CS 229, Fall 2018
Problem Set #2 Solutions: Supervised Learning II

Haiyuan Mei (hmei0411@stanford.edu)

1. Logistic Regression: Training stability

a) dataset A converges very quickly, but dataset B doesn't seem to be able to converge ever.

b) The short answer is that Logistic Regression will not converge when data is perfectly separable. I will try to explain this mathematically as following.

- First part of the proof below will show that if the data is perfectly separable, the gradient of loss function w.r.t. θ can never be 0, which means the log-likelihood can never reach to a maximum. The probability of a point is:

$$p\left(x^{(i)}\right) = \frac{1}{1 + e^{y^{(i)}\theta^T x^{(i)}}}$$

For simplicity, assume $x^{(i)}$ only has one feature, θ is just a scaler. A correctly classified point should satisfy $y^{(i)}x^{(i)}\theta < 0\ s.t.\ p\left(x^{(i)}\right) > 0.5$; Otherwise the sample is misclassified.

The MLE is the solution of $\ell'(\theta) = 0$. To further simply the notation let's assume $c^{(i)} = y^{(i)}x^{(i)}$, the log-likelihood loss can be written as:

$$\ell(\theta) = \frac{1}{m}\sum_{i=1}^{m}\log\left(h\left(x^{(i)}\right)\right) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(1 + e^{y^{(i)}x^{(i)}\theta}\right) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(1 + e^{c^{(i)}\theta}\right)$$

Take the derivative w.r.t. θ:

$$\ell'(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\frac{c^{(i)}e^{c^{(i)}\theta}}{1 + e^{c^{(i)}\theta}} = \frac{1}{m}\sum_{i=1}^{m}\frac{-c^{(i)}}{1 + e^{-c^{(i)}\theta}}$$

Since $c^{(i)}\theta < 0$, $1 + e^{-c^{(i)}\theta} > 0$, we have $\ell'(\theta) \neq 0$ for all θ. This explains why there is no convergence. The plot of $\ell'(\theta)$ should look something like the following. Note that when $\theta < 0$, we should have $c^{(i)} > 0$ to make $c^{(i)}\theta < 0$, which is the lower part of Figure 1; when $\theta > 0$, we should have $c^{(i)} < 0$ to make $c^{(i)}\theta < 0$, which is the upper part of Figure 1.

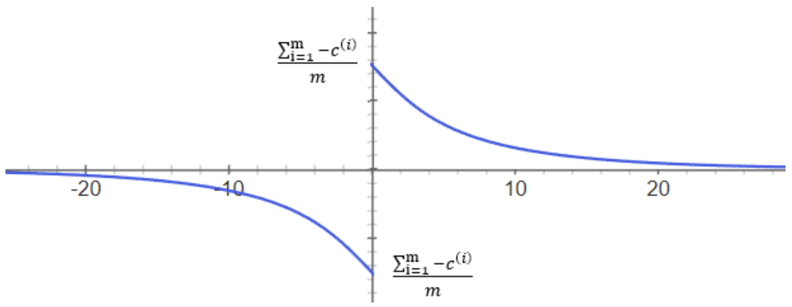


Figure 1, Gradient w.r.t. θ

- Now consider there is a misclassified point. Just one misclassified point will make $\ell'(\theta) = 0$ solvable, hence the algorithm can converge. A misclassified point means in the assumption made above, $c^{(j)}\theta > 0$, this will always make $p\left(x^{(i)}\right) = \frac{1}{1+e^{c^{(i)}\theta}} < 0.5$. Now we need to show that in this case $\ell'(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}} = 0$ will have a solution. Suppose the jth point is misclassified:

$$\ell'(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}} = 0$$

$$\Rightarrow -\frac{1}{m}\sum_{i=1,i\neq j}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}} = \frac{1}{m}\frac{c^{(j)}}{1+e^{-c^{(j)}\theta}}$$

