CS229 Fall 2017, Problem Set #2: Supervised Learning II

Armand Sumo – armandsumo@gmail.com

May 24, 2021

Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

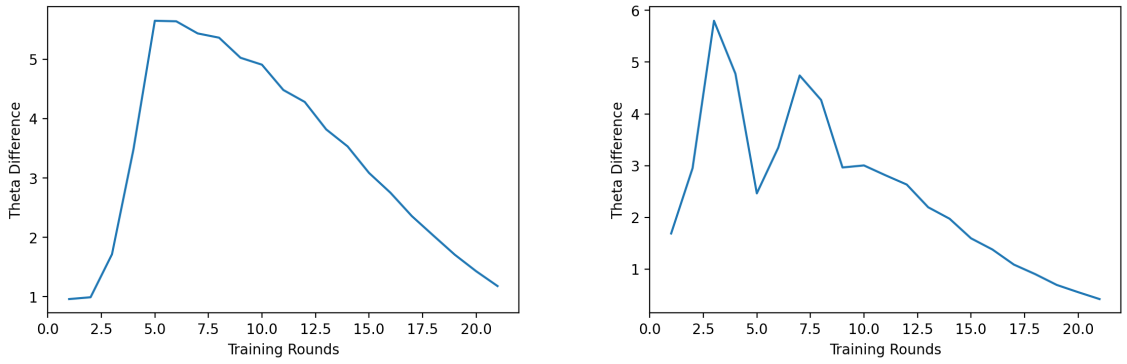
(The code used to generate the graphs can be found in the file *assignment-2.py*.)

1. Logistic regression: Training stability

(a) Training on data set A us much more stable than training on data set B. Moreover the training algorithm on data set B doesn't converge. Figure 1 shows how early in the training, the gradient updates are more stable on data set A than they are on data set B. After $n = 1 \times 10^6$ gradient update iterations, the difference between the norm of the parameters vector θ between n-th and $n-1$-th update is:

$$\theta_{diff} = 0.0 \qquad \text{(dataset A)}$$
$$\theta_{diff} = 7.274861440466199 \times 10^{-4} \qquad \text{(dataset B)}$$

On dataset B, θ_{diff} is far above the convergence threshold 1×10^{-15}.



(a) data set A (b) data set B

Figure 1: Training stability for the first 20 gradient updates.

(b) The situation we're facing is called *separation* [2], which refers to when an explanatory variable perfectly predicts some binary observations. Separation is said to be *complete* when the mode correctly classifies both all positive and negative examples, and *quasicomplete* when it only classifies either all positive or all negative examples but not both. *Overlap* refers the case in which he model achieves neither quasicomplete nor complete separation.
The likelihood $L(\theta)$ can be formulated as:

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) = \prod_{i=1}^{m} g(\theta^T x^{(i)})^{1|\{y^{(i)}=1\}}(1 - g(\theta^T x^{(i)}))^{1-1|\{y^{(i)}=1\}}$$

Under complete separation, to maximize the likelihood of the data, the logistic regression curve must assign $g_\theta(x^{(i)}) = \dfrac{1}{1+e^{-\theta^T x^{(i)}}} = 1$ when $y^{(i)} = 1$ and $g_\theta(x^{(i)}) = \dfrac{1}{1+e^{-\theta^T x^{(i)}}} = 0$ when $y^{(i)} = 0$. Because the logistic curve lies strictly between 0 and 1, this likelihood cannot be achieved, it can only be approached asymptotically as the coefficients of θ approach infinity. The likelihood function under complete separation is therefore monotonic which implies that a finite maximum likelihood estimate cannot be found.
Now,suppose the model achieves quasicomplete separation and correctly predicts all positive examples. To maximize the likelihood of the data, the logistic regression curve must assign $g_\theta(x^{(i)}) = \dfrac{1}{1+e^{-\theta^T x^{(i)}}} = 1$ when $y^{(i)} = 1$ and $g_\theta(x^{(i)}) = \dfrac{1}{1+e^{-\theta^T x^{(i)}}} \in \{0, 1\}$ when $y^{(i)} = 0$. Because the logistic curve lies strictly between 0 and 1, this likelihood cannot be achieved, for the same reasons as the complete separation case, a finite maximum likelihood estimate cannot be found.
By testing the correspondence between predicted and ground truth labels, I found that the model achieves complete separation on dataset B but neither complete nor quasicomplete on dataset A, which explains the training instability observed solely on dataset B.

