Lecture 16: SVM(contd.)

15 March 2024 10:58

Recap:

Primal:

$$\min \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1 \quad \forall i = 1, \ldots n$$

Dual: (Easier problem to solve for $d >> n$)

$$\max_{\lambda \geq 0} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum\sum \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{s.t. } \sum_{i=1}^{n} \lambda_i y_i = 0$$

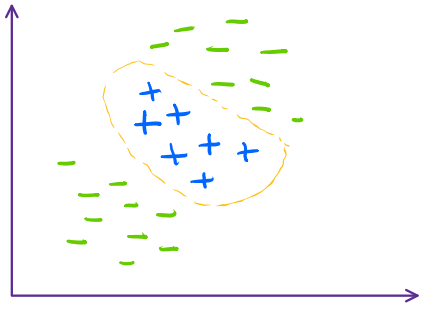
$x_i \in \mathbb{R}^d$, $d >> n$

$$w^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i$$

Why dual?
- less costly to solve
- Kernalization.

Recall that for a classification problem as below, we need to use a basis, ϕ i.e., $x_i \longmapsto \phi(x_i)$. But x_i are already in higher dimension, and $\phi(x_i)$ is even higher dimension.

So to computationally manage this, we use Kernalization.



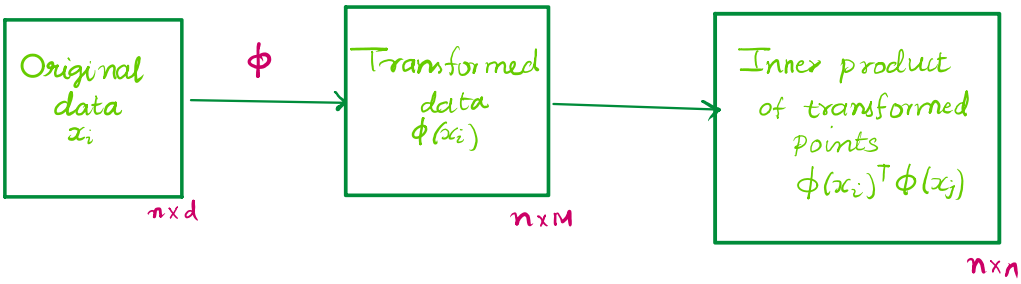
Given: linearly inseparable data

Goal: Project the data to a higher dimensional space
→ solve using SVM → find w, b in new dimension

Eg: Step 1:

$$x_i = \begin{pmatrix} x_i^{(1)} \\ x_i^{(2)} \end{pmatrix} \xrightarrow{\phi} \begin{pmatrix} 1 \\ x_i^{(1)} \\ x_i^{(2)} \\ x_i^{(1)} x_i^{(2)} \\ x_i^{(1)2} \\ x_i^{(2)2} \end{pmatrix} = \phi(x_i)$$

Step 2: Dual SVM: Objective fn has an inner product of the data in higher dimension.



Q: Do there exist fns that calculate the inner product in the transformed space without explicitly computing the transformations?

A: Yes, such fns are Kernal functions.

From the previous example, the terms involved = $\{1, x_i^{(1)}x_j^{(1)}, x_i^{(2)}x_j^{(2)}, x_i^{(1)}x_i^{(2)}x_j^{(1)}x_j^{(2)}, x_i^{(1)2}x_j^{(1)2}, x_i^{(2)2}x_j^{(2)2}\}$

Now consider the expression $(1+x_i^T x_j)^2$. This has exactly same terms as before

↳ KERNAL FUNCTION

Kernal trick: transformations are equivalent as long as we are calculating the dual of SVM.

Different Kernals:

i. Linear: $K(x,z) = x^T z$

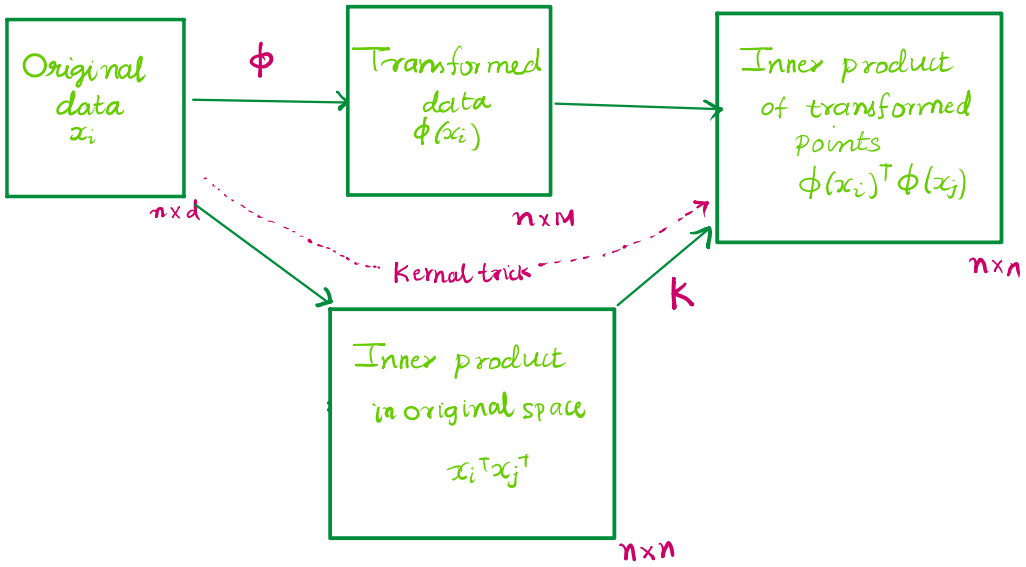
ii) Polynomial: $K(x,z) = (1+x^T z)^n$

iii) Gaussian Kernal: $K(x,z) = e^{-\|x-z\|^2/2\sigma^2}$

iv) Laplace / Radial: $K(x,z) = e^{-\|x-z\|/2\sigma}$

Uses of SVMs: Handwriting recognition, protein structure, medical image classification

A set of necessary and sufficient conditions govern the kernal functions.



The choice of Kernal to be used can be done using grid search.

Limitations of SVM

i) Binary Classification : One v/s rest classifier

ii) No probabilistic interpretation

iii) Does not work well when data is noisy.

What we had done so far is SUPERVISED LEARNING where $D = \{(x_i, y_i)\}_{i=1,\dots n}$

We will now deal with UNSUPERVISED LEARNING where $D = \{(x_i)\}_{i=1,\dots n}$

CLUSTERING: Unsupervised Learning

$$D = \{x_1, \dots x_n\} \quad , x_i \in \mathbb{R}^d$$

Goal: find a well-separated partition of data, $D = D_1 \cup D_2 \dots \cup D_k$ (hyperparameter), $D_i \cap D_j = \phi$
(hard clustering)

K-means Clustering

Consider a function C as below:

$C(i) = \tilde{k}$

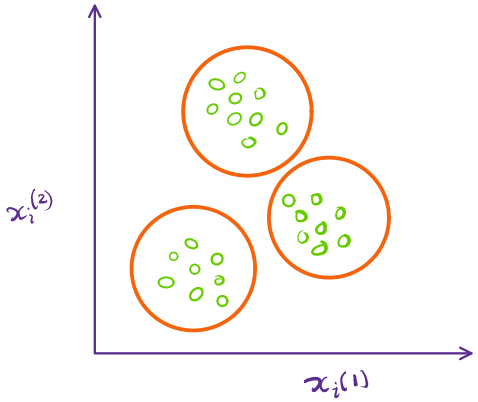
i^{th} data point

Cluster no.

Clustering function

$C : [n] \to [k]$

$[n] = \{1, 2, \dots n\}$



Algorithm:

- Input: $D = \{x_1, \dots x_n\}$
- Initialize: k cluster means $m_1, \dots m_k$ some arbitrary c^0
- Repeat until convergence (until assignments don't change):
  for every $i = 1, 2, \dots n$
  - if $\exists k' \neq c^t(i)$ such that $\|m^t_{k'} - x_i\| < \|m^t_{c^t(i)} - x_i\|$:

    ($\|\cdot\|$: distance metric; $m^t_{k'}$: cluster centre of k at time t.)

    $c^{(t+1)}(i) \leftarrow k'$

  - Update cluster centres by average of its current associated datapoints : $m^{t+1}_{k'}$

K-means clustering as an optimization problem:

$$\min_{c \in \mathcal{C}} \sum_{i=1}^{n} \|x_i - m_{c(i)}\|^2 \longrightarrow \text{NP-Hard!}$$

non-convex f^n

$\mathcal{C}$: Set of all possible clusters.

$$|\mathcal{C}| = k^n$$

Convergence:

Lemma: $\operatorname{argmin}_x \sum_{i=1}^{n_k} \|x_i - x\|^2 = \bar{x} = \frac{1}{n_k} \sum_{i=1}^{n} x_i$

Theorem: k-means converges to a local minima

- consider $t \to t+1$
- $SE(c^{t+1}, m^{t+1}) < SE(c^t, m^t)$ — SE : squared error

Why is this sufficient? → no clusters are repeated
→ finite number of clustering

Proof: $(m^t, c^t) \longrightarrow (m^t, c^{t+1}) \longrightarrow (m^{t+1}, c^{t+1})$

i) first we will prove $SE(c^{t+1}, m^t) < SE(c^t, m^t)$ → from algorithm itself

ii) Now show that $SE(c^{t+1}, m^{t+1}) \leq SE(c^{t+1}, m^t)$ → from Lemma

Combining (i) and (ii)

ii) Now show that $SE(c^{t+1}, m^{t+1}) \leq SE(c^{t+1}, m^{t})$

Combining (i) and (ii) $SE(c^{t+1}, m^{t+1}) < SE(c^{t}, m^{t})$