The left part of the above $-\frac{1}{\mathrm{m}}\sum_{i=1,i\neq j}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}}$ are all the correctly classified points which will take the value in $(\frac{\sum_{i=1,i\neq j}^{m}\frac{-c^{(i)}}{2}}{m}, 0)$ when $\theta > 0$, or $(0, \frac{\sum_{i=1,i\neq j}^{m}\frac{-c^{(i)}}{2}}{m})$ when $\theta < 0$, it will have the similar plot as Figure 1; The right side of the above $\frac{1}{\mathrm{m}}\frac{c^{(j)}}{1+e^{-c^{(j)}\theta}}$ will take the value $(\frac{c^{(j)}}{2m}, \frac{c^{(j)}}{m})$ when $\theta > 0$, $c^{(j)} > 0$; $(\frac{c^{(j)}}{m}, \frac{c^{(j)}}{2m})$ when $\theta < 0$, $c^{(j)} < 0$. The below Figure 2 shows their relationship. It can be seen that no matter whether θ is positive or negative, there will always be an intersection between $f(\theta) = -\frac{1}{\mathrm{m}}\sum_{i=1,i\neq j}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}}$ and $f(\theta) = \frac{1}{\mathrm{m}}\frac{c^{(j)}}{1+e^{-c^{(j)}\theta}}$.

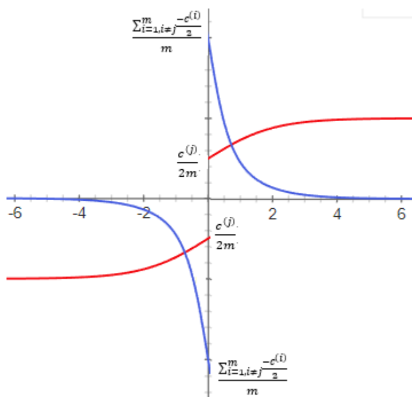


Figure 2, Gradient for both correct (blue) and misclassified(red) samples

Note that there is a possibility that even though there is a misclassified point, $-\frac{1}{m}\sum_{i=1,i\neq j}^{m}\frac{c^{(i)}}{1+e^{-c^{(i)}\theta}} = \frac{1}{m}\frac{c^{(j)}}{1+e^{-c^{(j)}\theta}}$ never have a solution, like the following, there is no intersection between red and blue plot:

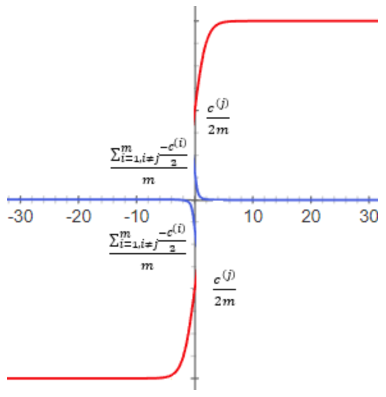


Figure 3, Not possible

It turns out that this is impossible so long as the data points are generated IID from the same distribution; there is always a solution for $\ell'(\theta) = 0$ in this case.

c) For each of these possible modifications, state whether it would lead to Convergence.

- Using a different constant learning rate.
  Answer:
  No. The problem is with the parameter being scaled infinitely to get a larger probability to get a maximum log-likelihood. Change of learning rate would not solve this problem.

- Decreasing the learning rate over time.
  Answer:
  No. the answer is same as above.

- Linear scaling of the input features.
  Answer:
  No. It has the same effect as to scale the parameters which doesn't help.

- Adding a regularization term $||\theta||_2^2$ to the loss function.
  Answer:
  Yes. L2 regularization will penalize large θ hence it will prevent the parameters to scale infinitely when it tries to maximize the log-likelihood.

- Adding zero-mean Gaussian noise to the training data or labels: will have misclassified points.
  Answer:
  Yes. It will add misclassified samples to the dataset which will make logistic regression converge.

d) In SVM, θ is going to be normalized and cannot be infinitely increased when trying to maximize the geometric margin; strict separation issue would no longer be a problem.