(c)
i. **Using a different constant learning rate** will only shift the problem to higher or lower values of gradient updates and would not stabilize the training. A significantly lower learning rate will cause slow convergence and a significantly higher learning rate may worsen training stability.

ii. **Decreasing the learning rate over time** can improve the training stability, by pushing the gradient updates to increasingly low values as the training progresses, thus ending up below the convergence threshold.

iii. **Adding a regularization term $\|\theta\|_2^2$ to the loss function** corresponds to introducing a gaussian prior for our parameter θ that shrinks the estimates towards zero. This causes our hypothesis h_θ to return values closer to 0.5 which reduces correct predictions and therefore pushes the model away from complete and quasicomplete separation.

iv. **Linear scaling on the input features** changes the speed of the updates but doesn't reduce instability.

v. **Adding a zero mean Gaussian noise to the training data or labels** can be seen as modeling our target Bishop(1995) has shown that adding noise to the input is equivalent to Tikhonov regularization. It can be formulated in the following way: we wish to find the parameter θ that minimizes a function of the form:

$$\|f_\theta(x) - y\|_2^2 + \|\Gamma\theta\|_2^2$$

Where Γ is a Tikhonov matrix. In the case of L_2 regularization, Γ is the identity matrix. This would, as in (iii.) skew the output of our hypothesis function h_θ towards value closer to 0.5, thus reducing correct predictions and reducing the occurrence of our model achieving complete and quasicomplete separation.

(d) Support vector machines(SVMs), which use hinge loss, will map data points on a higher-dimensional space where they are linearly separable. SVMs underperform when the data set contains overlapping(or wrongly assigned)labels. If we had overlapping labels, the logistic regression model would not have achieved complete separation on dataset B. Because it did, this potential cause of training instability of SVMs would not be relevant to dataset B.

2. Model calibration

(a) We have a training set $\{x^{(i)}, y^{(i)}\}_{i=1}^m$ with $x^{(i)} \in \mathbb{R}^{n+1}$ and $y^{(i)} \in \{0, 1\}$. We also have an intercept term $x_0^{(i)} = 1$ for all i and $\theta \in \mathbb{R}^{n+1}$ is our parameter vector. Additionally, g denotes the sigmoid function.
To find the maximum likelihood estimator of θ we start by writing down its likelihood $L(\theta)$:

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) = \prod_{i=1}^{m} g(\theta^T x^{(i)})^{1|\{y^{(i)}=1\}}(1 - g(\theta^T x^{(i)}))^{1-1|\{y^{(i)}=1\}}$$

Because the log function is monotone increasing, maximizing the likelihood of θ is equivalent to maximizing its log-likelihood.

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} 1|\{y^{(i)} = 1\} \log\big(g(\theta^T x^{(i)})\big) + (1 - 1|\{y^{(i)} = 1\}) \log\big(1 - g(\theta^T x^{(i)})\big)\end{aligned}$$

for each $j \in 1, \ldots, m$

$$\frac{\partial \ell(\theta)}{\partial \theta_j} = \left[1|\{y^{(i)} = 1\} - g(\theta^T x^{(i)}))\right] x_j^{(i)}$$

Particularly, for $j = 0$, we include a bias term $x_0^{(i)} = 1$. A necessary condition for $\hat{\theta}$ to be the maximum likelihood estimator of θ is therefore:

$$\frac{\partial \ell}{\partial \theta_0}(\hat{\theta}) = 0 = 1|\{y^{(i)} = 1\} - g((\hat{\theta})^T x^{(i)}) = \frac{1}{1 + \exp\left(-(\hat{\theta})^T x^{(i)}\right)}$$

Because for each $i \in \{1, \ldots, m\}$ we have $h_{\theta(x^{(i)})} \in (0, 1)$, the interval $I_{a,b}$ is $\{1, \ldots, m\}$.

