CS 383 – Machine Learning

Support Vector Machines

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

Objectives

- Support Vector Machines
  - Optimization Objective
  - Large Margin Intuition
  - Non-Linear SVMs

SVM Resources

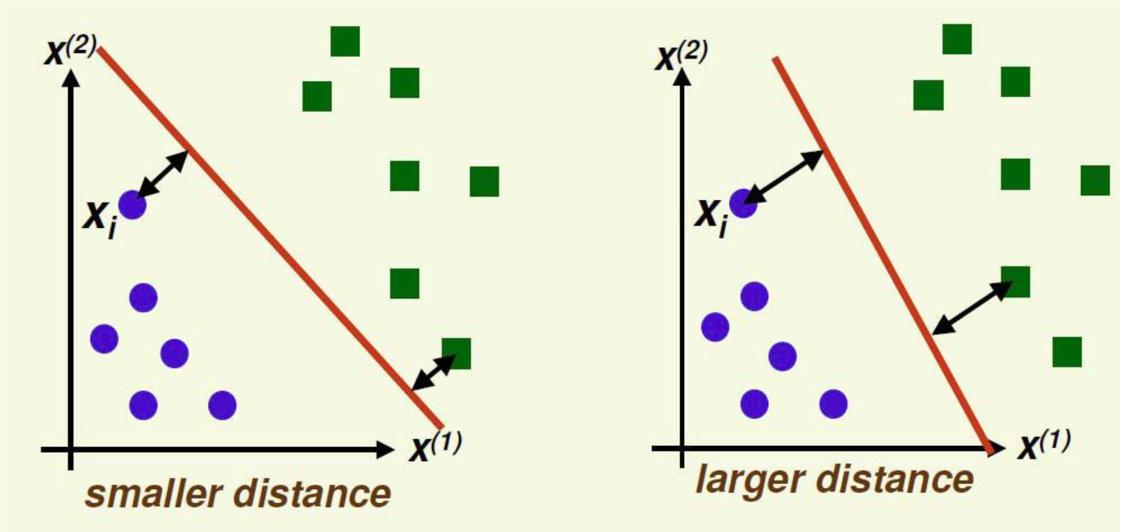
- Burges tutorial
  http://research.microsoft.com/enus/um/people/cburges/papers/SVMTutorial.pdf
- Shawe-Taylor and Christianini tutorial
  http://www.support-vector.net/icml-tutorial.pdf
- Lib SVM
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- LibLinear
  http://www.csie.ntu.edu.tw/~cjlin/liblinear/
- SVM Light
  http://svmlight.joachims.org/
- Power Mean SVM
  https://sites.google.com/site/wujx2001/home/power-mean-svm
- Matlab
  - `fitcsvm`
  - `predict`

SVMs

- Support Vector Machines (SVMs) are one of the most important developments in pattern recognition in recent years
- Elegant and successful

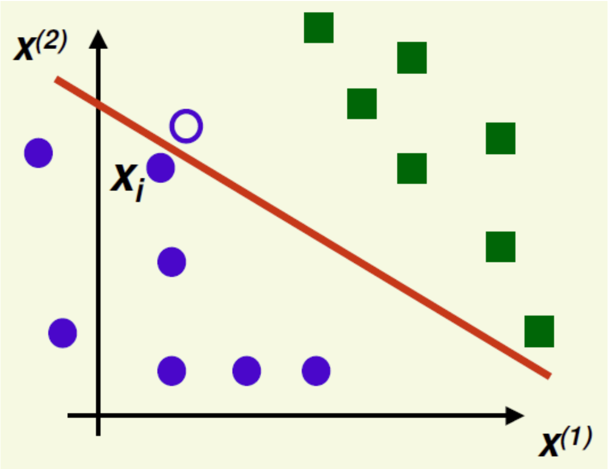
SVM Intuition

- Idea: Maximize distance to closest example (of each type)
  - For now we'll assume total separability



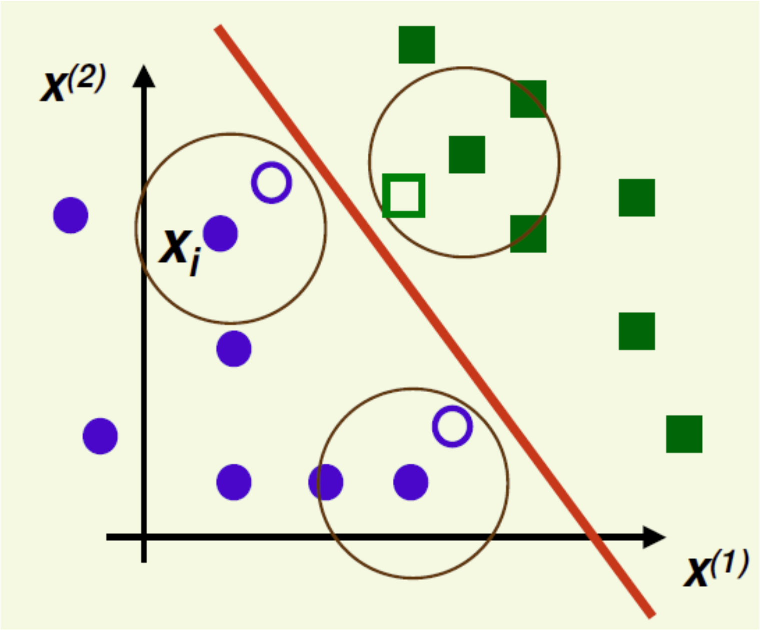
SVM Intuition

- Training data is just a subset of all possible data
  - Suppose hyperplane is close to sample X_i (open circle)
    - The *margin* is small
  - If we see a new sample close to X_i it may be on the wrong side of the hyperplane
- Therefore poor generalization



