Support Vector Machines

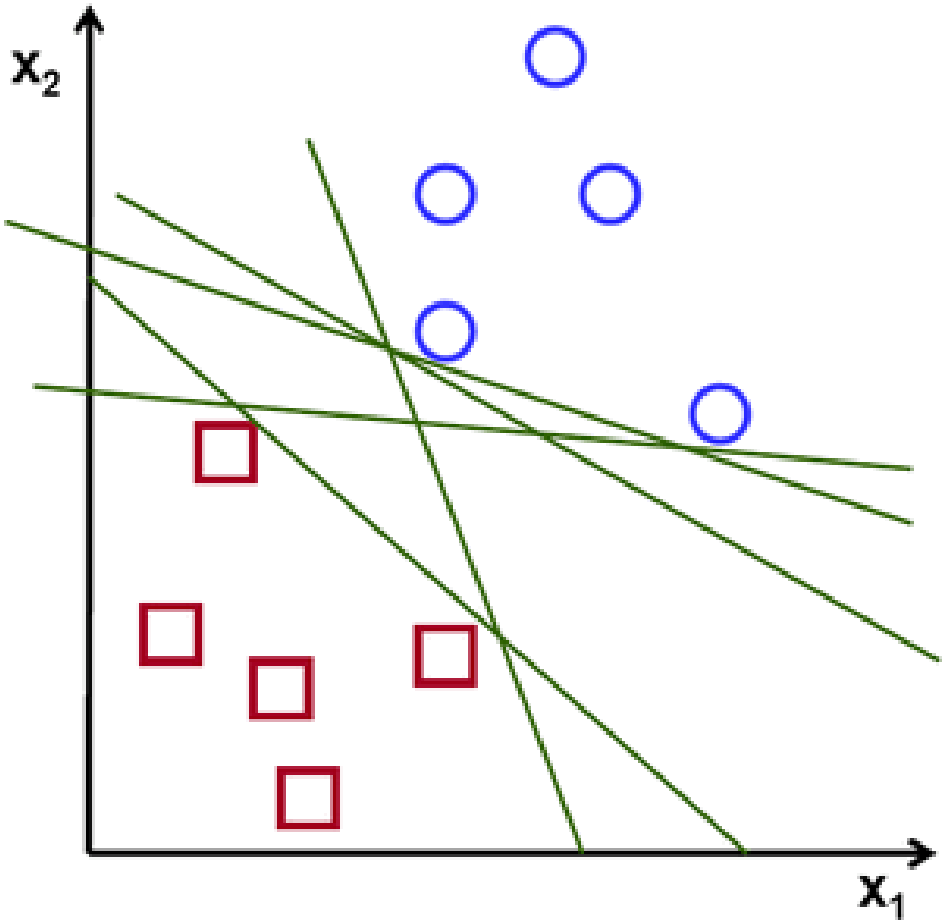
Programming Club

IIT Kanpur

pclub.iitk@gmail.com

March 11, 2016

Separating Hyperplanes



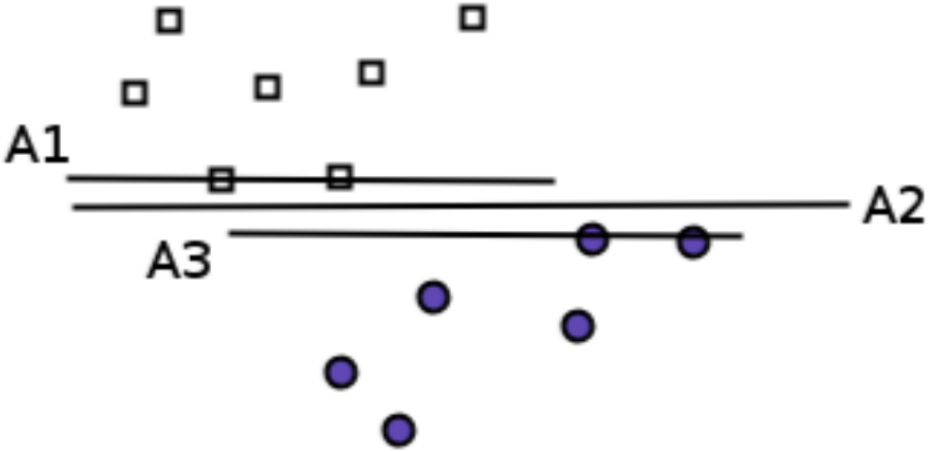
[0]Figure from [5]

