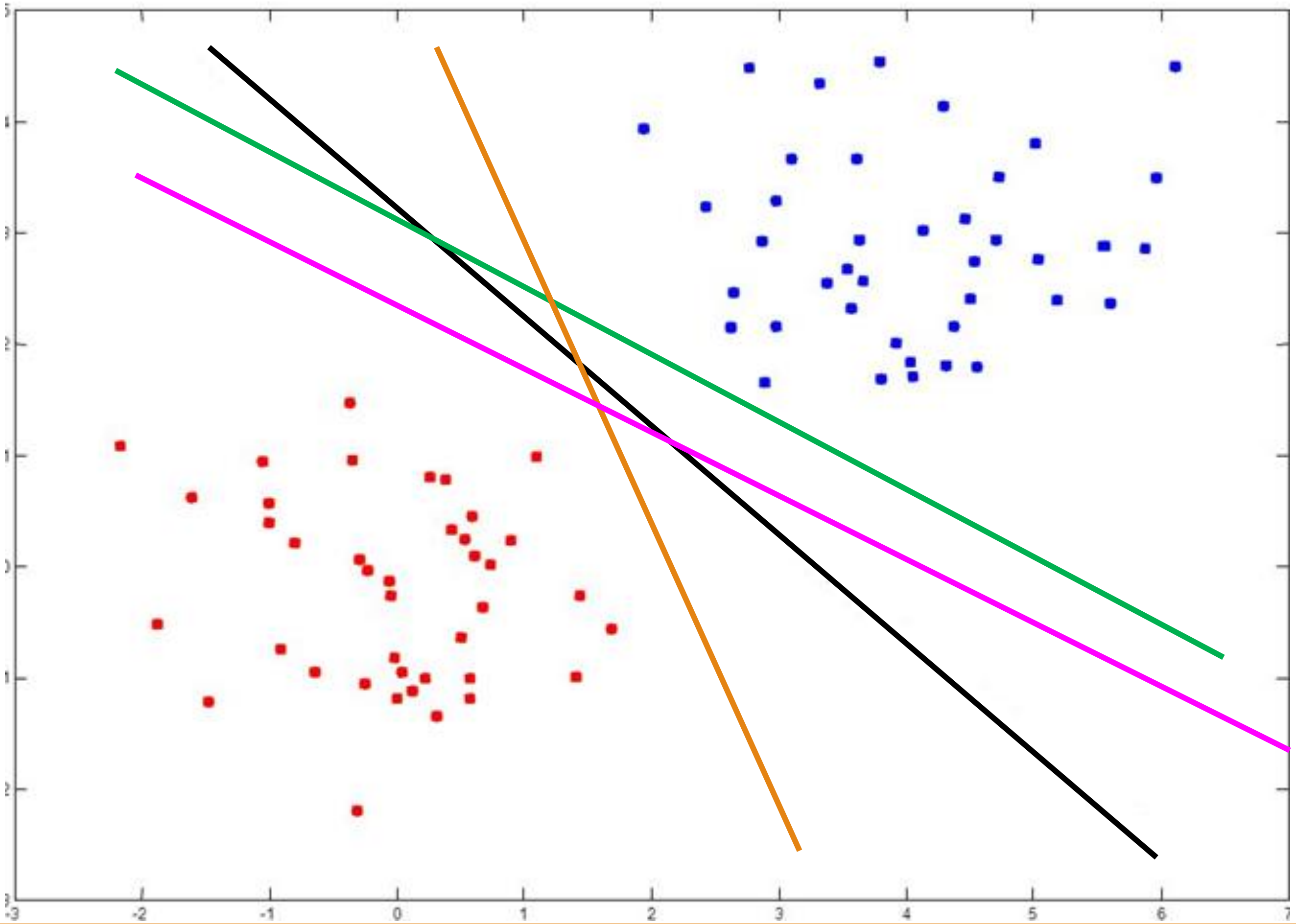
ML-101:
SVM, Unsupervised Learning

BY SARTHAK CONSUL

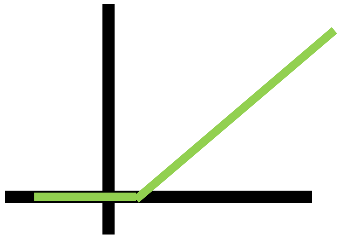
Linear Classifiers

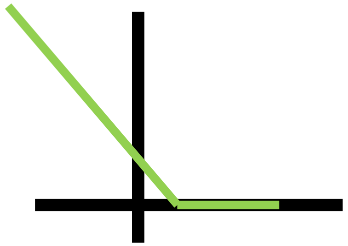


Support Vector Machines (SVMs)

- A max margin classifier
- Approximating loss in logistic regression with ReLU