$$\begin{aligned}\sum_{i \in I_{a,b}} \frac{p(y^{(i)} = 1|x^{(i)}, \hat{\theta})}{|\{i \in I_{a,b}\}|} &= \frac{1}{m} \sum_{i=1}^{m} p(y^{(i)} = 1|x^{(i)}, \hat{\theta}) = \frac{1}{m} \sum_{i=1}^{m} h_{\hat{\theta}}(x^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^{m} \frac{1}{1 + \exp\left(-\hat{\theta}^T x^{(i)}\right)} = \sum_{i=1}^{m} 1|\{y^{(i)} = 1\}\end{aligned}$$

Therefore for the described logistic regression model, the fraction of the positives in the set of examples for which $h_{\hat{\theta}}(x^{(i)})$ falls in the range $(a, b) = (0, 1)$ is equal to the average of the model outputs for these examples:

$$\boxed{\sum_{i \in I_{a,b}} \frac{p(y^{(i)} = 1|x^{(i)}, \hat{\theta})}{|\{i \in I_{a,b}\}|} = \sum_{i \in I_{a,b}} \frac{1|\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|}}$$

(b) Suppose the model is perfectly calibrated, i.e, the above property is true for any $(a, b) \subset (0, 1)$, then it must hold for the following case where $(a, b) = (0.4, 0.6)$ and:

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \begin{pmatrix} h_{\hat{\theta}}(x^{(1)}) \\ h_{\hat{\theta}}(x^{(2)}) \end{pmatrix} = \begin{pmatrix} 0.45 \\ 0.55 \end{pmatrix}; I_{a,b} = \{1, 2\}$$

The property does indeed hold:

$$\frac{h_{\hat{\theta}a}(x^{(1)}) + h_{\hat{\theta}}(x^{(2)})}{2} = 0.5 = \frac{1|\{y^{(1)} = 1\} + 1|\{y^{(2)} = 1\}}{2} = \frac{1 + 0}{2}$$

However, the accuracy achieved by the model is not 1, in this case it's equal to 0:

$$\begin{pmatrix} 1|\{h_{\hat{\theta}}(x^{(1)}) > 0.5\} \\ 1|\{h_{\hat{\theta}}(x^{(2)}) > 0.5\} \end{pmatrix} = \begin{pmatrix} 1|\{0.45 > 0.5\} \\ 1|\{0.55 > 0.5\} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \neq \begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Conversely, let's suppose our model's accuracy is equal to 1.Let's pick a case identical to the one in (b) except for:

$$\begin{pmatrix} h_{\hat{\theta}}(x^{(1)}) \\ h_{\hat{\theta}}(x^{(2)}) \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0.45 \end{pmatrix}$$

This case is in line with the model having perfect accuracy:

$$\begin{pmatrix} 1|\{h_{\hat{\theta}}(x^{(1)}) > 0.5\} \\ 1|\{h_{\hat{\theta}}(x^{(2)}) > 0.5\} \end{pmatrix} = \begin{pmatrix} 1|\{0.6 > 0.5\} \\ 1|\{0.45 > 0.5\} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix}$$

However, the property proved in (a) does not hold:

$$\frac{h_{\hat{\theta}}(x^{(1)}) + h_{\hat{\theta}}(x^{(2)})}{2} = 0.525 \neq \frac{1|\{y^{(1)} = 1\} + 1|\{y^{(2)} = 1\}}{2} = 0.5$$

Therefore, perfect calibration does not imply perfect accuracy and perfect accuracy does not imply perfect calibration.