2. Model Calibration

a) Prove calibration condition holds true for Logistic regression over range (a, b) = (0, 1). Firstly, the log-likelihood of all points is given by

$$\ell(\theta) = \sum_{1}^{m} y^{(i)} \log p^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - p^{(i)}\right)$$

The maximum likelihood parameter θ is just the solution given by $\nabla_\theta \ell(\theta) = \vec{0}$. Note that $p^{(i)} = g\left(z^{(i)}\right),\ \nabla_\theta p^{(i)\prime} = g\left(z^{(i)}\right)\left(1 - g\left(z^{(i)}\right)\right) x^{(i)}$

$$\nabla_\theta \ell(\theta) = \sum_{1}^{m} (y^{(i)} p^{(i)}\,(1 - p^{(i)})/p^{(i)}\ + (1 - y^{(i)})(-p^{(i)})(1 - p^{(i)})/(1 - p^{(i)}))x^{(i)}$$

$$\nabla_\theta \ell(\theta) = \sum_{1}^{m} (y^{(i)}\,(1 - p^{(i)})\ + (1 - y^{(i)})(-p^{(i)}))x^{(i)} = \vec{0} \Rightarrow$$

$$\sum_{1}^{m} y^{(i)}\, x^{(i)} = \sum_{1}^{m} p^{(i)} x^{(i)}$$

with $y = \left[y^{(0)}, y^{(1)}, \dots, y^{(m)}\right], p = [p^{(0)}, p^{(1)}, \dots p^{(m)}]$. Use the fact that we include a bias term: $x_0^{(i)} = 1$, by observing the first component of $\sum_1^m y^{(i)}\, x^{(i)}$ and $\sum_1^m p^{(i)} x^{(i)}$, we have:

$$\sum_{1}^{m} y^{(i)} = \sum_{1}^{m} p^{(i)} \Rightarrow$$

$$\frac{\sum_{i \in I_{0,1}} 1\{y^{(i)} = 1\}}{\left|i \in I_{0,1}\right|} = \frac{\sum_{i \in I_{0,1}} P\left(y^{(i)} = 1 \middle| x^{(i)}; \theta\right)}{\left|i \in I_{0,1}\right|}$$

b) Perfect calibration doesn't mean perfect accuracy. If for any $(\mathrm{a}, \mathrm{b}) \subset [0,1]$ the property in the question holds true, consider two train examples with index i and j, they have different probability range $\mathrm{h}_\theta\left(x^{(i)}\right) \in (a^{(i)}, b^{(i)})$ and $\mathrm{h}_\theta\left(x^{(j)}\right) \in (a^{(j)}, b^{(j)})$. By switching two samples' probability ranges, the perfect calibration still holds true, but the two training examples will have different probability ranges separately.

Conversely if the model achieves perfect accuracy, it is perfectly calibrated. This can be explained by clapping $\mathrm{h}_\theta\left(x^{(i)}\right) \in (a^{(i)}, b^{(i)})$ to the probability of every single training sample, the perfect calibration condition holds true always.

c) What effect including L2 regularization in the logistic regression objective has on model calibration.

L2 regularization penalizes large value of $\left||\theta|\right|_2^2$, which makes $\mathrm{h}_\theta(x) = \frac{1}{1+\exp(-\theta^T x)}$ not able to grow infinitely to get a larger probability, which means the calibration range is smaller than (0,1).

3. Bayesian Interpretation of Regularization

a) Show that $\theta_{\text{MAP}} = \arg\max_{\theta} p(y|x,\theta)p(\theta)$ if we assume that $p(\theta) = p(\theta|x)$.

PROOF:

The posterior distribution of parameter θ is:

$$\begin{aligned} p(\theta|x,y) &= \frac{p(y|x,\theta)p(\theta|x)}{p(y|x)} \\ &= \frac{p(y|x,\theta)p(\theta|x)}{\int p(y|x,\theta)p(\theta|x)d\theta} \end{aligned}$$

The first equality is based on Bayesian rule with a condition, the second equality is the marginal probability over θ conditioned on x. The denominator of the second is independent on the parameter, so we have:

$$\begin{aligned} \theta_{MAP} &= \arg\max_{\theta} p(\theta|x,y) \\ &= \arg\max_{\theta} \frac{p(y|x,\theta)p(\theta|x)}{\int p(y|x,\theta)p(\theta|x)d\theta} \\ &= \arg\max_{\theta} p(y|x,\theta)p(\theta|x) \\ &= \arg\max_{\theta} p(y|x,\theta)p(\theta) \end{aligned}$$

Which is exactly what we need to prove.

b) Show that MAP estimation with zero-mean Gaussian prior is equivalent to applying L2 regularization with MLE.

