Support Vector Machines

⇒ SVMs are among the best "off-the-shelf" supervised learning algorithm.

★ Margins

⇒ Consider logistic regression, where the probability $P(y=1|x;\theta)$ is modeled by $h_\theta(x) = g(\theta^T x)$

If $h_\theta(x) \geq 0.5 \longrightarrow y=0$ (Prediction)

or

$$\theta^T x \geq 0$$

⇒ The larger $\theta^T x$ is, the larger also is $h_\theta(x)$ and thus also the higher our degree of "confidence" that the label is 1.

⇒ Given a training set, we'd have found a good fit to the training data if we find θ so that:

$$\theta^T x^{(i)} \gg 0 \quad \text{if } y^{(i)}=1$$

&

$$\theta^T x^{(i)} \ll 0 \quad \text{if } y^{(i)}=0$$

★ Notation

⇒ We will be considering a linear classifier for a binary classification problem with labels y and features x.

⇒ We will use $y \in \{-1, 1\}$ to denot class labels.

⇒ Also, so ather than parameterizing our linear classifier with the vector θ, we will use parameters w, b and write our classifier as

$$h_{w,b}(x) = g(w^T x + b)$$

⇒ Here, $g(z) = 1$ if $z \geq 0$ & $g(z) = -1$ otherwise.

★ <u>Functional and geometric margins</u>

Functional margin

→ Given a training example $(x^{(i)}, y^{(i)})$, we define the functional margin of (w, b) with respect to the training example

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

→ Given g, if we replace w with $2w$ & b with $2b$, then since $g(w^T x + b) = g(2w^T x + 2b)$ this would not change $h_{w,b}(x)$ at all.

→ So $h_{w,b}(x)$ depend only on the sign, but not on the magnitude of $w^T x + b$.

→ However, replacing (w, b) with $(2w, 2b)$ also results in multiplying our functional margin by 2.

↳ It might therefore make sense to impose some sort of normalization condition.

$$(w,b) \Longrightarrow \left(\frac{w}{\|w\|_2}, \frac{b}{\|b\|_2}\right)$$

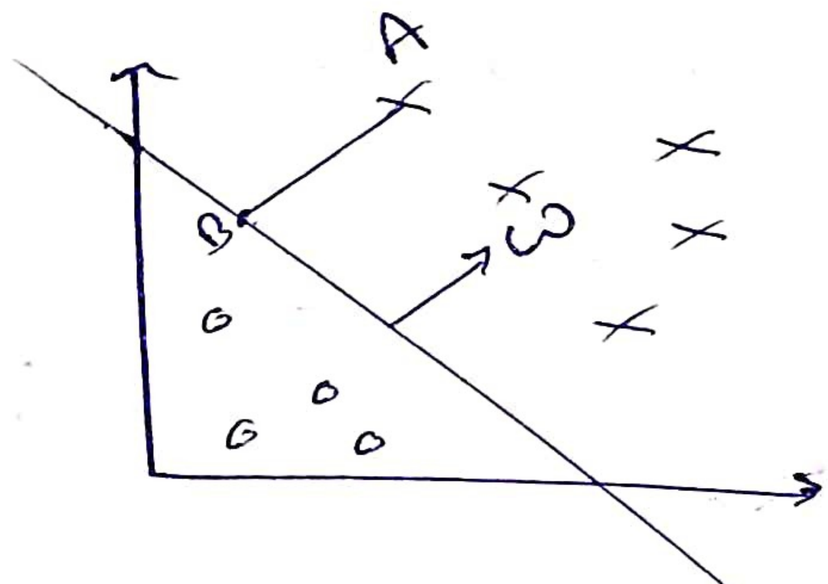
⇒ Given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \dots m\}$, we also define the functional margin (w,b) with respect to S as the smallest of the functional margin of the individual training examples.

$$\hat{\gamma} = \min_{i=1\dots m} \hat{\gamma}^{(i)}$$

Geometric margin

⇒ Note: w is orthogonal to the separating hyperplane.

⇒ Let A represent the input $x^{(i)}$ and $y^{(i)} = 1$



⇒ $\gamma^{(i)}$ distance of A from decision boundary. Given by AB.

$$\left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|_2}\right) \rightarrow \text{Point B}$$

$$w^T\left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|_2}\right) + b = 0 \quad \left\{\begin{array}{l}\text{Point B lies} \\ \text{on the hyperplane}\end{array}\right\}$$

$$\gamma^{(i)} = \left(\frac{w}{\|w\|_2}\right)^T x^{(i)} + \frac{b}{\|w\|} \quad \left\{\text{For positive training Example}\right\}$$

⇒ More generally, we define the geometric margin of (w,b) with respect to a training example $(x^{(i)}, y^{(i)})$ to be

$$\boxed{\gamma^{(i)} = y^{(i)}\left(\left(\frac{w}{\|w\|_2}\right)^T x^{(i)} + \frac{b}{\|w\|_2}\right)}$$

⇒ If we replace w with $2w$ and b with $2b$, then the geometric margin does not change.

