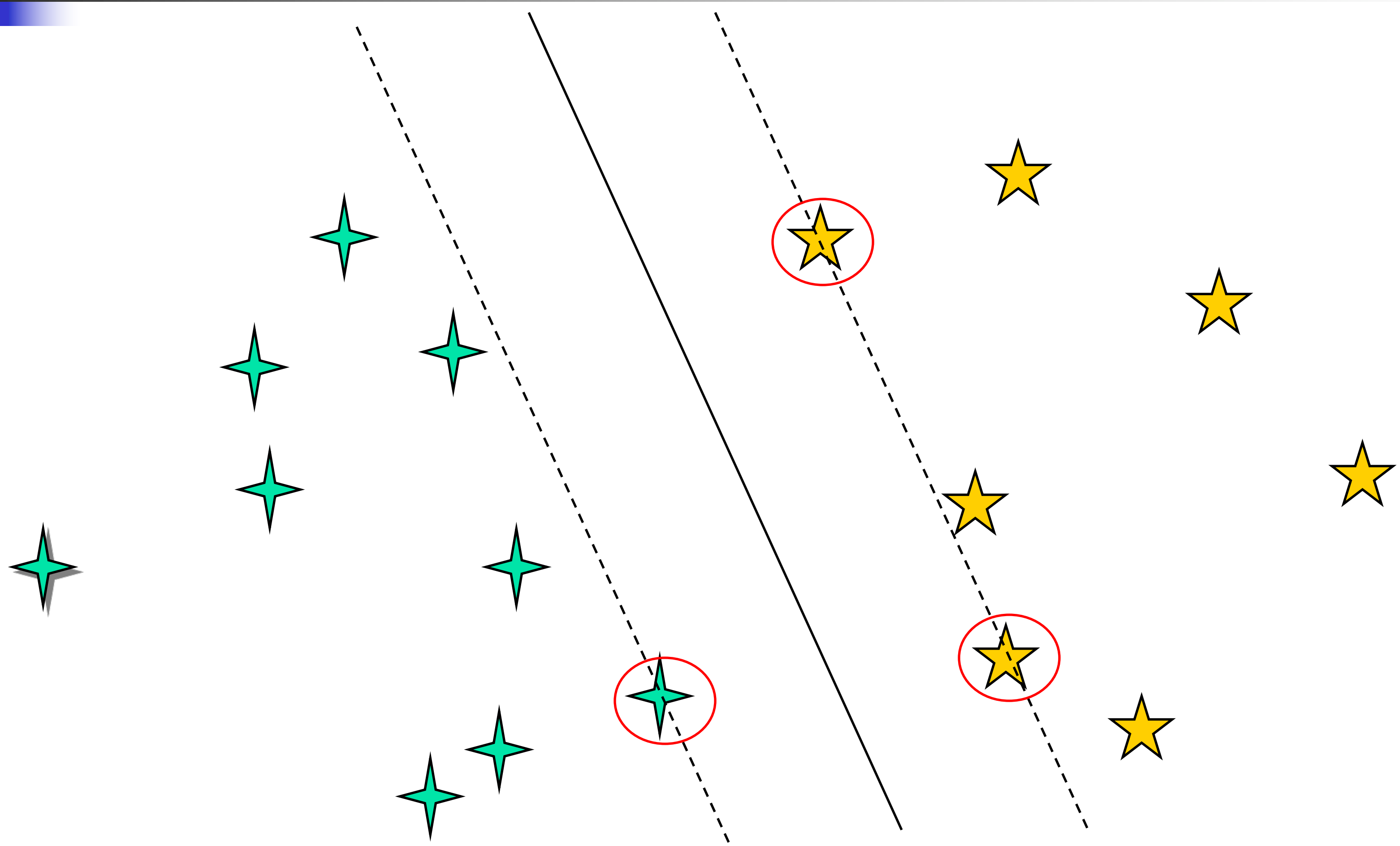
- Why does minimizing the size of w widen the margin?
  - The decision boundary is $w^T x^*=0$ and is orthogonal to the weight vector w (x^* are the values of the features that satisfy the equation)
  - We require $w^T x^{(i)} >=1$ with y=1 and $w^T x^{(i)} <=-1$ with y=-1
    - Remember that $w^T x^{(i)}=p||w||=p \, sqrt(\Sigma w_i^2)$ where p is the projection of vector $x^{(i)}$ onto vector w
  - We require $||w||^2$ to be small
    - The optimization problem forces the magnitude of the projection p to be large (very negative or very positive) in order to satisfy the restrictions thus pushing the decision boundary away from each $x^{(i)}$
    - For simplicity of explanation assume w0 =0 so that $||w||^2 =\Sigma w_j^2$

Geometric Interpretation



Which separeates the classes better h1 o h2 given x?