Support Vector Machines (SVM)

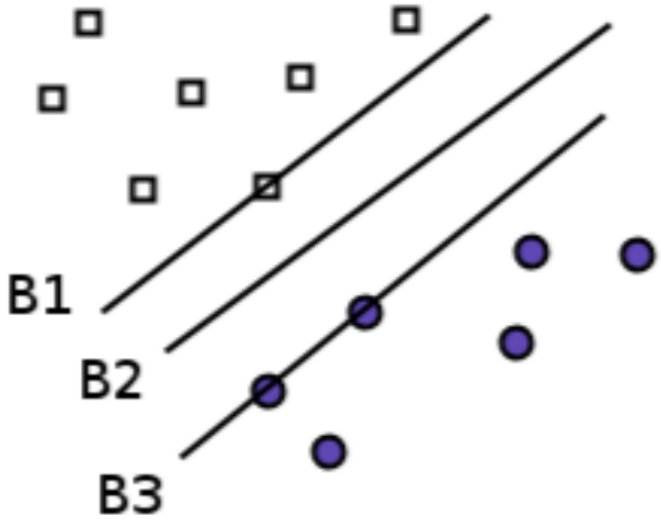
A binary classification problem.
Infinitely many separating hyper-planes.
Which is best?

[0]Figure from [1]

Support Vector Machines (SVM)



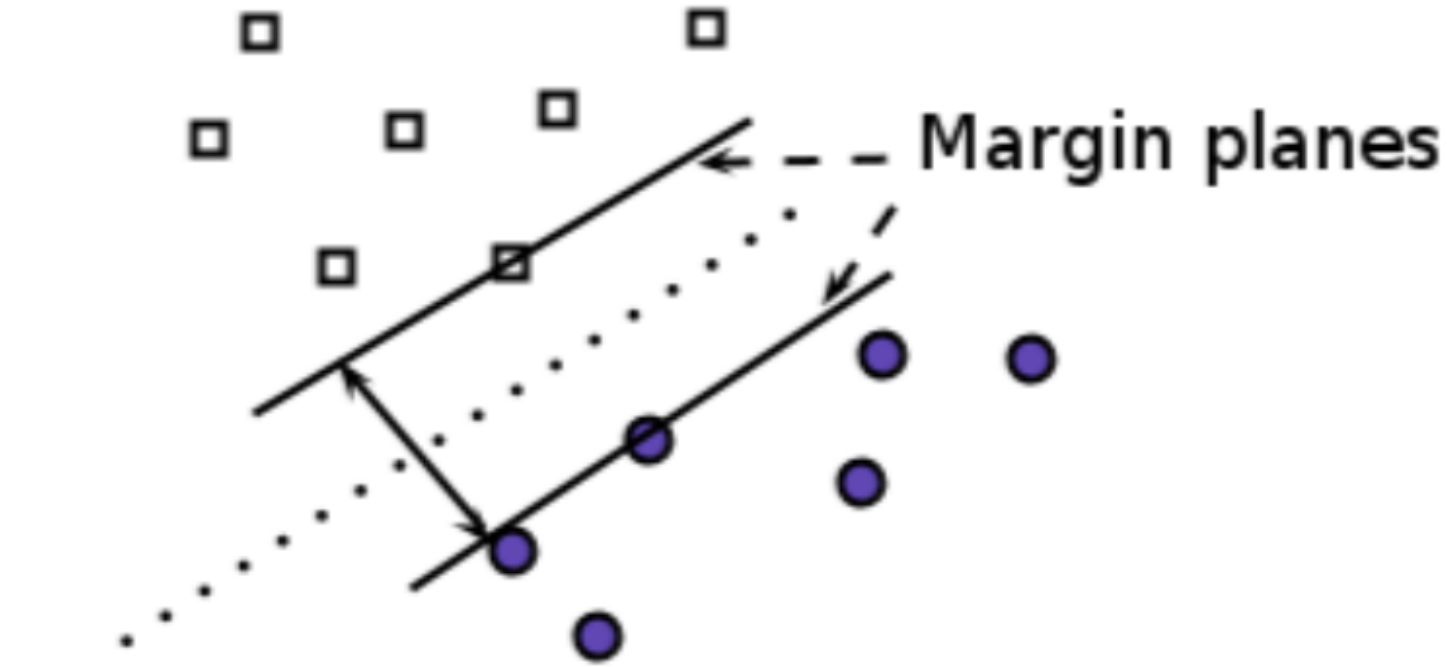
Which is the better separator amongst A1 to A3? A2