(c) To estimate the parameters θ we maximize an objective function f^*. We add a regularization term to *penalize* high values of θ so it should be preceded by a negative sign. For convenience we include a factor 0.5 in the L_2 regularization term.

$$f^*(\theta) = \ell(\theta) - \frac{1}{2}\sum_{i=1}^{n} \theta_i^2$$

From here we follow the same procedure as in (a).
For each $j \in 1, \ldots, m$

$$\frac{\partial f^*(\theta)}{\partial \theta_j} = \left[1|\{y^{(i)} = 1\} - g(\theta^T x^{(i)})\right] x_j^{(i)} - \theta_j$$
$$\frac{\partial f^*(\theta)}{\partial \theta_0} = \left[1|\{y^{(i)} = 1\} - g(\theta^T x^{(i)})\right] - \theta_0$$

Particularly, for $j = 0$, we include a bias term $x_0^{(i)} = 1$.A necessary condition for $\hat{\theta}$ to be the maximum likelihood estimator of θ is therefore:

$$\frac{\partial f^*}{\partial \theta_0}(\hat{\theta}) = 0 = 1|\{y^{(i)} = 1\} - g(\hat{\theta}^T x^{(i)}) - \theta_0 = \frac{1}{1 + \exp\left(-\hat{\theta}^T x^{(i)}\right)}$$

The effect on the model calibration is the following:

$$\sum_{i \in I_{a,b}} \frac{p(y^{(i)} = 1|x^{(i)}, \hat{\theta})}{|\{i \in I_{a,b}\}|} = \sum_{i \in I_{a,b}} \frac{1|\{y^{(i)} = 1\} - \theta_0}{|\{i \in I_{a,b}\}|}$$

Because of the addition of the $-\theta_0$ term, the property proved in (a) no longer holds.

3. Bayesian logistic regression and weight decay

We use the same notations and data as in the previous section. The maximum likelihood estimate of the parameters θ is given by:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) \tag{1}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} y^{(i)} \log\big(g(\theta^T x^{(i)})\big) + (1 - y^{(i)}) \log\big(1 - g(\theta^T x^{(i)})\big) \tag{2}$$

$$= \underset{\theta}{\operatorname{argmax}}\, J(x, \theta, y) \tag{3}$$

Where $J(x, \theta, y)$ is our objective function.
Suppose we choose a Bayesian prior $\theta \sim \mathcal{N}(0,\, \tau^2 I)$ where $\tau > 0$ and I is the $n+1$-by-$n+1$ identity matrix, and find the MAP estimate of θ as:

$$\begin{aligned}
\theta_{MAP} &= \underset{\theta}{\operatorname{argmax}}\, p(\theta) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) \\
&= \underset{\theta}{\operatorname{argmax}} \frac{1}{(2\pi)^{(\frac{n+1}{2})}} \exp\left(-\frac{\theta^T\theta}{2\tau^2}\right) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) \\
&= \underset{\theta}{\operatorname{argmax}} -\frac{\theta^T\theta}{2\tau^2} + \sum_{i=1}^{m} y^{(i)} \log\big(g(\theta^T x^{(i)})\big) + (1 - y^{(i)}) \log\big(1 - g(\theta^T x^{(i)})\big) \\
&= \underset{\theta}{\operatorname{argmax}}\, \tilde{J}(x, \theta, y)
\end{aligned}$$

where

$$\tilde{J}(x, \theta, y) = J(x, \theta, y) - \frac{\theta^T\theta}{2\tau^2}$$

$\tilde{J}(x, \theta, y)$ is our new, *regularized* objective function. We now write down the gradient ascent update rule with $\alpha > 0$ and $\alpha < \tau$:

$$\begin{aligned}
\theta &:= \theta + \alpha(\nabla_\theta \tilde{J}(x, \theta, y)) \\
&:= \theta + \alpha(\nabla_\theta J(x, \theta, y) - \frac{\theta}{\tau^2}) \\
&:= \theta(1 - \frac{\alpha}{\tau^2}) + \alpha \nabla_\theta J(x, \theta, y)
\end{aligned}$$

This shows that at each step, the addition of the term $\frac{\theta^T\theta}{2\tau^2}$ scales down the parameter vector θ by a constant factor.
To understand what happens over the entire course of training, we will start by making a

quadratic approximation of the objective function in the neighborhood of the values of the parameters θ that obtain maximum likelihood [1, Chapter 7]:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, J(x, \theta, y)$$