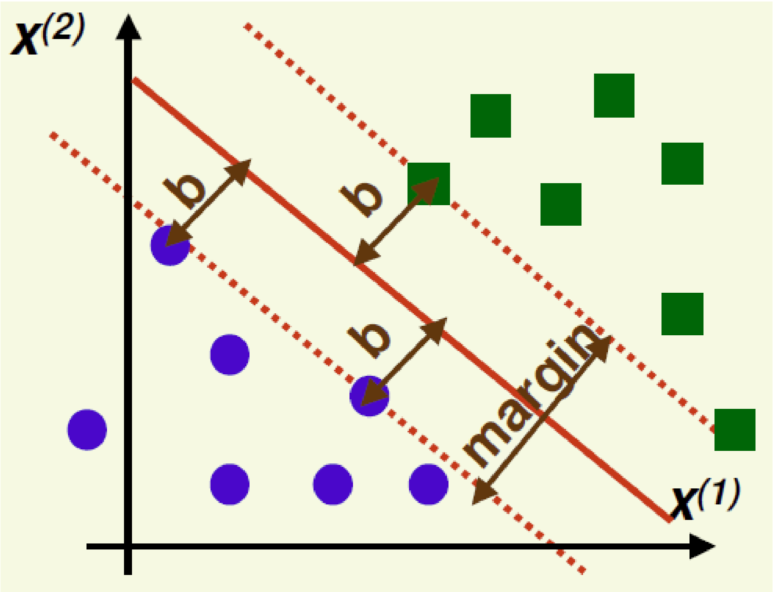
SVM Intuition

- Intuition: We want hyperplane as far as possible from any sample
- New samples close to old samples will then be classified correctly
- Good generalization



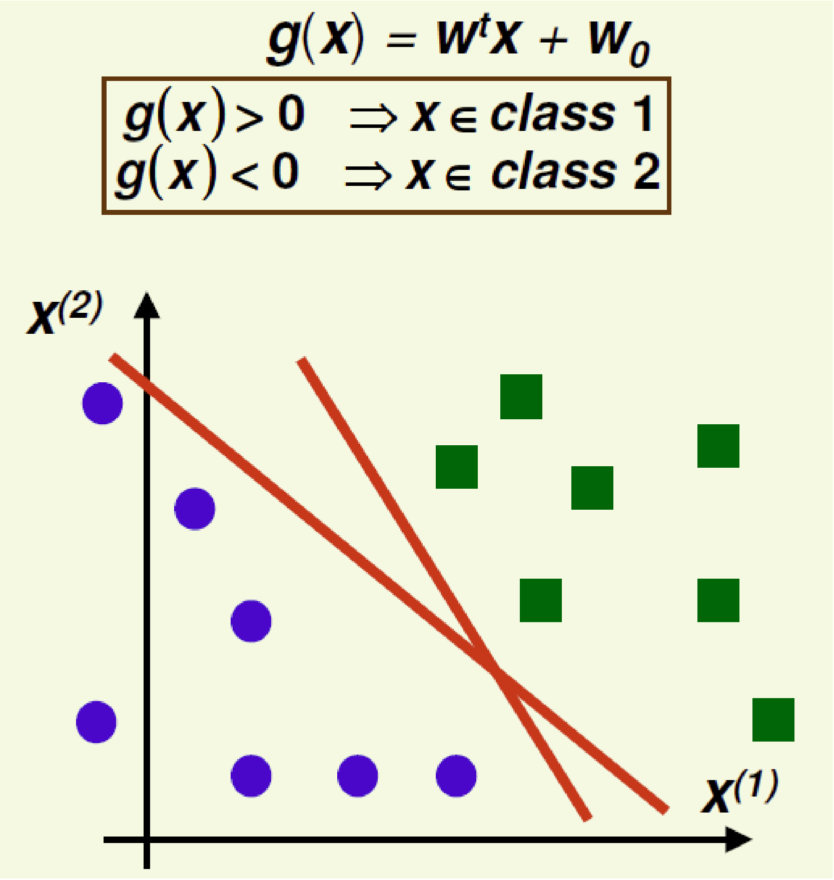
SVM – Linearly Separable Case

- Definition: Margin
  - The margin is twice the absolute value of distance b of the closest example to the hyperplane
- Our goal is to maximize the margin



Linear Discriminant Functions

- The equation of a linear hyperplane is
$$y = xw + w_0$$
- We can then create a **discriminant function** as
$$g(x) = xw + w_0$$
- We can then choose the class based on
  - $g(x) > 0 \rightarrow x$ in class 1
  - $g(x) \leq \; 0 \rightarrow x$ in class 2



Margin Formula

$$g(x) = xw + w_0$$

- So how do we find w, w_0 based on our training data in order to maximize the margin?
- As usual we'll minimize or maximize something
- We know we want to maximize the distance of the *closest samples* to our desired hyperplane
    - These closest samples are called the **support vectors**
    - Optimal hyperplane is completely defined by support vectors
        - Nice and compact representation!

