CS391L Machine Learning Assignment 3

Srinath Tankasala,
st34546

QUESTION 1

o = orange, a = apple, l = lime; g = green, r = red, b = blue;

$$P(a) = P(a|r) * P(r) + P(a|g) * P(g) + P(a|b) * P(b)$$
$$P(a) = \frac{3}{10} * 0.2 + \frac{3}{10} * 0.6 + \frac{1}{2} * 0.2$$
$$P(a) = 0.34 = P(apple)$$

$$P(g|o) = \frac{P(o|g) * P(g)}{P(o)}$$
$$P(g|o) = \frac{0.3 * 0.6}{P(o|g) * P(g) + P(o|b) * P(b) + P(o|r) * P(r)}$$
$$P(g|o) = \frac{0.18}{0.3 * 0.6 + 0.5 * 0.2 + 0.4 * 0.2}$$
$$P(g|o) = 0.5 = P(green\ box|\ fruit = orange)$$

QUESTION 2

Beta distribution PDF:

$$p(\mu, a, b) = \frac{\mu^{a-1}(1-\mu)^{b-1}}{\int_0^1 x^{a-1}(1-x)^{b-1}dx} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$$

Therefore,

$$E(\mu) = \int_0^1 \mu * p \cdot d\mu$$
$$\Rightarrow E(\mu) = \int_0^1 \mu^a(1-\mu)^{b-1}d\mu * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$
$$= \frac{\Gamma(a+b+1)}{\Gamma(a+1)\Gamma(b)} * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$
$$E(\mu) = \frac{a}{a+b}$$

Now to find $E(\mu^2)$,

$$\begin{aligned}
E(\mu^2) &= \int_0^1 \mu^2 * p \cdot d\mu \\
&= \int_0^1 \mu^{a+1}(1-\mu)^{b-1} d\mu * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \\
&= \frac{\Gamma(a+2)\Gamma(b)}{\Gamma(a+b+2)} * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \\
E(\mu^2) &= \frac{(a+1)\cdot a}{(a+b+1)\cdot(a+b)}
\end{aligned}$$

Thus we can find $Var(\mu)$ as

$$\begin{aligned}
Var(\mu) &= E(\mu^2) - (E(\mu))^2 \\
&= \frac{(a+1)\cdot a}{(a+b+1)\cdot(a+b)} - \frac{a^2}{(a+b)^2} \\
&= \frac{(a+1)\cdot a\cdot(a+b) - a^2(a+b+1)}{(a+b+1)\cdot(a+b)^2} \\
Var(\mu) &= \frac{a\cdot b}{(a+b+1)\cdot(a+b)^2}
\end{aligned}$$

QUESTION 3

Given, $E(\nu) = 0$ and $E((\nu-0)^2) = 1$. Also ν is a Gaussian independent random variable, i.e. independent of $z_i \forall i$. Hence,

$$E(z_i\nu) = E(z_i) * E(\nu) = 0 \ \forall \ i \ components \tag{1}$$

$$\begin{aligned}
z_i' &= \mu_i + \alpha(z_i - \mu_i) + \sigma_i\sqrt{1-\alpha^2}\nu \\
E(z_i') &= E(\mu_i) + E(\alpha(z_i-\mu_i)) + E(\sigma_i\sqrt{1-\alpha^2}\nu) \\
&= \mu_i + \alpha(E(z_i) - E(\mu_i)) + \sigma_i\sqrt{1-\alpha^2}E(\nu) \\
E(z_i') &= \mu_i + 0 + 0 = \mu_i
\end{aligned}$$