Which is the better separator amongst B1 to B3? B2

[0] Figure from [1]

Separating Hyperplane



[0]Figure from [1]

SVM - Separable Case

- The SVM classifier is a binary classifier that maximizes the margin.
- The separator hyperplane is parallel to the margin planes and is midway between the planes.
- Each margin plane passes through point(s) of the learning set that belong to a particular class and is closest to the margin plane of the other class. The distance between the margin planes is called the margin. Note that multiple pairs of margin planes are possible with different margins.
- The SVM algorithm finds the maximum margin separating hyperplane.
- The points from each class that determine the margin planes are called the support vectors (SVs).

[0] Credits to [1]

SVM - Optimization Objective

- We can set up an optimization problem by directly maximizing the margin (geometric margin).
- We want the classifier to be correct on all examples i.e. $y_t \theta^T x_t \geq \gamma$ $\forall\ t = 1 \ldots n$. Note that y_t is directly related to the true class.
- Subject to these constraints, we would like to maximize the margin i.e. maximize $\gamma / \|\theta\|$. Thus, we can alternatively minimize $\left(\|\theta\|/\gamma\right)^2$. Which gives us the following optimization problem,

$$\textit{minimize } \tfrac{1}{2}\|\theta/\gamma\|^2 \quad \textit{subject to} \quad y_t \left(\theta/\gamma\right)^T x_t \geq 1 \quad \forall\ t = 1 \ldots n$$

Which effectively becomes,

$$\textit{minimize } \tfrac{1}{2}\|\theta\|^2 \quad \textit{subject to} \quad y_t \theta^T x_t \geq 1 \quad \forall\ t = 1 \ldots n$$

[1] Inspired from [3]

SVM - Optimizing the Objective

- This is a convex (actually quadratic) optimization problem. Such problems have efficient algorithms to solve them. And an important property is that the local optimum is also the global optimum. So, we are guaranteed to find the optimal solution.
- We won't be going into the details. But further analysis requires knowledge of Lagrange Multipliers and KKT conditions.