$$J(\theta) = \frac{1}{\cancel{m}} \sum_{i=1}^{m} -y^{(i)} \cancel{\log h_\theta(x^{(i)})} - (1 - y^{(i)}) \cancel{\log(1 - h_\theta(x^{(i)}))}$$

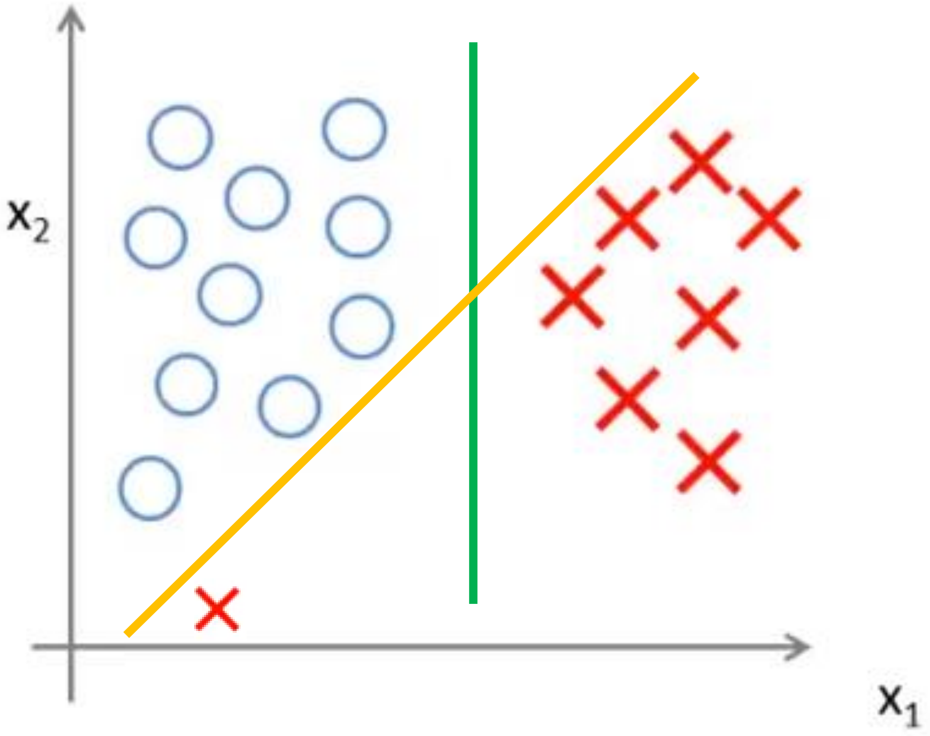




$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

SVMs contd.