⇒ Finally, given a training set $S = \{(x^{(i)}, y^{(i)}) ; i = 1, \dots m\}$, we also define the geometric margin of (w,b) with respect to S to be the smallest of the geometric margins on the individual training examples:

$$\boxed{\gamma = \min_{i=1\dots m} \gamma^{(i)}}$$

★ The Optimal margin classifier

⇒ Given a training set, try to find a <u>decision boundary</u> that maximizes the (geometric) <u>margin</u>, since this would reflect a very confident set of predictions on the training set.

⇒ For now, we will assume that we are given a training set that is linearly separable.

↳ It is possible to separate the positive & negative example using some separating hyperplane.

⇒ To achive the maximum geometric margin, we can pose the following optimization Problem:

$$\max_{\gamma,w,b} \gamma$$

$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \gamma \quad i = 1, \dots m$$

$$\|w\| = 1$$

⇒ If we could solve the optimization problem above, we'd be done.

↳ But the $\|w\|_2 = 1$ constraint is non-convex one, and this problem certainly isn't in any format that we can plug into standard optimization software to solve.

⇒ So, lets try transforming the problem into a nicer one.

⇒ Consider,

$$\max_{\gamma,w,b} \frac{\hat{\gamma}}{\|w\|}$$

$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma} \quad i = 1, \dots m$$

⇒ We've gotten rid of the constraint $\|w\| = 1$. The downside is that we have a nasty (again non-convex) objective $\frac{\hat{\gamma}}{\|w\|_2}$ function.

⇒ Recall, we can add an arbitrary scaling constraint on w & b without changing anything.

↳ functional margin being multiplied by that same constant.

⇒ Maximizing $\frac{\hat{\gamma}}{\|w\|}$ ⇒ minimizing $\|w\|^2$

⇒ We now have the following optimization problem

$$\min_{\gamma, w, b} \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 \quad i = 1, \dots$$

↓

{Optimization Problem with a Convex quadratic objective & Only linear Constraints}

↓

{Can be Solved using Commercial quadratic Programming Code (QP)}

★ Lagrange duality

⇒ Consider a problem of the following form:

$$\min_w f(w)$$
$$\text{s.t } h_i(w) = 0 \quad i = 1, \dots l$$

⇒ Let the Lagrangian be:

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

↓

Lagrange multipliers

⇒ $\frac{\delta \mathcal{L}}{\delta w_i} = 0 \; ; \; \frac{\delta \mathcal{L}}{\delta \beta_i} = 0$ and Solve for w and β.

⇒ On this Section, we will generalize this to Constrained optimization problems. In which we may have Inequality as well as Equality Constraints.

⇒ Consider the following, which we'll call the Primal optimization problem:

$$\min_w f(w)$$
$$\text{s.t. } g_i(w) \leq 0. \quad i = 1, \dots k$$
$$h_i(w) = 0 \quad i = 1, \dots l$$

⇒ To solve it, we start by defining the generalized Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

⇒ Here, the α_i's and β's are the Lagrange multipliers.

$$\theta_p(w) = \max_{\alpha,\beta;\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta)$$

↓

{Stands for Primal}

⇒ If w violates any of the primal constraints then you should be able to verify that

$$\theta_p(w) = \infty$$

⇒ Conversely, if the constraints are indeed satisfied for a particular value of w then $\theta_p(w) = f(w)$

$$\theta_p(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise} \end{cases}$$

⇒ Hence if we consider the minimization problem

$$\min_w \theta_p(w) = \min_w \max_{\alpha,\beta;\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta)$$

We see that it is the same problem as our original, primal problem.

$$\text{let } p^* = \min_w \theta_p(w)$$

⇒ Now, lets look at a slightly different problem.

$$\theta_D(\alpha,\beta) = \min_w \mathcal{L}(w,\alpha,\beta)$$

↓

Dual

$\Rightarrow$ We can now pose the dual optimization Problem:

$$\max_{\alpha,\beta:\alpha_i>0} \theta_D(\alpha,\beta) = \max_{\alpha,\beta,\alpha_i\geq 0} \min_{\omega} \mathcal{L}(\omega,\alpha,\beta)$$

$$d^* = \max_{\alpha,\beta,'\alpha_i\geq 0} \theta_D(\omega)$$

$$d^* \leq p^*$$

$\Rightarrow$ Under certain Condition $d^* = p^*$

- Karush-Kuhn-Tucker (KKT) Condition

$\uparrow$

{ $\omega^*, \alpha^*, \beta^*$ satisfy the KKT Condition }

* Optimal margin Classifiers

⇒ Optimization problem for finding the optimal margin classifier:

{Convex Optimization Problem}

$$\min_{\gamma,w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 \quad i = 1, \dots m$$

Choose $\|w\| = 1/\gamma$ as scaling factor

$$\max_{\gamma,w,b} \gamma$$

$$\text{s.t. } \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|} \geq \gamma \quad i = 1, \dots m$$

{This optimization problem can be solved using commercial quadratic programming code (QP)}

★ Kernels

⇒ We have a problem in which input is x.