SVM - Non Separable Case

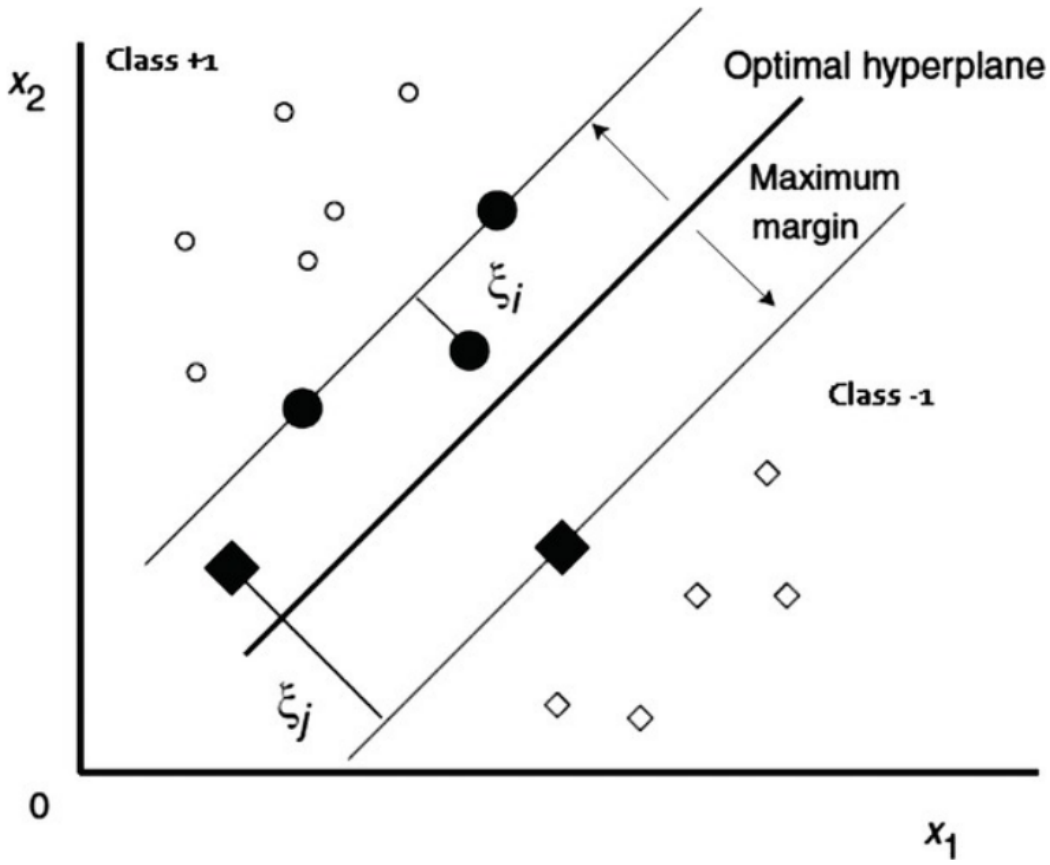


Figure: Misclassified, margin and regular points. $\xi_i \geq 0$ are called slack variables.(Figure from Wikipedia.)

[0]Figure from [1]

SVM - Non Separable Case

We change our constraint to $y_t\theta^T x_t \geq 1 - \xi_i$, $i = 1 \ldots n$ with $\xi_i \geq 0$. So, we have three types of vectors (Note that $g(\mathbf{x_i}) = \theta^T \mathbf{x_i}$) :

Regular Correctly classified vectors that are on or beyond the margin plane. Then $y_i g(\mathbf{x}_i) \geq 1$ and $\xi_i = 0$.

Margin Correctly classified vectors that are between the margin planes.Then $0 < y_i g(\mathbf{x}_i) < 1$ and $0 < \xi_i < 1$.

Misclassified Misclassified vectors that are on the wrong side of the hyperplane. Then $y_i g(\mathbf{x}_i) < 0$ and $\xi_i > 1$.

SVM - Optimization Objective

$$\min_{\mathbf{w},w_0,\bar{\xi}} \frac{\| \mathbf{w} \|^2}{2} + C\sum_{i=1}^{n} \xi_i$$

subject to:

$$y_i g(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1..n, \quad (\mathbf{x}_i, y_i) \in \mathcal{L}$$

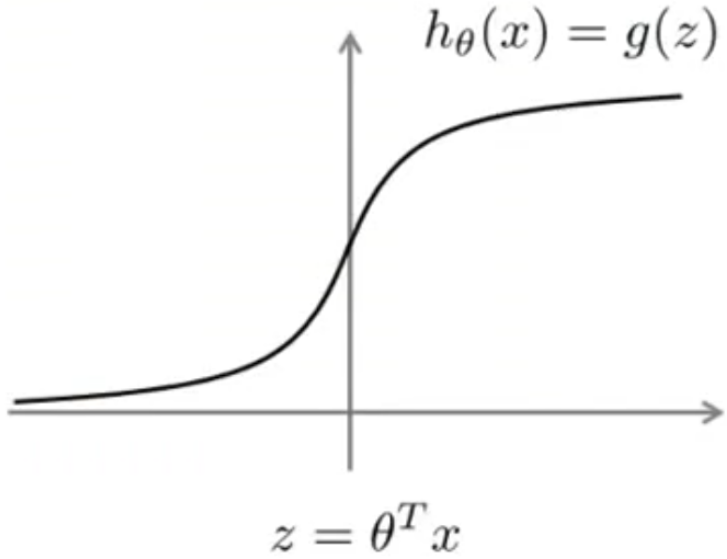
$$\xi_i \geq 0 \quad i = 1..n$$

- The second term $\sum_{i=1}^{n} \xi_i$ is called a regularizer and C is the regularization parameter. C balances between widening the margin and allowing misclassified and margin points.
- If we increase the penalty C for margin violations then at some point all $\xi_i = 0$ and we get back the maximum margin linear separator (If possible). On the other hand, for small C many margin constraints can be violated (Consider the case $C = 0$).

