1. **[15 points] Logistic Regression: Training stability**

In this problem, we will be delving deeper into the workings of logistic regression. The goal of this problem is to help you develop your skills debugging machine learning algorithms (which can be very different from debugging software in general).

We have provided a implementation of logistic regression in `src/p01_lr.py`, and two labeled datasets A and B in `data/ds1_a.txt` and `data/ds1_b.txt`

Please do not modify the code for the logistic regression training algorithm for this problem. First, run the given logistic regression code to train two different models on A and B. You can run the code by simply executing `python p01_lr.py` in the `src` directory.

(a) [2 points] What is the most notable difference in training the logistic regression model on datasets A and B?

The model has converged on dataset A with 278,192 iterations.

whereas not able to converge with tons of iterations.

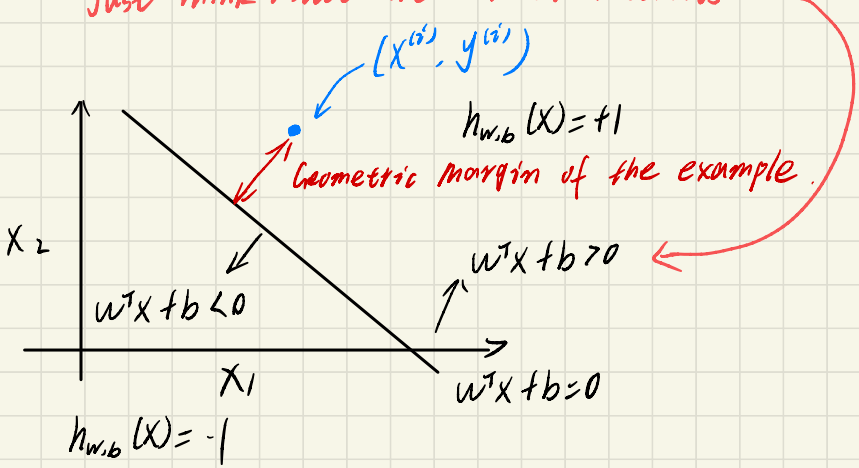
(b) [5 points] Investigate why the training procedure behaves unexpectedly on dataset B, but not on A. Provide hard evidence (in the form of math, code, plots, etc.) to corroborate your hypothesis for the misbehavior. Remember, you should address why your explanation does *not* apply to A.

Hint: The issue is not a numerical rounding or over/underflow error.

Notice dataset A is not linearly seperatable, whereas dataset B is able to. When calculate margin (in grad), $(y \in \{-1, 1\})$

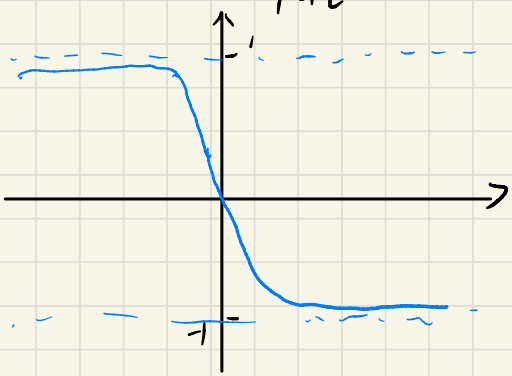
$\text{margin}^{(i)} = y^{(i)} \cdot x^{(i)}\theta$, it turns out, if the points are seperated by the line (with param θ), margin will be positive.

Just think about the similar restricts in SVM



Also, the probs function:

$$probs = \frac{1}{1+e^{z}}$$ looks like



$\therefore \forall z > 0$, probs will be very small.

hence, the abs value of grad will be small. compared to $z < 0$.

$\therefore$ The points that are misclassified contribute probs, thus, grad.

For dataset A, misclassified points alway exist as it is not linearly seperatable.

Grads from dataset B are always very small, it will converge with very large number of iterations.