Margin Formula

- Any sample *on* the hyperplane will have $g(x) = 0$
- So the distance (say the L1 distance) of any sample to the hyperplane is then:
$$d(x|W) = |g(x) - 0| = |xw + w_0|$$
- We can (somewhat arbitrarily) decide that the distance of any support vector to this hyperplane should be one:
$$|xw + w_0| = 1 \; \forall x \in \{support\ vectors\}$$
$$|xw + w_0| > 1 \; \forall x \notin \{support\ vectors\}$$

Margin Formula

$$d(x|W) = |xw + w_0|$$

- So we need to turn this into a minimization or maximization problem.
- Let's divide $d(x|W)$ by the magnitude of w, $||w||$.

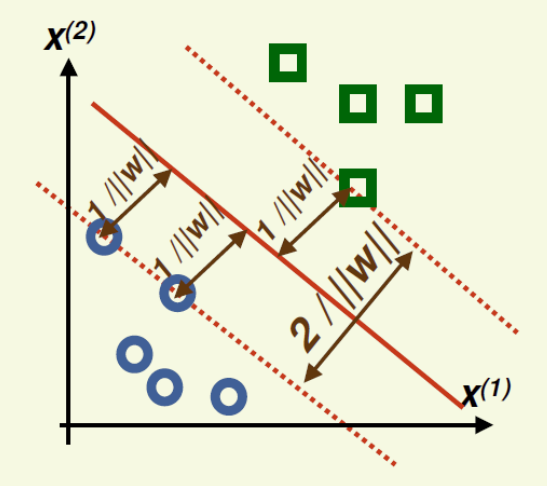
$$\frac{|xw + w_0|}{||w||}$$

- Then the closest samples (the support vectors) will have a value of

$$\frac{1}{||w||}$$

Margin Formula

- Two times this value $\frac{1}{||w||}$ gives us the **margin.**
- So we want to maximize $\frac{2}{||w||}$
- Subject to constrains
  - $xw + w_0 \geq 1$ for any positive example x
  - $xw + w_0 \leq\ 1$ for any negative example x
- We can convert this to a minimization problem
  - Minimize $\mathrm{J(w)} = \frac{||w||^2}{2}$
  - Constrained to $Y_i(X_i w + w_0) \geq 1\ \forall(X_i, Y_i), Y_i \in \{-1, 1\}$
- $J(w)$ is a quadratic function, thus there is a single finite global minimum!



Quadratic Programming

- This is called *quadratic programming* problem
  - Minimize/maximize some quadratic function subject to linear constraints
- You can solve quadratic programming problems using an extended simplex method
  - Here's a resource:
    - http://www.csse.monash.edu.au/~berndm/CSE460/Lectures/cse460-5b.pdf
- When coding you will be allowed to use SVM packages in your code
  - `Matlab:  fitcsvm, predict`
  - `Many other languages: Libsvm`
- But the next few slides will show you how to find solutions easily for a "simple" case

Finding Hyperplane

Minimize this

- To solve this constrained optimization problem we introduce *Lagrange multipliers* $a_n >= 0$ for each constraint
- This gives us the equation:

$$L(w,a) = \frac{||w||^2}{2} - \sum_{n=1}^{N} \underbrace{a_n(Y_n(X_n w + w_0) - 1)}$$

  - The minus sign in front of the Lagrange multiplier is because while we're minimizing with respect to w, we're maximizing with respect to a

Distance from margin, which we want to maximize

Finding Hyperplane

$$L(w,a) = \frac{||w||^2}{2} - \sum_{n=1}^{N} a_n(Y_n(X_n w + w_0) - 1)$$

- As with all of our quadratic minimization/maximization problems we need to look at the derivatives
- Since we have two variables (a and w), let's take the derivative with respect to each (one at a time)
- Taking the derivative with respect to w (ignoring w_0, we'll do this in a second..) we get
  - $\frac{\partial}{\partial w} L(w,a) = w - \sum_{n=1}^{N} a_n Y_n X_n$
  - Setting this to zero and solving for w…

$$w = \sum_{n=1}^{N} a_n Y_n X_n$$

Finding Hyperplane

$$L(w,a) = \frac{||w||^2}{2} - \sum_{n=1}^{N} a_n(Y_n(X_n w + w_0) - 1)$$

- Taking it with respect to w_0 we get
  - $\frac{\partial}{\partial w_0} L(w,a) = \sum_{n=1}^{N} a_n Y_n$
- Therefore

$$\sum_{n=1}^{N} a_n Y_n = 0$$

Finding Hyperplane

- Let's expand the original formula
- $L(w, a) = \frac{1}{2}||w||^2 - \sum_{n=1}^{N} a_n(Y_n(X_n w + w_0) - 1)$
  - $= \frac{1}{2}||w||^2 - \sum_{n=1}^{N}(a_n Y_n X_n w + Y_n w_0 - a_n)$
  - $= \frac{1}{2}||w||^2 - (\sum_{n=1}^{N} a_n Y_n X_n w + \sum_{n=1}^{N} a_n Y_n w_0 - \sum_{n=1}^{N} a_n)$
- From our derivatives we know that at the minimum
  - $w = \sum_{n=1}^{N} a_n Y_n X_n$
  - $\sum_{n=1}^{N} a_n Y_n = 0$

Finding Hyperplane

- So lets substitute in $w = \sum_{n=1}^{N} a_n Y_n X_n$ into our original formula