PROOF:

From above we have the MAP estimation given by:

$$\theta_{\text{MAP}} = \arg\max_{\theta} p(y|x,\theta)p(\theta)$$

Change the max log-likelihood estimation to min negative log-likelihood estimation, and use the fact the θ is a zero mean Gaussian:

$$\begin{aligned} \theta_{\text{MAP}} &= \arg\min_{\theta} -\log\big(p(y|x,\theta)p(\theta)\big) \\ &= \arg\min_{\theta} -\log p(y|x,\theta) - \log\left(\frac{1}{\sqrt{2\pi}\eta}\exp\left(-\frac{\theta^T I\theta}{2\eta^2}\right)\right) \\ &= \arg\min_{\theta} -\log p(y|x,\theta) + \frac{\theta^T I\theta}{2\eta^2} \\ &= \arg\min_{\theta} -\log p(y|x,\theta) + \frac{||\theta||_2^2}{2\eta^2} \end{aligned}$$

From the last equality we have:

$$\lambda = \frac{1}{2\eta^2}$$

c) Closed form expression for θ_{MAP} with Gaussian prior and Gaussian distributed dataset

i. The Likelihood function is:

$$\mathrm{L}(\theta) = \prod_{\mathrm{i}=1}^{\mathrm{m}} p\left(y^{(i)} | x^{(i)}, \theta\right) p(\theta) = \prod_{\mathrm{i}=1}^{\mathrm{m}} \frac{1}{\sqrt{2\pi}\delta} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\delta^2}\right) \frac{1}{\sqrt{2\pi}\eta} \exp\left(-\frac{\theta^T I \theta}{2\eta^2}\right)$$

ii. The negative log-likelihood loss can then be derived from above:

$$\ell(\theta) = \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\delta^2} + \frac{m}{2\eta^2} \theta^T I \theta - m \log(\sqrt{2\pi}\delta) - m \log(\sqrt{2\pi}\eta)$$

The MAP estimation is just the solution at the stationary point. Take derivative w.r.t. θ and set it to $\vec{0}$:

$$\sum_{\mathrm{i}=1}^{\mathrm{m}} -\frac{(y - \theta^T x) x}{\delta^2} + \frac{m}{\eta^2} \theta = \vec{0}$$

Vectorize the above sum, $\mathrm{X} \in \mathrm{R}^{\mathrm{m}\times\mathrm{n}}, \theta \in \mathrm{R}^{\mathrm{n}}, \vec{y} \in \mathrm{R}^{m}$:

$$\frac{\mathrm{X}^{\mathrm{T}}(\vec{y} - X\theta)}{\delta^2} = \frac{mI}{\eta^2} \theta \Rightarrow \frac{\mathrm{X}^{\mathrm{T}}\vec{y}}{\delta^2} = \left(\frac{X^T X}{\delta^2} + \frac{m}{\eta^2} I\right) \theta \Rightarrow$$

$$\theta = \left(X^T X + \frac{\delta^2 m}{\eta^2} I\right)^{-1} \mathrm{X}^{\mathrm{T}} \vec{y}$$

So the closed form $\theta_{\text{MAP}} = \left(X^T X + \frac{\delta^2 m}{\eta^2} I\right)^{-1} \mathrm{X}^{\mathrm{T}} \vec{y}$.

d) Show that θ_{MAP}in the case of Laplace prior is equivalent to the solution of a linear regression with L1 regularization.