(c) [5 points] For each of these possible modifications, state whether or not it would lead to the provided training algorithm converging on datasets such as B. Justify your answers.

i. Using a different constant learning rate.

ii. Decreasing the learning rate over time (e.g. scaling the initial learning rate by $1/t^2$, where t is the number of gradient descent iterations thus far).

iii. Linear scaling of the input features.

iv. Adding a regularization term $\|\theta\|_2^2$ to the loss function.

v. Adding zero-mean Gaussian noise to the training data or labels.

i. No, does not resolve the problem mentioned

ii. yes, as $\frac{1}{(11!)^2} \approx 6.28 \times 10^{-16}$, the algo will converge anyway within 11 iterations.

iii. No, scaling do not change decision line as indicated in svm

iv.

The cost function is defined by likelihood:

$P(y=1|x;\theta) = h_\theta(x)$ where $h_\theta(x)$ is the sigmoid function:

$P(y=-1|x;\theta) = 1-h_\theta(x)$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

$$1-h_\theta(x) = \frac{1}{1+e^{\theta^T x}}$$

$$\Downarrow$$

$$P(y|x;\theta) = \frac{1}{1+e^{-y\theta^T x}}$$

$$\therefore \quad l(\theta) = \log(L(\theta)) = -\sum_{i=1}^{m} \log\left(1+e^{-y^{(i)}\theta^T x^{(i)}}\right)$$

$$\downarrow$$

$$\theta = \arg\max_\theta \left(l(\theta)\right) \quad \text{or} \quad \theta = \arg\min_\theta \left(\log\left(1+\exp(-y^{(i)}\theta^T x^{(i)})\right)\right)$$

Let $J(\theta) = \sum_{i=1}^{m} \log(1+\exp(-y^{(i)}\theta^T x^{(i)})) + \|\theta\|_2^2$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{1+\exp(-y^{(i)}\theta^T x^{(i)})}(-y^{(i)}x_j^{(i)}) + 2\theta_j$$

L_2 Regularization works.

v. Yes, adding noise will add the points contributing grad to update θ

(d) [3 points] Are support vector machines, which use the hinge loss, vulnerable to datasets like B? Why or why not? Give an informal justification.

Hint: Recall the distinction between functional margin and geometric margin.

Hinge loss:

$l(\hat{y}) = \max(0, 1-y\cdot\hat{y})$ (1) where $\hat{y} = (w^T x^{(i)} + b)$

Notice for geometric margin, it needs

$\min_{w,b} \|w\|^2$

s.t. $y^{(i)}(w^T x^{(i)} + b) \geq 1$

when $y^{(i)}(w^T x^{(i)} + b) \geq 1$, $l(\hat{y}) = 0$, the algo converged.

2. **[10 points] Model Calibration**

In this question we will try to understand the output $h_\theta(x)$ of the hypothesis function of a logistic regression model, in particular why we might treat the output as a probability (besides the fact that the sigmoid function ensures $h_\theta(x)$ always lies in the interval $(0, 1)$).

When the probabilities outputted by a model match empirical observation, the model is said to be *well-calibrated* (or reliable). For example, if we consider a set of examples $x^{(i)}$ for which $h_\theta(x^{(i)}) \approx 0.7$, around 70% of those examples should have positive labels. In a well-calibrated model, this property will hold true at every probability value.

Logistic regression tends to output well-calibrated probabilities (this is often not true with other classifiers such as Naive Bayes, or SVMs). We will dig a little deeper in order to understand why this is the case, and find that the structure of the loss function explains this property.