Review - Logistic Regression

- In logistic regression, our hypothesis is of the following form

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



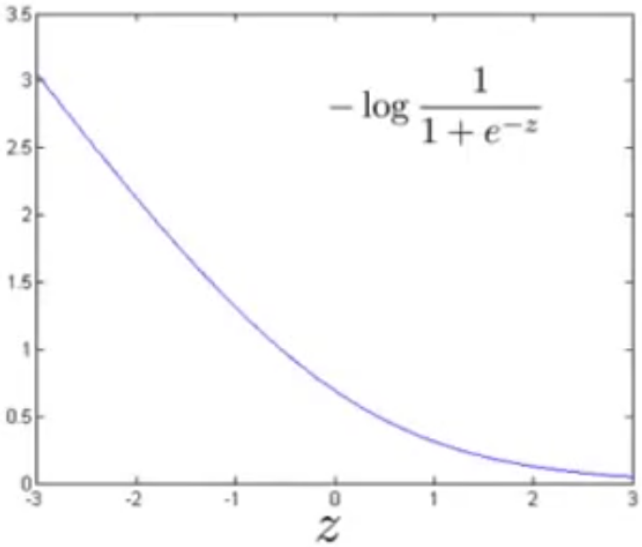
- If $y = 1$ we want $h_\theta(x) \approx 1,\ \theta^T x \gg 0$
- If $y = 0$ we want $h_\theta(x) \approx 0,\ \theta^T x \ll 0$

Logistic Regression - Cost Function

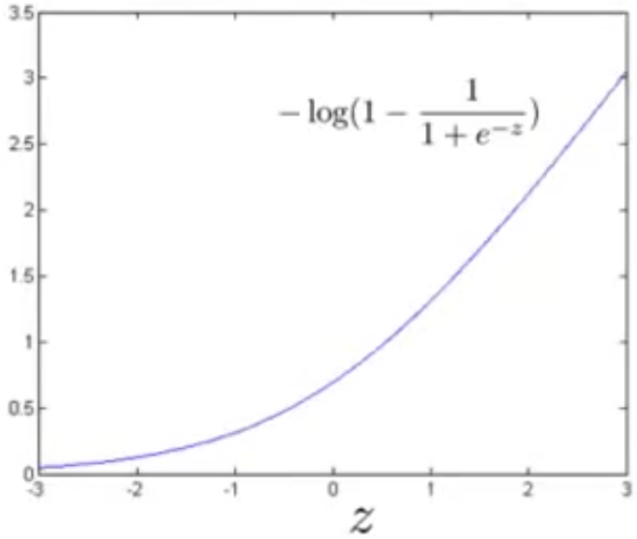
- The cost function for logistic regression is

$$J(\theta) = -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log(1 - \frac{1}{1+e^{-\theta^T x}})$$

- If $y = 1$, the cost function is



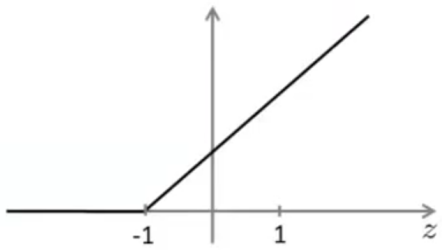
- If $y = 0$, the cost function is



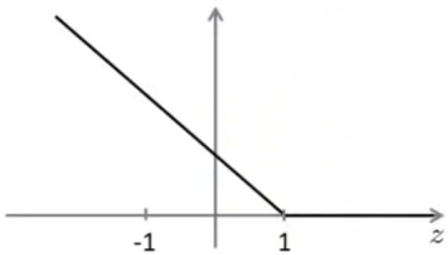
- Can we use simpler cost functions in place of the above functions that have similar nature?

SVM - Simplified Cost Function

$cost_0(z)$



$cost_1(z)$



$$\min_{\theta} \; C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

- The optimization objective can be simplified as follows for a large C

$$\min_{\theta} \; \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \quad s.t. \quad \begin{cases} \theta^T x^{(i)} \geq 1 & if \; y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 & if \; y^{(i)} = 0 \end{cases}$$