Like c), the Likelihood function is given by

$$\mathrm{L}(\theta) = \prod_{\mathrm{i}=1}^{\mathrm{m}} p\left(y^{(i)} | x^{(i)}, \theta\right) p(\theta) = \prod_{\mathrm{i}=1}^{\mathrm{m}} \frac{1}{\sqrt{2\pi}\delta} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\delta^2}\right) \frac{1}{2b} \exp\left(-\frac{|\theta|}{b}\right)$$

The negative log-likelihood loss is then:

$$\mathrm{J}(\theta) = \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\delta^2} + \frac{m}{b} |\theta|$$

Scale the loss doesn’t change anything to the problem, so the loss can just be written as:

$$\begin{aligned} \mathrm{J}(\theta) &= \sum_{\mathrm{i}=1}^{\mathrm{m}} \left(y^{(i)} - \theta^T x^{(i)}\right)^2 + \frac{2\delta^2 m}{b} \left||\theta|\right|_1 \\ &= (\mathrm{X}\theta - \vec{y})^T (\mathrm{X}\theta - \vec{y}) + \frac{2\delta^2 m}{b} \left||\theta|\right|_1 \\ &= \left||\mathrm{X}\theta - \vec{y}|\right|_2^2 + \frac{2\delta^2 m}{b} \left||\theta|\right|_1 \end{aligned}$$

Where $\lambda = \frac{2\delta^2 m}{b}$

4. **Constructing kernels.**

Apply Mercier's theorem, to show a matrix is a kernel matrix, just need to show that the matrix is symmetric Positive Semi-Definite. (For the ease of readability, $\mathrm{K1}$ And K_1 are the same thing in this answer.)

a) $K(x,z) = K_1(x,z) + K_2(x,z)$:
It's a kernel, it is obvious that if K1 and K2 are both symmetric and positive semi-definitive.
Frist, K is symmetric because, from:

$$\begin{aligned} K1_{ij} &= K1_{ji} \\ K2_{ij} &= K2_{ji} \\ &\Rightarrow \\ K1_{ij} + K2_{ij} &= K1_{ji} + K2_{ji} \Rightarrow \\ K_{ij} &= K_{ji} \end{aligned}$$

Second, K is positive semi-definitive, because both K1 and K2 are PSD, K=K1+K2 is also a PSD matrix:

$$\mathrm{z}^{\mathrm{T}}Kz = z^T(K1+K2)z = z^TK1\,z + z^TK2\,z \geq 0$$

b) $K(x,z) = K_1(x,z) - K_2(x,z)$:
Not necessarily a kernel. Same as (a) it is symmetric, but not always positive definitive.
For example, $\mathrm{K1} = \mathrm{I}$, $\mathrm{K2} = 2\mathrm{I}$ (I is identity matrix), they both PSDs, but $\mathrm{K1} - \mathrm{K2} = -\mathrm{I}$ is negative definite.

c) $K(x,z) = a\,K_1(x,z)$:
It is a kernel only when a >= 0. If a<0, $\mathrm{K1} = \mathrm{I}$, then $\mathrm{aK1(x,z)}$ is negative definite.

d) $K(x,z) = -a\,K_1(x,z)$:
Similar to c), It is a kernel only when a <= 0. For example, a = 1, $\mathrm{K1} = \mathrm{I}$, we have $\mathrm{z}^{\mathrm{T}}K1\,z = -az^Tz \leq 0$

e) $K(x,z) = K_1(x,z)K_2(x,z)$:
It is a kernel since because,

$$\begin{aligned} K1(x,z)K2(x,z) &= K2^T(x,z)K1^T(x,z) = K2(x,z)K1(x,z) \\ K_{ij} &= \sum_k K1_{ik}K2_{kj} \\ K_{ji} &= \sum_k K1_{jk}K2_{ki} = \sum_k K1_{kj}K2_{ik} = \sum_k K2_{ik}K1_{kj} \end{aligned}$$

Summarize the above 3 equations, $K_{ij} = K_{ji}$ which means K is symmetric;
since both K1 and K2 are PSD, for any given vector x, we have $x^TK1\ x >= 0, x^TK2\,x >= 0$, it implies that:

$$\begin{aligned} &x^TK1\,xx^TK2\,x \geq 0 \\ &\Rightarrow x^TK1\,K2\,x \geq 0 \end{aligned}$$

f) $K(x,z) = f(x)f(z)$:
It is a kernel. From class we know that for a feature mapping $\phi(\mathrm{x})$, $\mathrm{K(x,z)} = \phi(\mathrm{x})^{\mathrm{T}}\phi(z)$ is a kernel. In this case we simply can set $\phi(\mathrm{x}) = \mathrm{f(x)}$, it's just a 1d mapping.