- C is like $1/\lambda$



Large C
Small C

Kernels

- The SVM is linear in some implicit higher dimensional space
- This higher dimensional space need not be explicitly obtained
- Given x, compute new features w.r.t. landmarks $l^{(1)}, l^{(2)}, l^{(3)}$
- f_i =similarity(x,$l^{(i)}$) = $e^{-\left|\left|x-l^{(i)}\right|\right|^2/2\sigma^2}$ GAUSSIAN KERNEL
- Choosing landmarks
  - At training examples

$$\min_\theta C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1-y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2}\sum_{j=1}^{m}\theta_j^2$$

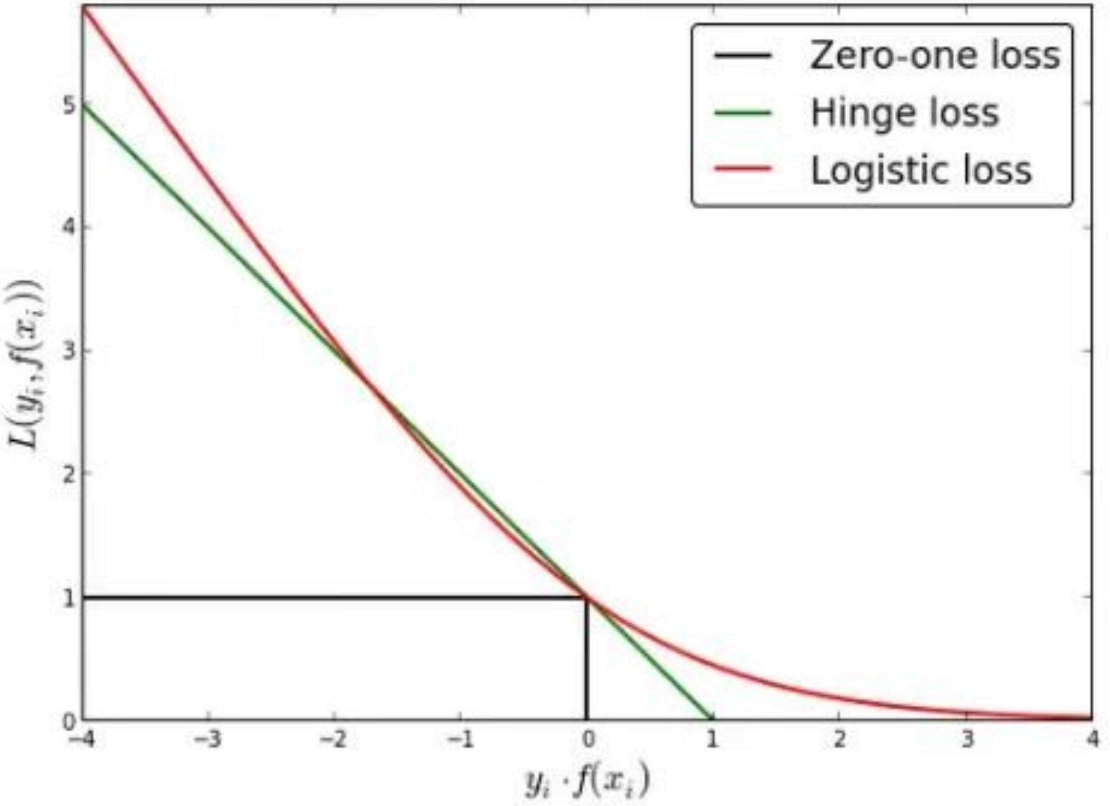
SVM Parameters

- C (=1/λ)
  - Large C: lower bias, higher variance
  - Smaller C: higher bias, lower variance

- σ
  - Large: Smoother f – higher bias, lower variance
  - Small: f changes abruptly – lower bias, higher variance