  $L(w, a) = \frac{1}{2}||w||^2 - (\sum_{n=1}^{N} a_n Y_n X_n w + \sum_{n=1}^{N} a_n Y_n w_0 - \sum_{n=1}^{N} a_n)$
- $L(a) = \frac{1}{2}(\sum_{n=1}^{N} a_n Y_n X_n)^T(\sum_{n=1}^{N} a_n Y_n X_n)$ - $(\sum_{n=1}^{N} a_n Y_n X_n(\sum_{n=1}^{N} a_n Y_n X_n) + \sum_{n=1}^{N} a_n Y_n w_0 - \sum_{n=1}^{N} a_n)$
- $L(a) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i^T X_j - (\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i^T X_j + \sum_{n=1}^{N} a_n Y_n w_0 - \sum_{n=1}^{N} a_n)$
- $L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i^T X_j + \sum_{n=1}^{N} a_n Y_n w_0$

Finding Hyperplane

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i^T X_j + \sum_{n=1}^{N} a_n Y_n w_0$$

- We can also use the fact that at the minimum $\sum_{n=1}^{N} a_n Y_n = 0$ to eliminate the last term
- This results in

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i X_j^T$$

- We have transformed this into a minimization problem of only a single variable!

Finding Hyperplane

- Finally we take the derivative of this equation with regards to each of the variables a_n
- Setting these to zero we get a system of N equations
- After solving, all vectors (observations) associated with non-zero Lagrange multipliers are support vectors
- Using the solved Lagrange multipliers we can substitute back to solve for w since:

$$w = \sum_{n=1}^{N} a_n Y_n X_n$$

And then solve for w_0 since $|xw + w_0| = 1$ for all support vectors

Simple Example

- Find the support vectors for a simple case.
- You'll need:

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i X_j^T$$

$$w = \sum_{n=1}^{N} a_n Y_n X_n$$

- Follow this example….

$$X = \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Example

$$X = \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

- Writing the minimization problem out for $N = 4$ we have:
  - $L(a) = (a_1 + a_2 + a_3 + a_4) - ½(2a_1{}^2 + 0a_1a_2 - 0a_1a_3 + 2a_1a_4 + 0a_2a_1 + 2a_2{}^2 + 2a_2a_3 + 0a_2a_4 + 0a_3a_1 + 2a_3a_2 + 2a_3{}^2 + 0a_3a_4 + 2a_4a_1 + 0a_4a_2 + 0a_4a_3 + 2a_4{}^2)$
- Collecting terms…
  - $L(a) = (a_1 + a_2 + a_3 + a_4) - ½(2a_1{}^2 + 2a_1a_4 + 2a_2{}^2 + 2a_2a_3 + 2a_3a_2 + 2a_3{}^2 + 2a_4a_1 + 2a_4{}^2)$

$$\boxed{L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i X_j^T}$$

Example

$$X = \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

$L(a) = (a_1 + a_2 + a_3 + a_4) - ½(2\, a_1{}^2 + 2a_1a_4 + 2a_2{}^2 + 2a_2a_3 + 2a_3a_2 + 2a_3{}^2 + 2a_4a_1 + 2a_4{}^2)$

- Now let's take the derivative of this with regards to each variable
  - $\frac{d}{da_1} = 1 - \frac{1}{2}(4a_1 + 2a_4 + 2a_4) = 1 - 2a_1 - 2a_4 = 0$
  - $\frac{d}{da_2} = 1 - \frac{1}{2}(4a_2 + 2a_3 + 2a_3) = 1 - 2a_2 - 2a_3 = 0$
  - $\frac{d}{da_3} = 1 - \frac{1}{2}(4a_3 + 2a_2 + 2a_2) = 1 - 2a_3 - 2a_2 = 0$
  - $\frac{d}{da_4} = 1 - \frac{1}{2}(4a_4 + 2a_1 + 2a_1) = 1 - 2a_4 - 2a_1 = 0$
- Setting these equations equal to zero we get the following system of equations:
  - $2a_1 + 2a_4 = 1$ (twice)
  - $2a_2 + 2a_3 = 1$ (twice)

Example

$$2a_1 + 2a_4 = 1 \text{ (twice)}$$
$$2a_2 + 2a_3 = 1 \text{ (twice)}$$

- Since there's only 2 independent equations and 4 variables there are multiple solutions.
- However each should gives us the same hyperplane

$$w = \sum_{n=1}^{N} a_n Y_n X_n$$

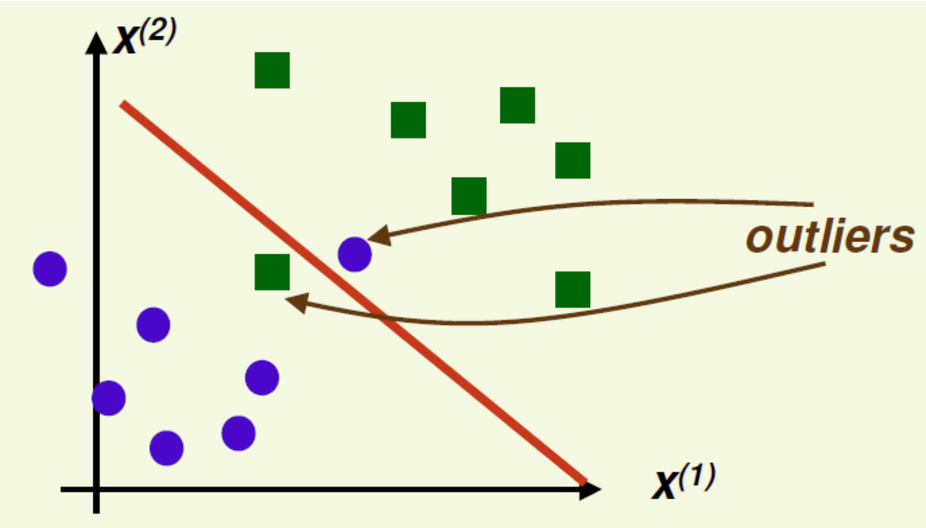
- $a_1 = 1/2, a_4 = 0, a_2 = 0, a_3 = 1/2$
  - Only the vectors associated with a_1 and a_3 are support vectors
  - $w = -1/2[-1,-1] + 1/2[1,-1] = [1,0]$
  - $|X_1 w| = |[-1,-1][1,0]^T| = 1$ so $w_0 = 0$
- $a_1 = a_2 = a_3 = a_4 = 1/4$
  - So all four vectors are support vectors
  - $w = -1/4[-1,-1] - 1/2[-1,1] + 1/2[1,-1] + 1/2[1,1] = [1,0]$
  - $|X_1 w| = |[-1,-1]\,[1,0]| = 1$ so $w_0 = 0$