g) $K(x,z) = K_3(\phi(x),\phi(z))$:
It is a kernel. Since K3 is a Kernel, $\mathrm{K(x,z)} = \mathrm{K3}(\phi(\mathrm{x}),\phi(\mathrm{z})) = \psi(\phi(x))^T\psi(\phi(z)) = (\psi \circ \phi(x))^T(\psi \circ \phi(z))$ for a feature mapping $\psi \circ \phi$.

h) $K(x,z) = p\left(K_1(x,z)\right)$:

It is a kernel. First, $K_1^n(x,z)$ is still a kernel; $aK_1(x,z)$ is still a kernel when $a \geq 0$, $K_1(x,z) + c$ is a kernel if $c \geq 0$; combine them together we have that $p\left(K_1(x,z)\right)$ is still a kernel.

5. Kernelizing the Perceptron

a) Kernel trick questions.

i. How you would apply the "kernel trick" to the perceptron

Answer:

Observe the update rule, the series of $\theta^{(i)}$ is given by:

$$\theta^{(i+1)} = \theta^{(i)} + \alpha\left(y^{(i+1)} - g\left(\theta^{(i)}\phi\left(x^{(i+1)}\right)\right)\right)\phi\left(x^{(i+1)}\right)$$

Since θ is infinite dimensional and cannot be represented by computer, the relationship between two successive $\theta^{(i)}$ and $\theta^{(i+1)}$ however can be calculated iteratively, the values are just scalers which are denoted by $\beta^{(i)}$ given by:

$$\beta^{(1)} = \alpha\left(y^{(1)} - g\left(\theta^{(0)}\phi\left(x^{(1)}\right)\right)\right), \theta^{(0)}\phi\left(x^{(1)}\right) = \left(\theta^{(0)}\right)\phi\left(x^{(1)}\right) = 0$$
$$\beta^{(2)} = \alpha\left(y^{(2)} - g\left(\theta^{(1)}\phi\left(x^{(2)}\right)\right)\right), \theta^{(1)}\phi\left(x^{(2)}\right) = \left(\theta^{(0)} + \beta^{(1)}\phi\left(x^{(1)}\right)\right)\phi\left(x^{(2)}\right)$$
$$\beta^{(3)} = \alpha\left(y^{(3)} - g\left(\theta^{(2)}\phi\left(x^{(3)}\right)\right)\right), \theta^{(2)}\phi\left(x^{(3)}\right) = \left(\theta^{(0)} + \beta^{(1)}\phi\left(x^{(1)}\right) + \beta^{(2)}\phi\left(x^{(2)}\right)\right)\phi\left(x^{(3)}\right)$$

...

$$\beta^{(i+1)} = \alpha\left(y^{(i+1)} - g\left(\theta^{(i)}\phi\left(x^{(i+1)}\right)\right)\right), \theta^{(i)}\phi\left(x^{(i+1)}\right)$$
$$= \left(\theta^{(0)} + \beta^{(1)}\phi\left(x^{(1)}\right) + \beta^{(2)}\phi\left(x^{(2)}\right) + \cdots + \beta^{(i)}\phi\left(x^{(i)}\right)\right)\phi\left(x^{(i+1)}\right)$$

We can see that $\beta^{(i+1)} = \sum_{k=1}^{i}\beta^{(k)}\phi\left(x^{(k)}\right)\phi\left(x^{(i+1)}\right) = \sum_{k=1}^{i}\beta^{(k)}Kernel\left(x^{(k)}, x^{(i+1)}\right)$

We do not need to store any $\theta^{(i)}$ value, including $\theta^{(0)}$, only need to store the array of $\beta^{(i)}s$. Whenever there is a need for $\theta^{(i)}$, it is just used to calculate the margin $\theta^{(i)}\phi(x)$ which can just be written as $\sum_{k=1}^{i}\beta^{(k)}Kernel(x^{(k)}, x)$. For $\theta^{(0)}$ the margin is just $\theta^{(0)}\phi(x) = 0$.

ii. How you will efficiently make a prediction on a new input $x^{(i+1)}$.