Suppose we have a training set $\{x^{(i)}, y^{(i)}\}_{i=1}^m$ with $x^{(i)} \in \mathbb{R}^{n+1}$ and $y^{(i)} \in \{0, 1\}$. Assume we have an intercept term $x_0^{(i)} = 1$ for all i. Let $\theta \in \mathbb{R}^{n+1}$ be the maximum likelihood parameters learned after training a logistic regression model. In order for the model to be considered well-calibrated, given any range of probabilities (a, b) such that $0 \le a < b \le 1$, and training examples $x^{(i)}$ where the model outputs $h_\theta(x^{(i)})$ fall in the range (a, b), the fraction of positives in that set of examples should be equal to the average of the model outputs for those examples. That is, the following property must hold:

$$\frac{\sum_{i \in I_{a,b}} P\left(y^{(i)} = 1 | x^{(i)}; \theta\right)}{|\{i \in I_{a,b}\}|} = \frac{\sum_{i \in I_{a,b}} \mathbb{I}\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|},$$

where $P(y = 1|x; \theta) = h_\theta(x) = 1/(1 + \exp(-\theta^\top x))$, $I_{a,b} = \{i | i \in \{1, ..., m\}, h_\theta(x^{(i)}) \in (a, b)\}$ is an index set of all training examples $x^{(i)}$ where $h_\theta(x^{(i)}) \in (a, b)$, and $|S|$ denotes the size of the set S.

(a) [5 points] Show that the above property holds true for the described logistic regression model over the range $(a, b) = (0, 1)$.

Hint: Use the fact that we include a bias term.

$\forall i \in \{1, \dots, m\}$, $h_\theta(x^{(i)}) \in (0,1)$

Since the model is well trained, the

$$\text{gradient} = \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} = 0$$

$$\therefore \sum_{i=1}^{m} y^{(i)} = \sum_{i=1}^{m} h_\theta(x^{(i)})$$

$$\sum_{i=1}^{m} y^{(i)} = \sum_{i=1}^{m} I\{y^{(i)} = 1\} \quad \text{as } y^{(i)} \in \{0, 1\}$$

$$\therefore \sum_{i=1}^{m} I\{y^{(i)} = 1\} = \sum_{i=1}^{m} P(y^{(i)} = 1 | x^{(i)}; \theta)$$

as when $(a,b) = (0,1) \Rightarrow \Sigma_{i \in I_{a,b}} = \sum_{i=1}^{m}$

(b) [3 points] If we have a binary classification model that is perfectly calibrated—that is, the property we just proved holds for any $(a, b) \subset [0, 1]$—does this necessarily imply that the model achieves perfect accuracy? Is the converse necessarily true? Justify your answers.

assume $(a,b) = (0.5, 1)$, hence

if the model is 100% accurate,

$$\sum_{i \in I_{a,b}} I\{y^{(i)} = 1\} = |\{i \in I_{a,b}\}|$$

whereas $0.5 < P(y^{(i)} = 1 \mid x^{(i)}; \theta) < 1$

$$0.5 < \frac{\sum_{i \in I_{a,b}} P(y^{(i)} = 1 \mid x^{(i)}; \theta)}{|\{i \in I_{a,b}\}|} < 1$$

$$\therefore \frac{\sum_{i \in I_{a,b}} P(y^{(i)} = 1 \mid x^{(i)}; \theta)}{|\{i \in I_{a,b}\}|} < \frac{\sum_{i \in I_{a,b}} I\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|}$$

Converse dose not hold neither.

(c) [2 points] Discuss what effect including L_2 regularization in the logistic regression objective has on model calibration.

Regularized grad:

$$\sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)} + 2\theta_j = 0$$

for $j = 0$, $x_j^{(i)} = x_0^{(i)} = 1$

$$\sum_{i=1}^{m} y^{(i)} = \sum_{i=1}^{m} h_\theta(x^{(i)}) - 2\theta_0$$

model is not well-calibrated.

3.

In this problem, we explore the connection between MAP estimation, and common regularization techniques that are applied with MLE estimation. In particular, you will show how the choice of prior distribution over θ (e.g Gaussian, or Laplace prior) is equivalent to different kinds of regularization (e.g L_2, or L_1 regularization). To show this, we shall proceed step by step, showing intermediate steps.