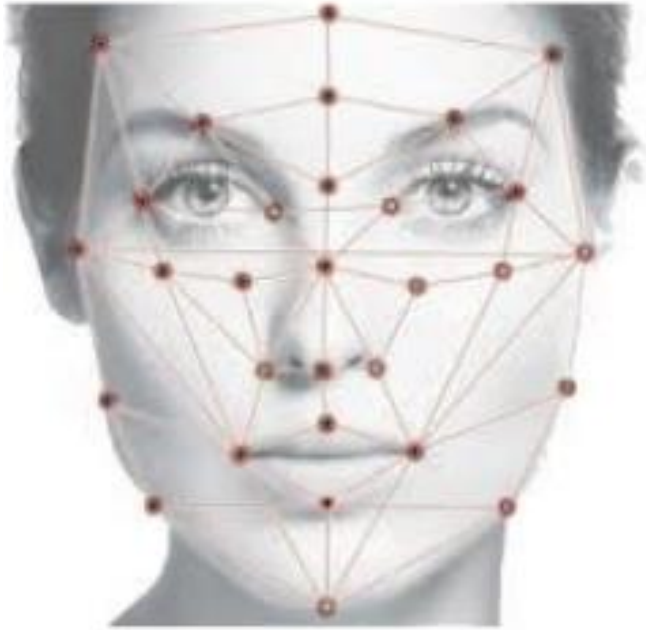
Linear SVM vs Logistic Regression

- Logistic loss does not go to zero even if the confidently.
- Logistic loss diverges faster than hinge loss
  - So, in general, it will be more sensitive to outliers
  - BOTH SENSITIVE
- Logreg. has a probabilistic interpretation



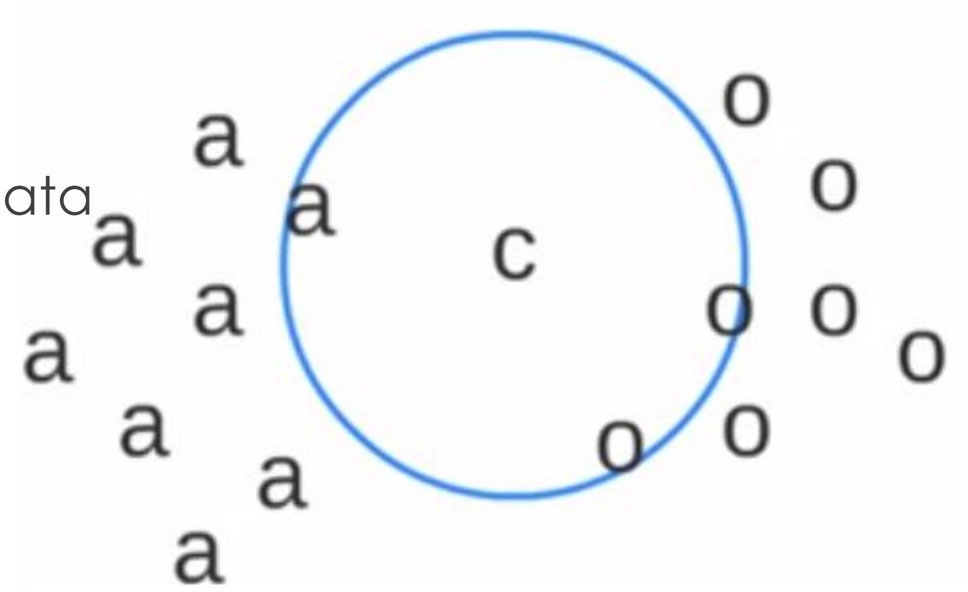
Applications of SVMs

- Face Detection
- Handwritten digit recognition
- Protein Recognition from mRNA sequence
- Text Categorization