⇒ We consider performing regression using the features x, x^2 & x^3 to obtain a cubic function.

⇒ To distinguish between these two set of variables, we'll call the "original" input value the **input attributes** of a problem.

⇒ When that is mapped to some new set of quantities that are passed to the learning algorithm, we'll call those new quantities the **input features**.

⇒ We will also **let ϕ denote the feature mapping**, which maps from the attributes to the features.

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

⇒ Rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$

↳ To do so, we simply need to go over our previous algorithm, & replace x everywhere in it with $\phi(x)$

⇒ Since the algorithm can be written entirely in terms of the inner products $\langle x, z \rangle$, this means that we would replace all those inner product with $\langle \phi(x), \phi(z) \rangle$.

⇒ Specificically, given a feature mapping ϕ, we define the ~~kernel~~ corresponding **Kernel** to be:

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

⇒ Then, everywhere we previously had $\langle x, z \rangle$ in our algorithm, we could simply replace it with $K(x, z)$, and our algorithm would now be learning using the features ϕ.

⇒ Lets see an example. Suppose $x, z \in R^n$ and Consider

$$K(x, z) = (x^T z)^2$$

$$K(x, z) = \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_i z_i \right)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j$$

$$= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)$$

⇒ Thus, we see that $K(x, z) = \phi(x)^T \phi(z)$, where the feature mapping ϕ is given by: (for $n = 3$)

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

⇒ Note that whereas calculating the high-dimensional $\phi(x)$ requires $O(n^2)$ time, finding $K(x, z)$ takes only $O(n)$ time.

⇒ For a related Kernel, also consider

$$K(x, z) = (x^T z + c)^2$$

⇒ This corresponds to feature mapping:

{for $n = 3$}

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ \sqrt{2c}\, x_3 \\ c \end{bmatrix}$$

⇒ More broadly the Kernel $K(x, z) = (x^T z + c)^d$ corresponds to a feature mapping to an $\binom{n+d}{d}$ feature space.

↳ Corresponding of all monomials of the form $x_{i_1} x_{i_2} \cdots x_{i_k}$ and are up to order d.

⇒ Intuitively:

- If $\phi(x)$ & $\phi(z)$ are close together, then we might expect $K(x,z) = \phi(x)^T\phi(z)$ to be large.
- If $\phi(x)$ & $\phi(z)$ are far apart (say nearly orthogonal to eachother) then $K(x,z) = \phi(x)^T\phi(z)$ will be small.

⇒ So, we can think of $K(x,z)$ as some measure of how similar are $\phi(x)$ and $\phi(z)$ or of how similar are x & z.

⇒ Given some function K, how can we tell if it's a valid Kernel?

⇒ Suppose for now that K is indeed a valid Kernel corresponding to some feature mapping ϕ.

- Now consider a finite set of m points
$$\{x^{(1)}, x^{(2)} \dots x^{(m)}\}$$
- Let a square, $m \times m$ matrix K be defined so that (i,j) entry is given by $K_{ij} = K(x^{(i)}, x^{(i)})$
- This matrix is called **Kernel Matrix**.

⇒ Now, if K is a valid Kernel, then $K_{ij} = K_{ji}$ and hence K must be Symmetric.

⇒ Let $\phi_k(x)$ denote the k^{th} coordinate of the vector $\phi(x)$, we find that for any vector z we have

$$z^T K z = \sum_i \sum_j z_i K_{ij} z_j$$

$$= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(i)}) z_j$$

$$= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(i)}) z_j$$

$$= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j$$

$$= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \geq 0$$

⇒ Since z was arbitrary, this shows that K is Positive Semi-definite.

⇒ If K is a valid Kernel, then the corresponding Kernel matrix $K \in \mathbb{R}^{m \times m}$ is Symmetric positive Semidefinite.

↳ This turns out to be not only a necessary, but also a sufficient condition for K to be a valid Kernel. (also called a Mercer Kernel)

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

{Gaussian Kernel}

Theorem (Mercer): Let $K: R^n \times R^n \to R$ be given. Then for K to be a valid (Mercer) Kernel, it is necessary and sufficient that for any $\{x^{(1)} \cdots x^{(m)}\}$, $(m < \infty)$, the corresponding Kernel matrix is symmetric positive semi-definite.

★ Regularization and the non-separable case

⇒ The derivation of the SVM as presented so far assumed that the data is linearly separable.

⇒ While mapping data to a high dimensional feature space via ϕ does generally increase the likelyhood that the data is separable, but we can't guarantee that it always will be so.

⇒ It is not clear that finding a separating hyperplane is exactly what we'd want to do, since that might be susceptible to outliers.

⇒ To make the algorithm work for non-linearly separable datasets as well as be less sensitive to outliers, we reformulate our optimization (using l_1 regularization) as follows:

$$\min_{\gamma, w, b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \quad i = 1, \cdots m$$

$$\xi_i \geq 0 \quad i = 1, \cdots m$$