The approximation $\hat{J}$ is given by:

$$\hat{J}(\theta) = J(\theta_{ML}) + \frac{1}{2}(\theta - \theta_{ML})^T H(\theta - \theta_{ML}) \tag{4}$$

Where $H \in \mathbb{R}^{(n+1)\times(n+1)}$ is the Hessian matrix of J with respect to θ evaluated at θ_{ML}. Because θ_{ML} is a maximum, $[\nabla_\theta J](\theta_{ML}) = 0$. This explains the absence of a first order term in (4). Moreover, because θ_{ML} is a location of a maximum of J, H is negative semidefinite. The maximum of $\hat{J}$ occurs when its gradient is equal to 0:

$$\nabla_\theta \hat{J}(\theta) = H(\theta - \theta_{ML}) \tag{5}$$

To study the effect of the regularization term, we add the regularization term to the right hand side of (5). We now solve for the regularized version of $\hat{J}$ and define θ_{MAP} to be the location of the maximum.

$$\begin{aligned} -\frac{\theta_{MAP}}{\tau^2} + H(\theta_{MAP} - \theta_{ML}) &= 0 \\ (H - \frac{I}{\tau^2})\theta_{MAP} &= H\theta_{ML} \\ \theta_{MAP} &= (H - \frac{I}{\tau^2})^{-1} H\theta_{ML} \end{aligned} \tag{6}$$

As τ grows, the regularized solution θ_{MAP} approaches θ_{ML}.
Let's study what happens as τ approaches 0, which means that the diagonal terms of the covariance matrix $\tau^2 I$ of the random vector θ get smaller.
Because H is real and symmetric, we can decompose it into a diagonal matrix Λ and an orthonormal basis of eigenvectors Q such that $H = Q\Lambda Q^T$ Applying the decomposition to (6), we obtain:

$$\begin{aligned} \theta_{MAP} &= (Q\Lambda Q^T - \frac{I}{\tau^2})^{-1} Q\Lambda Q^T \theta_{ML} \\ &= \left[Q(\Lambda - \frac{I}{\tau^2})Q^T\right]^{-1} Q\Lambda Q^T \theta_{ML} \\ &= Q(\Lambda - \frac{I}{\tau^2})^{-1} \Lambda Q^T \theta_{ML} \\ &= Q \begin{bmatrix} \frac{\lambda_1}{\lambda_1 - \frac{1}{\tau^2}} & & \\ & \ddots & \\ & & \frac{\lambda_{n+1}}{\lambda_{n+1} - \frac{1}{\tau^2}} \end{bmatrix} Q^T \theta_{ML} \end{aligned}$$

Where $(\lambda_1, \cdots, \lambda_{n+1})$ are the (negative) eigenvalues of H. The effect of the introduction of a Gaussian prior to our parameters θ is to rescale θ_{ML} along the axes defined by the eigenvectors of H. Particularly, the component of θ_{ML} that is aligned with the i-th eigenvector of H is rescaled by a factor

$$0 \leq \frac{\lambda_i}{\lambda_i - \frac{1}{\tau^2}} = \frac{\lambda_i}{-(-\lambda_i + \frac{1}{\tau^2})} \leq 1$$

Along the directions where the eigenvalues of H are very large , per example $|\lambda_i| \gg \frac{1}{\tau^2}$, the scaling coefficients are close to 1, which means regularization has little to no impact.
Along the directions where the eigenvalues of H are large, the components of θ_{ML} will be shrunk to zero. This effect is even more pronounced if we choose τ very large, i.e. if we choose a covariance matrix with a large norm for the Gaussian prior of θ.
Because the scaling factor is always lower than 1, we can conclude the following inequality:

$$\boxed{\|\theta_{MAP}\|_2 \leq \|\theta_{ML}\|_2}$$

(The next result is out of the scope of this exercise.)
Because Q is orthonormal and $(\Lambda - \dfrac{I}{\tau^2})$ is diagonal, we can derive the inequality:

$$\|\theta_{MAP}\|_2 \leq \sqrt{\sum_{i=1}^{n+1} \left(\frac{\lambda_i}{\lambda_i - \frac{1}{\tau^2}} \right)^2} \|\theta_{ML}\|_2$$

4. Constructing kernels

Let K_1,K_2 be kernels over $\mathbb{R}^n \times \mathbb{R}^n$, let $a \in \mathbb{R}^+$ be a positive real number, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a real-valued function, let $\phi : \mathbb{R}^n \mapsto \mathbb{R}^d$ be a function mapping from $\mathbb{R}^n$ to $\mathbb{R}^d$, let K_3 be a kernel over $\mathbb{R}^d \times \mathbb{R}^d$ and let $p(x)$ a polynomial over x with positive coefficients.