Example

- Sanity check!
- All support vectors should have $|xw + w_0| = 1$
- All non-support vectors should have $|xw + w_0| > 1$

SVM: Non-Separable Case

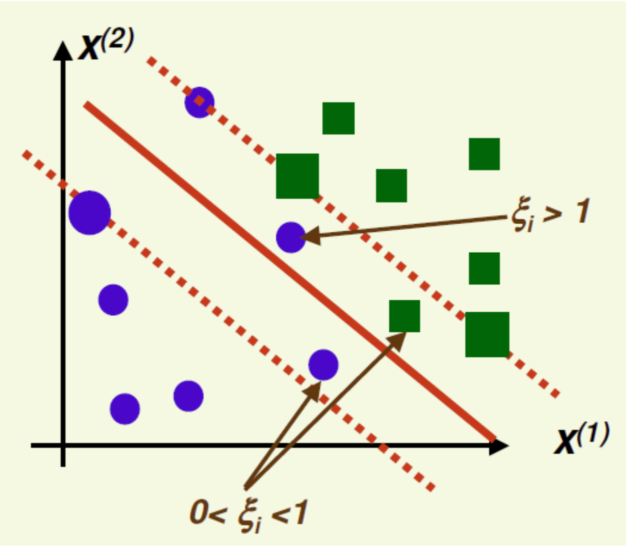
- Data is most likely to be not linearly separable
  - But linear classifier may still be appropriate
- There are basically to approaches to this issue:
  - Allow for some margin of error
  - Move the data to a *higher* dimensionality where it may be separable



Pattern Classification, Chapter 5

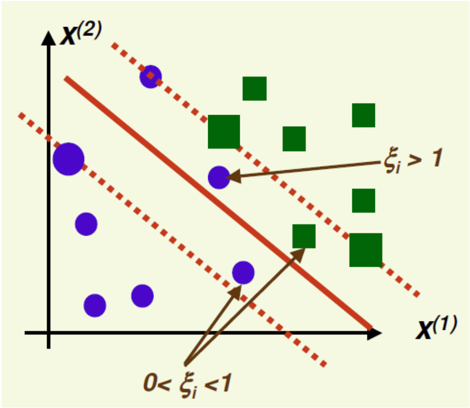
Non-Separable Linear SVM

- First we'll look at keeping the data in its current dimensionality but add a margin for error
  - Still the data should be "almost" linearly separable for good performance
- The solution uses *slack variables* $\xi_1, \ldots, \xi_N$ (one for each sample)
  - Which is essentially the amount of error (how far from the margin) for each sample given the current plane