SVM - Simplified Cost Function

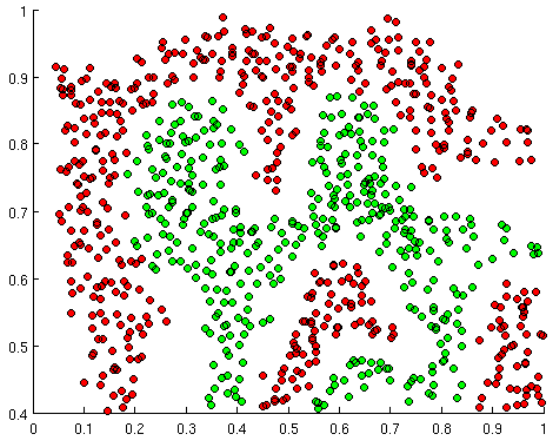
$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \quad s.t. \begin{cases} \theta^T x^{(i)} \geq 1 & if\ y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 & if\ y^{(i)} = 0 \end{cases}$$

$$\min_{\theta} \frac{1}{2}\|\theta\|^2 \quad s.t. \begin{cases} p^{(i)} \cdot \|\theta\| \geq 1 & if\ y^{(i)} = 1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & if\ y^{(i)} = 0 \end{cases}$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ

- This condition results in the decision boundary being chosen as the one with a large margin
- Thus, SVM is also referred to as a large margin classifier

Non-Linear Decision Boundaries



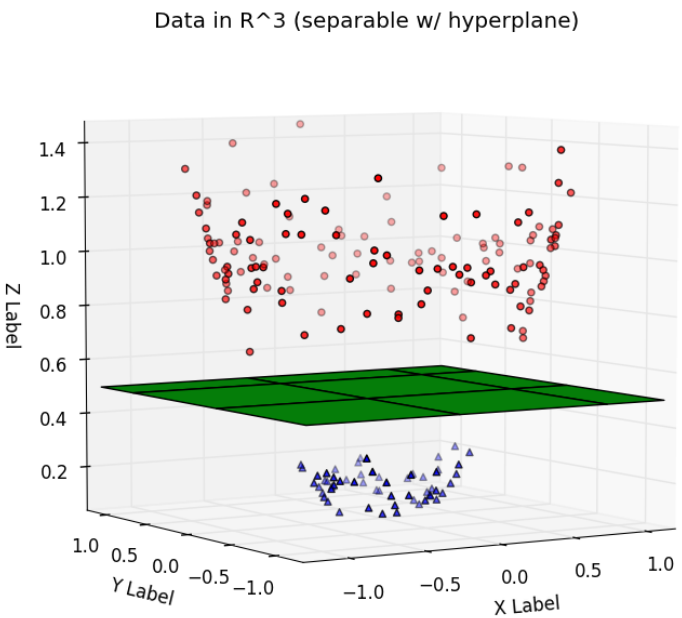
- We earlier discussed that to learn a non-linear boundary, we need higher order features

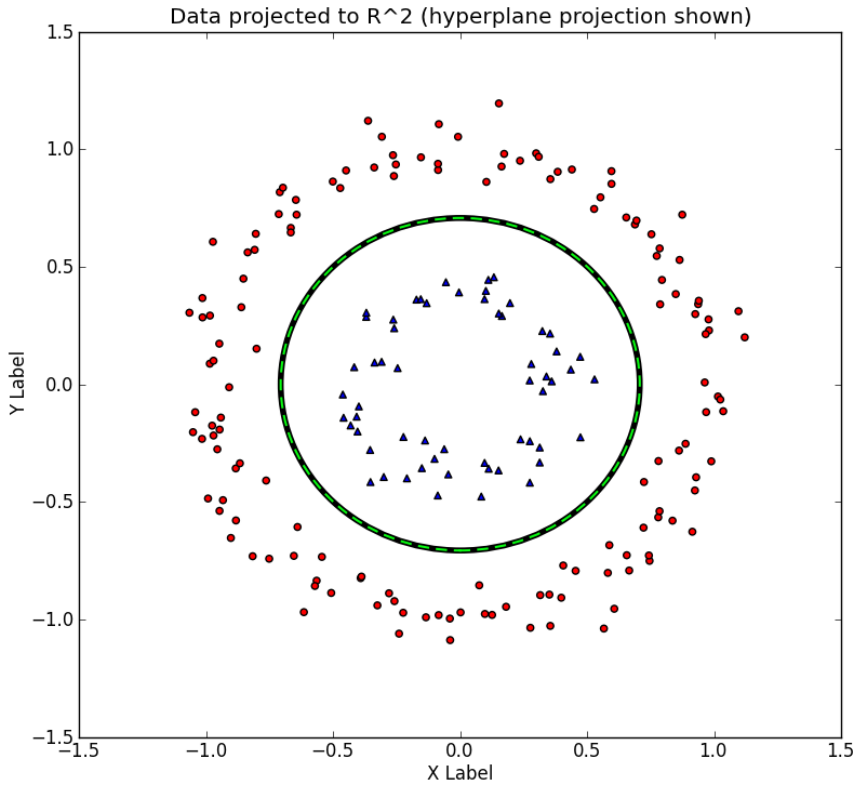
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots$$

- Problem - Hard to select which higher order features to use.