(a) [3 points] Show that $\theta_{\text{MAP}} = \text{argmax}_\theta \, p(y|x,\theta)p(\theta)$ if we assume that $p(\theta) = p(\theta|x)$. The assumption that $p(\theta) = p(\theta|x)$ will be valid for models such as linear regression where the input x are not explicitly modeled by θ. (Note that this means x and θ are marginally independent, but not conditionally independent when y is given.)

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

$$P(\theta|x,y) = \frac{P(y|\theta,x)\,P(\theta|x)}{P(y|x)}$$

given by dataset

$$\propto P(y|x,\theta)\,P(\theta)$$

$$\therefore \quad \arg\max_\theta P(\theta|x,y) = \arg\max_\theta P(y|x,\theta)\,P(\theta)$$

(b) [5 points] Recall that L_2 regularization penalizes the L_2 norm of the parameters while minimizing the loss (i.e., negative log likelihood in case of probabilistic models). Now we will show that MAP estimation with a zero-mean Gaussian prior over θ, specifically $\theta \sim \mathcal{N}(0, \eta^2 I)$, is equivalent to applying L_2 regularization with MLE estimation. Specifically, show that

$$\theta_{\text{MAP}} = \arg\min_\theta - \log p(y|x,\theta) + \lambda ||\theta||_2^2.$$

Also, what is the value of λ?

$$J(\theta) = -\ell(\theta) = -\log P(y|x,\theta) + \lambda \|\theta\|_2^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = -\frac{1}{P(y|x,\theta)} \frac{\partial}{\partial \theta_j} P(y|x,\theta) + 2\lambda\theta_j \quad (1)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0$$

$$\Rightarrow \quad \frac{\partial}{\partial \theta_j} P(y|x,\theta) - 2\lambda\theta_j \cdot P(y|x,\theta) = 0 \quad (2)$$

$$J_{map}(\theta) = p(y|X,\theta)\,p(\theta) = \frac{1}{(2\pi)^{n/2}\eta^n} \exp\left(\frac{\theta^2}{2\eta^2}\right) p(y|X,\theta)$$

$$\frac{\partial}{\partial \theta_j} J_{map}(\theta) = \frac{2\theta_j/2\eta^2}{(2\pi)^{n/2}\eta^n} \exp\left(\frac{\theta^2}{2\eta^2}\right) p(y|X,\theta)$$

$$+ \frac{1}{(2\pi)^{n/2}\eta^n} \exp\left(\frac{\theta^2}{2\eta^2}\right) \cdot \frac{\partial}{\partial \theta_j} p(y|X,\theta)$$

$J_{map}(\theta) = 0$ as $\theta \sim N(0, \eta^2 I)$, Multivariable Gaussian Distribution

$\Rightarrow$

$$\frac{\partial}{\partial \theta_j} p(y|X,\theta) - \frac{\theta_j}{\eta^2} p(y|X,\theta) = 0 \quad (3)$$

$$\frac{\partial}{\partial \theta_j} p(y|X,\theta) - 2\lambda\theta_j \cdot p(y|X,\theta) = 0 \quad (2)$$

let $2\lambda = \frac{1}{\eta^2}$ or $\lambda = \frac{1}{2\eta^2}$, (2) is equivalent to (3)

(c) [7 points] Now consider a specific instance, a linear regression model given by $y = \theta^T x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Like before, assume a Gaussian prior on this model such that $\theta \sim \mathcal{N}(0, \eta^2 I)$. For notation, let X be the design matrix of all the training example inputs where each row vector is one example input, and $\vec{y}$ be the column vector of all the example outputs.