Pattern Classification, Chapter 5

Non-Separable Linear SVM

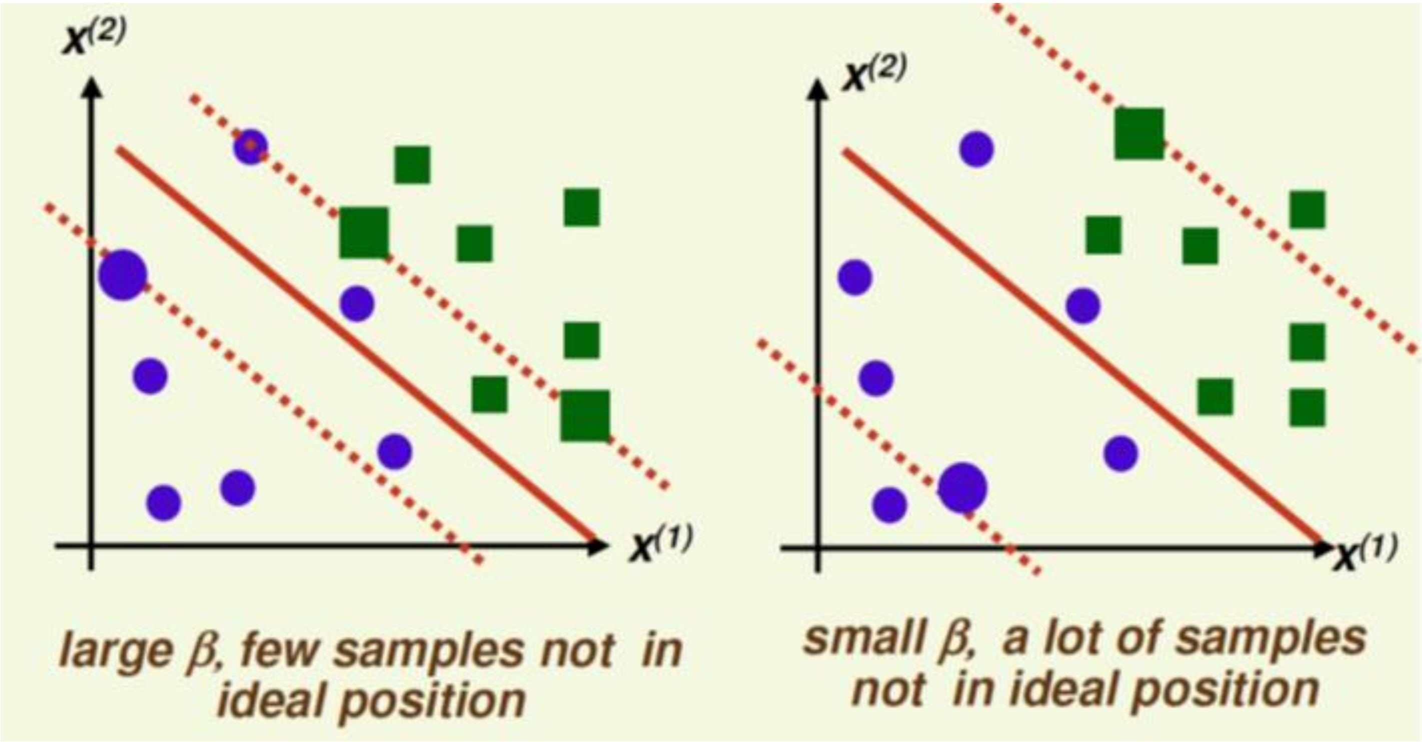


- So we minimize

$$J(w, \xi) = \frac{||w||^2}{2} + \beta \sum_{i=1}^{N} \xi_i$$

- Subject to:
  - $\forall i, Y_i(X_i w + w_0) \geq 1 - \xi_i$
  - $\forall i, \xi_i \geq 0$
- This can be interpreted as maximizing the width (although we inverted it to make it a minimization problem) while minimizing the amount of miss-classifications
- β is a constant that measure the relative weight of the first and second term
  - If β is small, we allow a lot of samples to be in not ideal positions (closer to linear SVM)
  - If β is large then the original non-ideal samples may be closer to the margin (better) but new ones may now be inside it.

SVM: Non-Separable Case



Finding Hyperplane

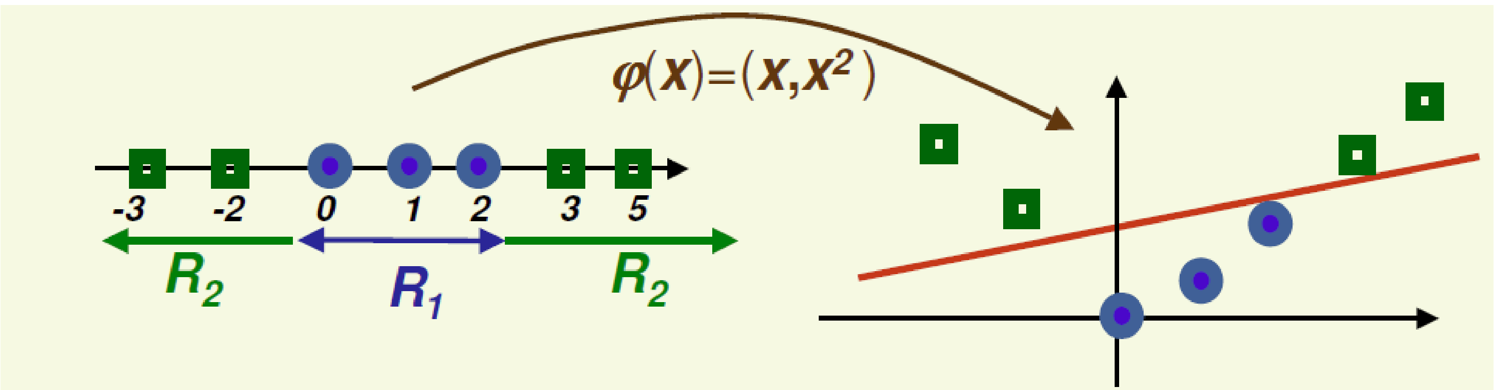
- If we add in our Lagrange multipliers, take the derivatives with respect to w, w_0 and ξ, and do substitution we get the equation:

$$L(a) = \sum_{n=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j X_i X_j^T$$

- Subject to
  - $0 \leq a_i \leq \beta$
  - $\sum_{i=1}^{N} a_i Y_i = 0$
- This is the same as our linearly separable case!
  - The only difference is the upper bound constraint on the values of a_i

Nonlinear SVM (kind of)

- SVMs define hyperplanes
  - So it's a linear classifier
- How can we deal with data that isn't linear separable?
  - We can try to deal with it in a higher dimension!
  - Doing so hopefully makes the data more separable



Non-Linear SVM

- Typically SVMs use the kernel trick to simulate computing the cosine similarity in higher dimensions without actually going to higher dimensions.
- Usually the choice is between a linear and an RBF kernel, the latter of which is non-linear
- As a rule of thumb
  - If the number of features is large (larger than number of observations) you may not need to map to a higher dimensional space therefore a linear kernel may be suffice
    - Plus it can become computationally expensive
  - Otherwise use RBF