Answer:

As described in i), we only need to figure out the margin value $\theta^{(i)}\phi\left(x^{(i+1)}\right)$ which is just given by:

$$z = \theta^{(i)}\phi\left(x^{(i+1)}\right) = \sum_{k=1}^{i}\beta^{(k)}Kernel\left(x^{(k)}, x^{(i+1)}\right)$$

And use g(z)=sign(z) to make a prediction.

iii. How you will modify the update rule given above to perform an update to θ

Answer:

As described in i), θ is never explicitly represented by computer, only the linear coefficients $\beta^{(i)}s$ are stored; The learning update will only update the coefficients. Whenever the calculation of margins needed, the coefficients are used to calculate margins:

$$z = \theta^{(i)}\phi(x) = \sum_{k=1}^{i}\beta^{(k)}Kernel\left(x^{(k)}, x\right)$$

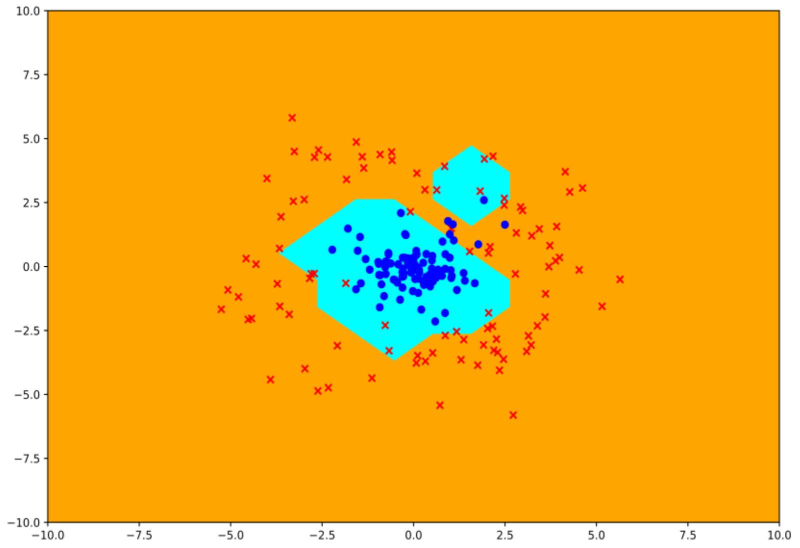
b) Implement kernerlized perceptron.

Answer:

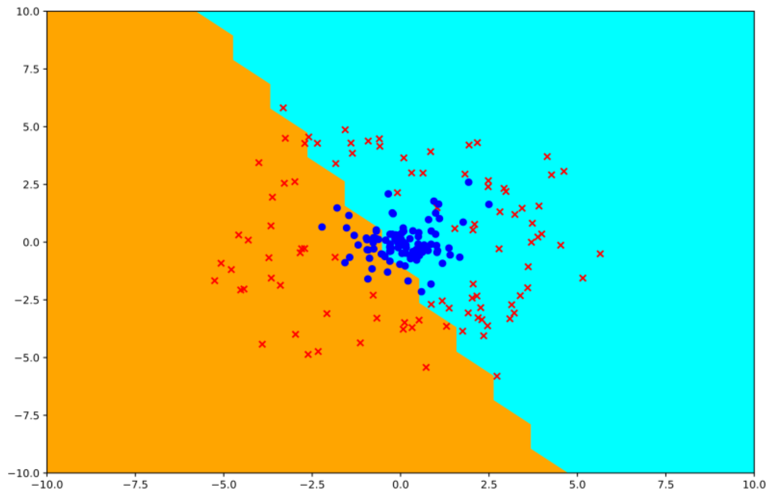
c) Which kernel performs badly and why.

Answer:

From the plots we can see that RBT kernels makes pretty decent predictions and indeed maps to higher dimensional features;



However the dot product kernel cannot make good prediction; It divides the sample space into two halves with a waved line.



The reason for this is because the feature map doesn’t contain multiple degree monomials, which means it cannot regress complex decision boundary.