Now, $z_i' - E(z_i') = z_i' - \mu_i$. Also, by definition,

$$\begin{aligned}
z_i' - \mu_i &= \alpha(z_i-\mu_i) + \sigma_i\sqrt{1-\alpha^2}\nu \\
\Rightarrow (z_i'-\mu_i)^2 &= \alpha^2(z_i-\mu_i)^2 + \sigma_i^2(1-\alpha^2)\nu^2 + 2\sigma_i\sqrt{1-\alpha^2}\nu\alpha(z_i-\mu_i) \\
E((z_i'-\mu_i)^2) &= E(\alpha^2(z_i-\mu_i)^2) + E(\sigma_i^2(1-\alpha^2)\nu^2) + E(2\sigma_i\sqrt{1-\alpha^2}\nu\alpha(z_i-\mu_i)) \\
Var(z_i') &= \alpha^2 E((z_i-\mu_i)^2) + \sigma_i^2(1-\alpha^2)E(\nu^2) + 2\sigma_i\sqrt{1-\alpha^2}\alpha E(\nu(z_i-\mu_i)) \\
&= \alpha^2\sigma_i^2 + \sigma_i^2(1-\alpha^2) + 2\sigma_i\sqrt{1-\alpha^2}\alpha(E(\nu\cdot z_i) - E(\nu\mu_i)); \text{ Using 1,} \\
\Rightarrow Var(z_i') &= \sigma_i^2
\end{aligned}$$

QUESTION 4

A. Kernel trick

To understand the Kernel trick let us take the example of the LMS regression problem with Kernels. A kernel function maps the given data features into a higher dimensional space. For instance, in the MNIST problem, each image $(x^{(i)})$ had d = 784 components. A kernel function $\phi(x^{(i)})$ would map those d features

into a hyperspace of more dimensions. Thus the regression problem for recognition of one digit becomes:

$$\min_{\theta} f = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T\phi(x^{(i)}))^2;\ \text{label } y^{(i)} \text{ is +1 if it's the digit, -1 otherwise} \tag{2}$$

N = number of data points (60000 for MNIST). $x^{(i)}$ has dimension dx1 and θ has the dimension of the hyperspace $\phi(x^{(i)})$. Using gradient descent,

$$\frac{\partial f}{\partial \theta} = \sum_{i=1}^{N}(y^{(i)} - \theta^T\phi(x^{(i)}))\phi(x^{(i)}) \tag{3}$$

for classic regression, $\phi(x^{(i)}) = x^{(i)}$, but take for instance a cubic polynomial feature set, i.e.

$$\big[\phi(x)\big] = \begin{bmatrix} x_1 \\ ... \\ x_n \\ x_1^2 \\ x_1x_2 \\ ... \\ x_d^2 \\ x_1^3 \\ x_1^2x_2 \\ ... \\ x_d^3 \end{bmatrix} = O(d^3)$$

Thus the hyperspace has more than d^3 dimensions. Thus our θ vector has d^3 = 784^3 components which is computationally expensive to store and manipulate for each iteration of the gradient descent. So instead we use the "kernel trick" where instead of solving for θ we introduce a new vector β that is easier to handle. Let's rewrite θ as:

$$\theta = \sum_{j=1}^{N}\beta_j\phi(x^{(j)}) \tag{4}$$

Proving β exists for every θ is done using induction. Let at some iteration 'k', a β exists, hence:

$$\begin{aligned}
\theta[k] &= \sum_{j=1}^{N}\beta_j\phi(x^{(j)}), \\
\theta[k+1] &= \theta[k] + \eta\cdot\frac{\partial f}{\partial\theta} \\
&= \theta[k] + \eta\cdot\sum_{j=1}^{N}(y^{(j)} - \theta^T\phi(x^{(j)}))\phi(x^{(j)});\ \eta = \textit{learning rate} \\
&= \sum_{j=1}^{N}\beta_j\phi(x^{(j)}) + \eta\cdot\sum_{j=1}^{N}(y^{(j)} - \theta^T\phi(x^{(j)}))\phi(x^{(j)}) \\
&= \sum_{j=1}^{N}(\beta_j + \eta\cdot(y^{(j)} - \theta^T\phi(x^{(j)}))\phi(x^{(j)})
\end{aligned}$$

rewriting θ^T,

$$\theta[k+1] = \sum_{j=1}^{N}(\beta_j + \eta \cdot (y^{(j)} - \sum_{i=1}^{N} \beta_i \phi^T(x^{(i)})\phi(x^{(j)}))\phi(x^{(j)}) \tag{5}$$

$$\theta[k+1] = \sum_{j=1}^{N} \beta'_j \phi(x^{(j)}) \tag{6}$$

Thus if β exists at the k^{th} step then it exists for future steps too, and the trivial case $\theta = 0, \beta = 0$ vectors is an obvious solution. So the problem of solving for θ has been transformed into solving for β. From equation 5 the iterative algorithm to solve for β is:

$$\beta'_j := \beta_j + \eta \cdot (y^{(j)} - \sum_{i=1}^{N} \beta_i \phi^T(x^{(i)})\phi(x^{(j)}) \ \forall \ j = 1, ..., N \tag{7}$$

$$\beta'_j := \beta_j + \eta \cdot (y^{(j)} - \sum_{i=1}^{N} \beta_i K(i,j)) \tag{8}$$