(a) Let v be a nonzero vector in $\mathbb{R}^m$.

$$K(x,z) = K_1(x,z) + K_2(x,z)v^T K v \qquad = v^T(K_1 + K_2)v = \underbrace{v^T K_1 v}_{\geq 0} + \underbrace{v^T K_2 v}_{\geq 0} \geq 0$$

K is positive semidefinite. By Mercer's theorem, $K(x,z)$ is a kernel.

(b)

$$K(x,z) = K_1(x,z) + K_2(x,z)$$

$K(x,z)$ is not necessarily a Kernel. Suppose $K_1(x,z)$ and $K_2(x,z)$ are kernels such that $v^T K_1 v = a$; $v^T K_2 v = b$ and $a < b$ then $v^T K v = a - b < 0$.

(c)

$$K(x,z) = aK_1(x,z)$$

where $a \in \mathbb{R}^+$.
$v^T K v = v^T a K_1 v = \underbrace{a}_{\geq 0} \underbrace{v^T K_1 v}_{\geq 0} \geq 0.$
K is positive semidefinite. By Mercer's theorem, $K(x,z)$ is a kernel.

(d)

$$K(x,z) = -aK_1(x,z)$$

where $a \in \mathbb{R}^+$. K is not a kernel since $v^T K v = v^T(-a)K_1 v = -\underbrace{a}_{\geq 0} \underbrace{v^T K_1 v}_{\geq 0} \leq 0.$

(e)

$$K(x,z) = K_1(x,z)K_2(x,z)$$

Because K_1 and K_2 are positive semidefinite(PSD), their product is also positive semidefinite as long as $K_1 K_2$ is symmetric.

$$\begin{aligned} K_1K_2(x^{(i)}, x^{(j)}) &= \sum_{i=1}^{m}\sum_{j=1}^{m} K_1(x^{(i)}, x^{(j)})K_2(x^{(j)}, x^{(i)}) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} K_1(x^{(j)}, x^{(i)})K_2(x^{(i)}, x^{(j)}) \\ &= K_1K_2(x^{(j)}, x^{(i)}) \end{aligned}$$

Hence $K_1 K_2$ is symmetric thus K is positive semidefinite and by Mercer's theorem, K is a kernel.

(f)

$$K(x,z) = f(x)f(z)$$

$K(x,z)$ Is not necessarily a Kernel, per example if we define $f(x) = -x^2$ and $f(z) = z^2$ then $f(x)f(z) = -(xz)^2$ and $v^T K v = -(xz)^2 \underbrace{v^T v}_{\geq 0} \leq 0$

(g)

$$K(x,z) = K_3(\phi(x), \phi(z))$$

$$v^T K v = v^T K_3^2 v = v^T \underbrace{K_3 K_3}_{\text{PSD}} v \geq 0$$

K is positive semidefinite. By Mercer's theorem, K is a Kernel.

(h)

$$K(x,z) = p(K_1(x,z))$$

$$\begin{aligned} v^T K v &= v^T \left(\sum_{i=0}^{\infty} a_i K_1^i \right) v \\ &= \sum_{i=0}^{\infty} \underbrace{a_i}_{\geq 0} v^T \underbrace{K_1^i}_{PSD} v \geq 0 \end{aligned}$$

By Mercer's theorem, K is a kernel.