Nonlinear SVM Step-by-Step

1. Start with data $X_1, \dots, X_n$ in D-dimensional space
2. Choose kernel κ
   - From this you should know its mapping function $\phi(x)$
3. Choose a blending value β (if you want)
4. Find maximum margin linear discriminant function in higher dimensional space by solving quadratic programming problem (use some package)

$$L(a) = \sum_{n=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j Y_i Y_j \kappa\left(X_i, X_j\right)$$

   - Constrained to $0 \le a_i \le \beta \; \forall i$ and $\sum_{i=1}^{n} a_i Y_i \; = 0$
   - This will give us S is the set of support vectors

$$S = \{X_i | a_i \neq 0\}$$

Nonlinear SVM Step-by-Step

5. Given new vector x, determine which class it belongs to using the linear discriminant function

$$g(x) = \phi(x)w + w_0$$

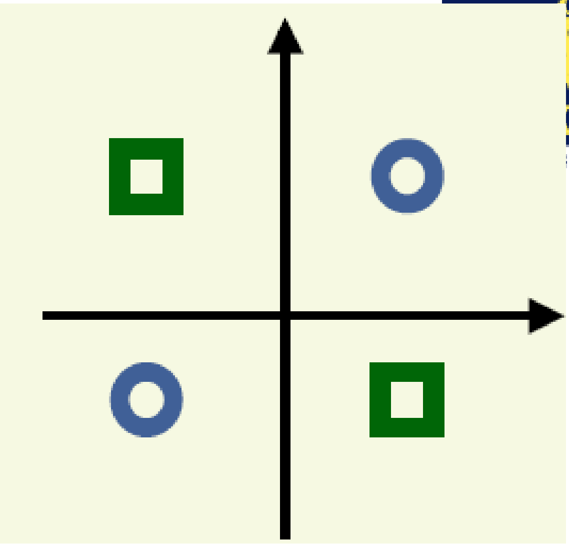
- However, due to the kernel trick we can actually just compute this using the support vectors and the kernel function without ever computing $\phi(x)$ or finding w or w_0

$$g(x) = \sum_{X_i \in S} a_i Y_i \kappa(X_i, x)$$

  - Note again we don't actually need to project x into higher dimensional space to do the computation

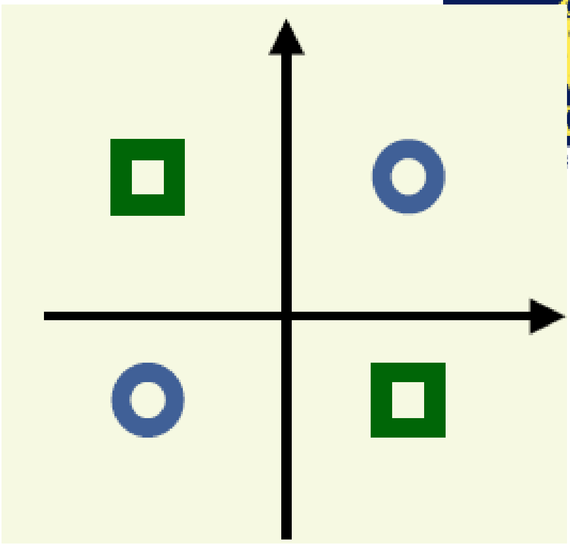
- $g(x) > 0$ class 1, otherwise class 2

SVM Example: XOR



- Class 1 (positive): X_1=[1,-1], X_2=[-1,1]
- Class 2 (negative): X_3=[1,1], X_4=[-1,-1]
- Kernel choice: Polynomial of degree 2
  - $\kappa(x,y) = (xy^T + 1)^2$
  - So $\phi(x) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2\right]$
- Need to maximize
  - $L_D(a) = \sum_{i=1}^{4} a_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4} a_i a_j Y_i Y_j \left(X_i X_j^T + 1\right)^2$
  - Constrained to $0 \leq a_i\ \forall i$ and $\sum Y_i \alpha_i = 0$
- Using a quadratic programming package we get:
  - $a_1 = a_2 = a_3 = a_4 = \frac{1}{8}$
  - All samples are support vectors since they are all non-zero

SVM Example: XOR



- Let's test our sample [1,1]
- Recall $\kappa(x, y) = (xy^T + 1)^2$
- $g(1,1) = \frac{1}{8}(+)\kappa([1,-1],[1,1]) + \frac{1}{8}(+)\kappa([-1,1],[1,1]) + \frac{1}{8}(-)\kappa([1,1],[1,1]) + \frac{1}{8}(-)\kappa([-1,-1],[1,1])$
- $= \frac{1}{8}(0+1)^2 + \frac{1}{8}(0+1)^2 - \frac{1}{8}(3)^2 - \frac{1}{8}(-2+1)^2$
- $= \frac{1}{8} + \frac{1}{8} - \frac{9}{8} - \frac{1}{8} = -1$

SVM Example: XOR Problem

- If we actually were interested in the weight vector we could compute it (albeit with the help of ϕ):

$$w = \sum_{i=1}^{n} a_i Y_i \phi(X_i)$$
$$= \frac{1}{8}(+)\phi(X_1) + \frac{1}{8}(+)\phi(X_2) + \frac{1}{8}(-)\phi(X_3) + \frac{1}{8}(-)\phi(X_4)$$