Solution?

- Generally the learning set is not linearly separable.
- **Solution:** Map the input space by a non-linear mapping to another feature space F (Usually of higher dimension). Find a linear separator in the higher order space. This would be equivalent to a non-linear separator in the original space.





Possible Issues

- Let the dimension of the original space d. Let the higher order space have dimension D. Usually, $D >> d$, D can even be infinite.
- Thus, calculations in the higher order space can be very expensive.
- **Solution:** Kernels!

Kernels

- **What is a Kernel:** A kernel is a similarity function. It is a function that takes two inputs (i.e. vectors) and returns how similar they are.
- Provide an **alternative solution** to the problem of Non-Linear Boundaries.
- Instead of defining new features (i.e. Transforming everything to a higher order space), we define a single kernel to compute similarity between the vectors. We provide this kernel, together with the vectors and labels to the learning algorithm, and finally we get a classifier (Wait for it).

[0] Inspired from [4]

Learning with Kernels

- But the algorithms we've discussed so far work with feature vectors, not Kernels. **What to do?**
- Here are a few Mathematical facts which come to our rescue,
  - **Mercer's Theorem:** Under some conditions, every kernel function can be expressed as a dot product in a (Possibly Infinite dimensional) feature space.
  - Many machine learning algorithms can be expressed entirely in terms of **Dot-Products** (Including SVM)
- **What does it mean?** Take our machine learning algorithm $\Rightarrow$ Express in terms of Dot-Products $\Rightarrow$ Replace the Dot-Product by a kernel (Since the kernel is also a Dot-Product in some space)

[0] Inspired from [4]

Why Kernels?

- Why prefer Kernels over Feature Vectors?
- As discussed earlier, computing the kernel is easy but computing the feature vector (And then computing the distances/similarity) corresponding to the kernel can be really hard.
- E.g. **RBF kernel** $k(x, y) = e^{-\|x-y\|^2}$, the corresponding feature vector is infinite dimensional. Yet, computing the kernel is almost trivial.
- **Another Example:** Let $x = (x_1, x_2, x_3),\ y = (y_1, y_2, y_3)$. Then for the function $f(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$, the kernel is $K(x, y) = (x \cdot y)^2$.

SVM with Kernels

- Assume that, after solving the earlier objective we get the **optimal** hyperplane, $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i y_i x_i$
- Thus, we classify a new point (vector) as $sign(\mathbf{w}^{*T}\mathbf{x})$, which becomes, $sign((\sum_{i=1}^{n} \alpha_i y_i x_i)^T \mathbf{x}) = sign(\sum_{i=1}^{n} \alpha_i y_i (x_i^T \mathbf{x}))$
- Note that $(x_i^T \mathbf{x})$ represents a **dot product**.
- Using kernels, we first change the **SVM objective** (It doesn't change much (almost no change), trust me!).
- Next, we slightly change how to **infer** labels. The label is now given as $sign(\sum_{i=1}^{n} \alpha_i y_i \langle \phi(x_i), \phi(\mathbf{x}) \rangle)$. Here $\phi()$ is the feature function. This can simply replaced by any appropriate kernel.

Why SVMs?

- Regularization parameter, which helps avoiding over-fitting.
- Can easily use the Kernel trick. Thus, able to build in expert knowledge about the problem via engineering the kernel.
- Defined by a convex optimization problem (no local minima), for which there are efficient methods.
- Lots of theory around it which supports its awesomeness. Theoretically proven error bounds.

References


CS771: Dr. Harish Karnick.
http://www.cse.iitk.ac.in/users/hk/cs771/.


Eric: Kernel trick.
http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html.


MIT OCW support vector machines.
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec3.pdf.


Quora: Kernels in ML.
https://www.quora.com/What-are-Kernels-in-Machine-Learning-and-SVM.


Separating lines.
http://docs.opencv.org/2.4/_images/separating-lines.png.

Questions?