5. Kernelizing the perceptron

Let there be a binary classification problem with $y \in \{-1, 1\}$. The perceptron uses hypotheses of the form $h_\theta(x) = g(\theta^T x)$ where $g(z) = sign(z)$. The update rule for this version of the perceptron algorithm is:

$$\theta^{(i+1)} := \theta^{(i)} + \alpha \ \mathbf{1}\{g(\theta^{(i)T} x^{(i+1)}) \ y^{(i+1)} < 0\} y^{(i+1)} x^{(i+1)}$$

Where $\theta^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Prior to seeing any example, $\theta^{(0)} = \vec{0}$. Let K be a Mercer Kernel corresponding to some very high-dimensional(suppose ∞ -dimensional) feature mapping ϕ such that it's impossible to compute $\phi(x)$ explicitly. We apply the "kernel trick" to the perceptron to make it work in the high dimensional feature space ϕ but without ever explicitly computing $\phi(x)$.
First we replace all occurrences of $x^{(i)}$ in the algorithm above with $\phi(x^{(i)}$ and obtain the update rule:

$$\theta^{(i+1)} := \theta^{(i)} + \alpha \ \mathbf{1}\{g(\theta^{(i)T} \phi(x^{(i+1)})) \ y^{(i+1)} < 0\} y^{(i+1)} \phi(x^{(i+1)}) \tag{7}$$

We assume $\theta^{(i)}$ can be represented as a linear combination of $\{\phi(x^{(0)}) \cdots \phi(x^{(n)})\}$,
$(n < \infty)$.

$$\theta^{(i)} = \sum_{j=0}^{i} \beta_j \phi(x^{(j)})$$

By setting $\beta_0 = 0$, we initialize $\theta^{(0)}$ as $\vec{0}$. We can now rewrite (1) as

$$\begin{aligned} \theta^{(i+1)} &:= \sum_{j=0}^{i} \beta_j \phi(x^{(j)}) + \alpha \ \mathbf{1}\{g(\theta^{(i)T} \phi(x^{(i+1)})) \ y^{(i+1)} < 0\} y^{(i+1)} \phi(x^{(i+1)}) \\ &:= \sum_{j=0}^{i+1} \beta_j \phi(x^{(j)}) \end{aligned}$$

Where $\beta_{j+1} = \alpha\ \mathbf{1}\{g(\theta^{(i)T}\phi(x^{(i+1)}))y^{(i+1)} < 0\}y^{(i+1)}$.
We then derive the update rule for β:

$$\beta_{i+1} := \alpha\ \mathbf{1}\{g\left(\left(\sum_{j=0}^{i}\beta_j\phi(x^{(j)})\right)^T\phi(x^{(i+1)})\right)\ y^{(i+1)} < 0\}y^{(i+1)}$$

$$:= \alpha\ \mathbf{1}\{g\left(\sum_{j=0}^{i}\beta_j\phi(x^{(j)})^T\phi(x^{(i+1)})\right)\ y^{(i+1)} < 0\}y^{(i+1)}$$

By definition of the (Mercer) Kernel K,

$$K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T\phi(x^{(j)})$$

The update rule for B can therefore be rewritten as

$$\beta_{i+1} := \alpha\ \mathbf{1}\{g\left(\sum_{k=0}^{i}\beta_k K(x^{(k)}, x^{(i+1)})\right)\ y^{(i+1)} < 0\}y^{(i+1)}$$

One of the indexes was changed from j to k to avoid confusion. We can now make the following remarks:

1. If we have n data-points(examples), then K will be a n-by-n matrix.
2. We can precompute the pairwise inner products $\phi(x^{(i)})^T\phi(x^{(j)})$ before the loop starts. For this version of the perceptron algorithm, we don't need to compute such products in case $i = j$.
3. As an example, the parameter $\beta_{(i)}$ is calculated by computing the sum of the $i-1$ first elements of the j-th row of K.
4. Once we've stored all values of β_i we can make a prediction by computing :

$$\begin{aligned}\theta^T x &= \sum_{i=0}^{n}\beta_i\phi(x^{(i)})^T\phi(x)\\ &= \sum_{i=0}^{n}\beta_i K(x^{(i)}, x)\end{aligned}$$

References

[1] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Rainey, C. Dealing with separation in logistic regression models. *Political Analysis 24*, 3 (2016), 339–355.