- Recall that we're using

$$\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]$$

- So with some arithmetic:

$$w = \left[0, 0, 0, -\frac{\sqrt{2}}{2}, 0, 0\right]^T$$

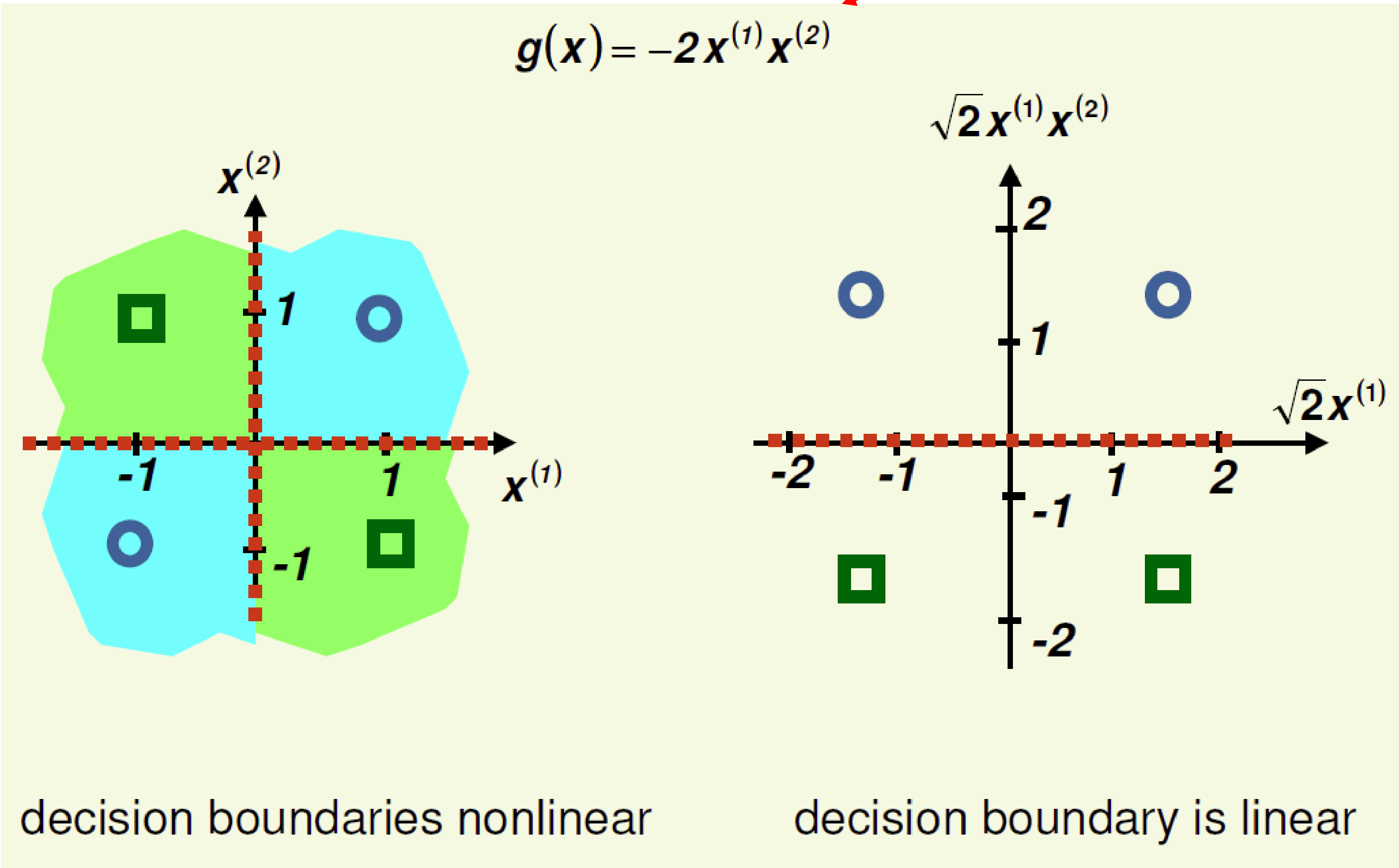
SVM Example: XOR Problem

- Let's find w_0
  - $|\phi(X_s)w + w_0| = 1$ for any support vector X_s
  - Let's use support vector $X_s = [1, 1]$
  - $\left[1,1,\sqrt{2},\sqrt{2},\sqrt{2},1,1\right]\left[0,0,0,-\frac{\sqrt{2}}{2},0,0\right]^T = -1$
  - So $w_0 = 0$
- Finally we can say

$$g(x) = \phi(x)w + w_0 = -x_1x_2$$

SVM Example: XOR Problem

Error: $g(x) = -x_1 x_2$



Matlab: svmtrain/svmclassify

- If you read the Matlab `fitcsvm` documentation you'll see that you can choose the kernel function type
  - And specify parameters for the chosen kernel
- In addition, after training your svm model contains:
  - `IsSupportVector` – A Boolean vector saying whether or not each observation is a support vector.
  - `Alpha` – Vector of weights for the support vectors
    - This is basically w
  - `Bias` – w_0
- You can then use `predict` to classify observations
  - This just gives you a class ID
  - If you want a distance, you need to compute it yourself using `Alpha` and `Bias`!

SVM Summary

- Pros
  - Strong theory
  - Good generalization
  - Global minima/maxima
- Cons
  - Directly applicable only to binary classification
  - Quadratic programming is expensive
  - Need to choose kernel

Final Observations

- Let's think about this algorithm
  - Supervised or non?
  - Classification or regression?
  - Model-based or instance-based?
    - When it comes time to test/use, are we using the original data?
  - Linear vs Non-Linear?
  - Can this work on categorical data?
  - Can this work on continuous valued data?
  - Training Complexity?
  - Testing Complexity?
  - How to deal with overfitting?
  - Directly handles multi-class?