Optimal Margin Classifier



The three circled vectors are the support vectors

Next Steps

- We want to find the vectors that characterize the optimal margin
  - This will help with the efficiency of the algorithm
  - This will help with generalization
- Increase the dimensionality of the problem so that the data points are linearly seprable (or almost)
- For this…we need to transform our problem

Problem Transformation

- The dual problem
  - Transforms the original problem into another such that the solution the new problem, under certain circumstanes, is the solution to the original
- The goal is to have a problem that is easier or more efficient to solve
  - Our original optimization problem has an inequality that complicate the optimization (que $|w^T x|>=1$)
  - It allows us to conserve only some vectors (the support vectors), the ones that characterize the optimal margin
  - It will allow the use of kernels for elevating the dimensionality of the problem in a computationally efficient manner

Problem Transformation

- We express the problem using Lagrange multipliers

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)} (w^T x^{(i)} + w_0) - 1 \right]$$

- Note that the restrictions appear as a subtraction in the formula
- There is a set of conditions (KKT) under which the solution to the dual problem apply to the primal problem with the inequality restrictions
- This conditons give rise to the support vectors

Condiciones Karush-Kuhn-Tucker

1. $$\frac{\partial}{\partial w_i} L(w, w_0, \alpha) = 0$$

2. $$\frac{\partial}{\partial wo} L(w, w_0, \alpha) = 0$$

3. $$\alpha(y(w^T x + w_0) - 1) = 0$$

4. $$y(w^T x + w_0) - 1 \geq 0$$

5. $$\alpha \geq 0$$

The Dual Optimization Problem

$$\max_\alpha W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum\nolimits_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

$$\text{tal que } \alpha_{\mathrm{i}} \geq 0, i = 1, \ldots, m$$

$$\sum_{\mathrm{i}=1}^{m} \alpha_i y^{(i)} = 0$$

- To construct the optimization problem we minimize with respect to w y w_0 and substitute in the eqn for the lagrangian.
- If we solve the dual problem then we can recover the w's (except w_0) using:

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

  - This comes from differentiating an setting to zero (condition 1)
  - w_o is obtained similarly from condition 2

Support Vectors

- Note that we are still assuming data to be linearly separable
- Note that the problem maximizes α and forces it to be greater than or equal tho zero
- Not that condition 3 forces many α to become zero. Form here that only some vectors matter, those that support the separating hyperplane

$$\alpha(y(w^T x + w_0) - 1) = 0$$

  - This $y(w^T x + w_0) - 1$ is 0 only for those vectors x that define the margin, for the res α must be 0

The Dual Problem

- To make a prediction we need to compute $w^T x + w_0$. We substitute using the above equations

$$w^T x + w_0 = \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + w_0 = \sum_{i=1}^{m} \alpha_i y^{(i)} (x^{(i)})^T x + w_0$$

- This is our model
- If we have the α's then the only expensive thing we need to do is compute the inner product of the support vectors with the input datum

Adding Dimensions

- This is relevant since there is a way to compute the inner product of vectors without having to manipulate them explicitly
- This, together with the fact that we will only use a few vectors (the support vectors) will allow us to increase the dimentionality at a low cost

Adding Dimensions

- Remember when we saw linear regresion that we included features like x^2 in our data (we only had x as attribute)?
  - We talked about adding x^3, *sen(x)* or whatever we could think of
  - One problem was that we could over fit
    - This will be mitigated since we only use a few vectors
  - Another that we did not have a principaled way to add them

Kernels

- What we are looking for is a way to increase the dimensionality of the data that allows us to compute the inner product efficiently
  - If we can do this we can add a lot of dimensions…even infinite
- This is what we will use the kernel trick for

Kernels

- The Kernel between two vectors is defined as:

$$K(x,z) = \phi(x)^T \phi(z)$$

- Where ϕ is a mapping of attributes to new features. For example, for a vector *x* in R