Where, K is called the kernel matrix of size NxN where $K(i,j) =< \phi(x^{(i)}), \phi(x^{(j)}) >$. K can be precomputed for all N data points before the loop for β is started, thus accelerating the computation. A prediction for a new data point x is given by,

$$y = \sum_{i=1}^{N} \beta_i \phi^T(x^{(i)})\phi(x)$$

Note that β is a vector of N components = 60000 for the MNIST problem $<< 784^3$. The only computational cost comes from storing the kernel matrix K = 60000x60000 elements. However, since we are working with a better kernel function, our model will need far lesser data to perform the classification. So the kernel trick is advantageous in problems where the Hyperspace dimensionality is much bigger than the amount of training data.

B. Maximum margin proof

Recall that the goal of our classifier was to maximize the geometric margin ρ, given N data points $x^{(1)}, x^{(2)}, ...x^{(N)}$ with labels $y^{(i)}$,

$$\max_{w,b} \rho = \left(\min_i \frac{y^{(i)}(w^T\phi(x^{(i)}) + b)}{||w||} \right), \ i \ \epsilon \ [1, 2, .., N] \tag{9}$$

Without loss of generality, w can be scaled as $w \rightarrow \kappa w$ and $b \rightarrow \kappa b$ so that

$$\min_i y^{(i)}(w^T\phi(x^{(i)}) + b) = 1$$

Thus the problem can be transformed as:

$$\max_{w,b} \rho = \frac{1}{||w||} \Rightarrow \min_{w,b} \frac{1}{2}||w||^2 \tag{10}$$

subject to, $y^{(i)}(w^T\phi(x^{(i)}) + b) \geq 1 \ \forall \ i$. This is converted into a Lagrangian problem to include the constraints as,

$$\min_{w,b,\lambda} L(w, b, \lambda) = \left(\frac{1}{2}||w||^2 - \sum_{i=1}^{N} \lambda_i (y^{(i)}(w^T\phi(x^{(i)}) + b) - 1) \right) \tag{11}$$

Now taking partial derivatives gives,

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y^{(i)}(\phi(x^{(i)})) = 0 \tag{12}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \tag{13}$$

Additionally, it can be shown that for a constrained optimization in the form above, it must satisfy the Karush-Kuhn-Tucker (KKT) conditions (refer Bishop, Appendix E), which in this case requires the following two properties to hold:

$$\lambda_i > 0 \tag{14}$$

$$\lambda_i(y^{(i)}(w^T\phi(x^{(i)}) + b) - 1) = 0 \ \forall \ i \tag{15}$$

Thus summing equation (15) over all i, we get:

$$\sum_{i=1}^{N} \lambda_i(y^{(i)}(w^T\phi(x^{(i)}) + b) - 1) = 0 \tag{16}$$

$$\Rightarrow \sum_{i=1}^{N} \lambda_i y^{(i)}(w^T\phi(x^{(i)})) + b\sum_{i=1}^{N} \lambda_i y^{(i)} - \sum_{i=1}^{N} \lambda_i = 0 \tag{17}$$

$$w^T \sum_{i=1}^{N} (\lambda_i y^{(i)}\phi(x^{(i)})) + b\sum_{i=1}^{N} \lambda_i y^{(i)} - \sum_{i=1}^{N} \lambda_i = 0 \tag{18}$$

Using equations (12) and (13), we can simplify equation (18) as:

$$w^T w + 0 = \sum_{i=1}^{N} \lambda_i$$

$$||w||^2 = \sum_{i=1}^{N} \lambda_i$$

substituting equation (10),

$$\frac{1}{\rho^2} = \sum_{i=1}^{N} \lambda_i$$