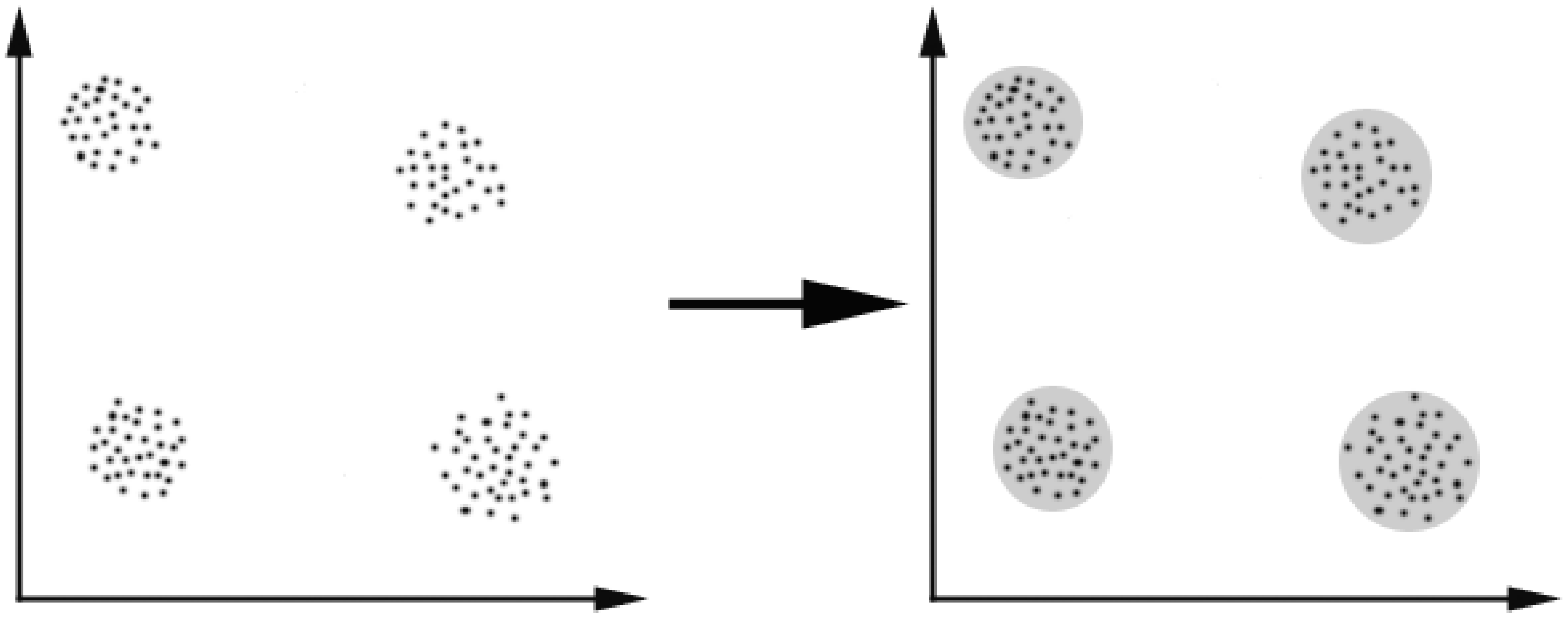
K-Nearest Neighbours (kNN)

- **Supervised Learning** Algorithm (used for classification and sometimes even for regression)
- think of curling
  - For k=3, Class of c will be ‘o’
- No training time: simply remembering the data
- Testing Time complexity!



Unsupervised Learning

- No label on data



Clustering

Applications Unsupervised Learning

- Market Segmentation
- Population Demographics
- Social Network Analysis
- Organising Computer Clusters
- Astronomical Data Analysis
- …

k-Means

- k is the no. of clusters (Input)
- Training set is $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ where $x^{(i)} \in \mathbb{R}^n$
- Goal: To label all m points into k clusters (centroids are $\mu^{(i)}$)
- Objective Function (Metric of model Distortion) – Squared distance error

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(k)}, \mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(1)} - \mu_{c^{(i)}}||^2$$

k-Means Algorithm

Randomly initialize the k cluster centroids $\{\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(m)}\}$, $\mu^{(i)} \in \mathbb{R}^n$

Repeat{

For i=1 to m

c(i):=closest cluster centroid to x(i) //Range of {1 to k}

For i=1 to k

$\mu^{(k)}$ = average of all points assigned to k^{th} cluster

}

How to randomly initialize the centroids?

- K<m (obviously)

Idea: Randomly pick the k training points as cluster centroids

```
For i=1 to 100{
      Randomly initialize k-Means
      Run k-Means Algorithm
      Compute J(c,μ)
}
Pick model with least distortion.
```

WARNING

Convergence to local Optima

- Another example of ensemble

Conclusion

- This is just the beginning
- Current techniques are far from human learning
- True thinking/ imagination in AI
- Adversarial Examples
- Responsible usage of ML