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

Kernels
Examples

- Suppose we have vectors *x* y *z* in R^3
- And suppose our mapping consists on multiplying every pair of components:

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

Kernels Example

- The kernel

$$K(x,z) = \phi(x)^T \phi(z) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j$$

$$= \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_j z_j \right) = \left(x^T z \right)^2$$

- Computing the kernel takes time proportional to the number of attributes while the number of generated features is proportional to the square of the number of attributes

Other Kernels

- Not all mappings have this property (Mercer's theorem establishes when a Kernel is valid).
- These are some common kernels

$$K(x,z) = (x^T z + c)^d$$

$$K(x,z) = \exp\left(\frac{-\left\| x - z \right\|^2}{2\sigma^2} \right)$$

$$K(x,z) = \tanh(\kappa x^T z - \partial)$$

Kernels

- Picking a Kernel an its parameters is a bit of an art
  - Not much theory
  - We could even design our own. Mercer's theorem will guide us to design a valid one
- One useful intuition to guide the selection of a Kernel is to think of $K(x,z)$ as a measure of how similar $\phi(x)$ y $\phi(z)$ are
  - The value of $K(x,z)$ will be large when they are similar and small when they are not

The New Model

$$\Pr ediccion(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)} x) + w_0$$

- Note that the model only uses the support vectors and that K only requires computing the inner product without enumerating the added attributes

Other Details

- All of the above was done assuming that the classes are lineaarly separable in some space
  - We can relax this supposition so that they are "almost" linearly separable and allow some datapoints to cross the border.This is called the soft margin
- There are a few efficient algorithms for solving the dual optimization problem
  - One of the most common is called "sequential minimal optimization" (SMO)

Soft Margin

- Data might not be linearly separable for two reasons:
  - The presence of noisy or mislabelled data
  - Data is intrinsically non-separable in this space
- For the second point we use Kernels to elevate the dimensionality
- For the first issue we use the soft margin

Soft Margin

- The idea of the soft margin is to permit some data points to be missclassified and not affect the margin
- We introduce a tolerance or slack parameter
  - Before

$$y(w^T x + w_0) \geq 1$$

  - Now

$$y(w^T x + w_0) \geq 1 - \zeta$$

  - We allow for data points to be “a little” misclassified, we allow their distance to the correct side of the boundary to be at most psi

The Optimization Problem: soft margin classifier

$$Min \frac{1}{2} w^T w + C \sum_{i=1}^{M} \xi_i$$

$$tal \quad que$$

$$y_i (w^T x_i + w_0) \geq 1 - \xi_i \quad para \quad i = 1, \ldots, M$$

$$y \quad \xi_i \geq 0 \quad para \quad i = 1, \ldots, M$$

The Dual Optimization Problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

$$\text{tal que } 0 \leq \alpha_{\mathrm{i}} \leq C, i = 1, \ldots, m$$

$$\sum_{\mathrm{i}=1}^{m} \alpha_i y^{(i)} = 0$$

- Almoust the same!
- The difference is that alfas must be smaller than or equal to a constant
- The support vector returned that are equal to C or –C are those that don’t respect the margin

Some advice from Andrew Ng

- If you have more attributes than data points
  - Use regularized logistic regression of SVMs with a linear kernel
- Few attributes and less than 10k examples
  - SVM with gaussina kernel
- Few attributes and more than 10k training examples
  - Add additional attributes and use logistic regression or SVM with linear kernel

Other comments

- The complexity of the resulting model is related to the relative number of support vectors (VC dimension)

Packages

- Implementing an SVM is a bit hard. A well tested and widely used library is libSVM

Exercise

- Download andSVM_2.csv
- Train a perceptron and plot the decision boundary
- Train an SVM using:
  - from sklearn.svm import SVC
  - Use a linear kernel
- Plot the data points and the decision boundary
- Plot the margin: the lines that pass through the support vectors
  - Try different values of C (1 and 100)

Exercise

- Create a data set in 2D x_1 y x_2 such that the points that lie within a circle are class 1 and those outside class 0
  - $x_1^2+x_2^2=r^2$
- Train a Neural net for this problem
  - Compute the confusion matrix and visualize the misclassified points
- Repeat using an SVM and compare