Come up with a closed form expression for θ_{MAP}.

$$\theta_{map} = \arg\min_{\theta} -\log p(y|\theta, x) + \lambda \|\theta\|_2^2$$

$$\epsilon = \theta^T x - y$$

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

$$-\log p(y|\theta, x)$$
$$= -\log p(y = \theta^T x + \epsilon)$$
$$= -\log p(\epsilon = y - \theta^T x)$$
$$= -\log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(y-\theta^T x)^2}{2\sigma^2}\right)$$
$$= -\left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{(y-\theta^T x)^2}{2\sigma^2}\right]$$

$$\frac{\partial}{\partial \theta} \|X\theta - \vec{y}\|_2^2 = 2X^T X\theta - 2X^T\vec{y}$$

$$\theta_{map} = \arg\min_{\theta} \frac{(y-\theta^T x)^2}{2\sigma^2} + \frac{1}{2\eta^2}\|\theta\|_2^2$$

$$J(\theta) = \frac{1}{2\sigma^2}(\vec{y} - X\theta)^T(\vec{y} - X\theta) + \frac{1}{2\eta^2}\|\theta\|_2^2$$

$$\nabla_\theta J(\theta) = \nabla_\theta\left(\frac{1}{2\sigma^2}(\vec{y}^T\vec{y} - \vec{y}^T X\theta - \theta^T X^T\vec{y} + \theta^T X^T X\theta) + \frac{1}{2\eta^2}\|\theta\|_2^2\right)$$

$$= \frac{1}{2\sigma^2}(-2X^T\vec{y} + 2X^T X\theta) + \frac{1}{\eta^2}\theta$$

$$\nabla_\theta J(\theta) = 0$$

$$\Rightarrow (X^T X\theta - X^T\vec{y}) + \frac{\sigma^2}{\eta^2}\theta = 0$$

$$\left(X^T X - \frac{\sigma^2}{\eta^2} I\right)\theta = X^T\vec{y} \qquad \theta = \left(X^T X - \frac{\sigma^2}{\eta^2} I\right)^{-1} X^T\vec{y}$$

$$\theta_{map} = \arg\min_{\theta} J(\theta) = \left(X^T X - \frac{\sigma^2}{\eta^2} I\right) X \vec{y}$$

(d) [5 points] Next, consider the Laplace distribution, whose density is given by

$$f_{\mathcal{L}}(z|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|z-\mu|}{b}\right).$$

As before, consider a linear regression model given by $y = x^T\theta + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Assume a Laplace prior on this model where $\theta \sim \mathcal{L}(0, bI)$.

Show that θ_{MAP} in this case is equivalent to the solution of linear regression with L_1 regularization, whose loss is specified as

$$J(\theta) = ||X\theta - \vec{y}||_2^2 + \gamma||\theta||_1$$

Also, what is the value of γ?

Note: A closed form solution for linear regression problem with L_1 regularization does not exist. To optimize this, we use gradient descent with a random initialization and solve it numerically.

$$\theta_{map} = \arg\max_{\theta} P(y|X,\theta) P(\theta)$$

$$\Rightarrow \arg\max_{\theta} \log\left(P(y|X,\theta)\right) + \log\left(P(\theta)\right)$$

$$\Rightarrow \arg\max_{\theta} -\frac{\|X\theta - \vec{y}\|_2^2}{2\sigma^2} - \frac{|\theta|}{b}$$

$$\Rightarrow \arg\min_{\theta} \frac{\|X\theta - \vec{y}\|_2^2}{2\sigma^2} + \frac{|\theta|}{b}$$

$$\propto \|X\theta - \vec{y}\|_2^2 + \frac{2\sigma^2}{b}|\theta| = J(\theta)$$

$$\boxed{\gamma = \frac{2\sigma^2}{b}}$$

4.

(a) [1 points] $K(x,z) = K_1(x,z) + K_2(x,z)$

(b) [1 points] $K(x,z) = K_1(x,z) - K_2(x,z)$

(c) [1 points] $K(x,z) = aK_1(x,z)$

(d) [1 points] $K(x,z) = -aK_1(x,z)$

(e) [5 points] $K(x,z) = K_1(x,z)K_2(x,z)$

(f) [3 points] $K(x,z) = f(x)f(z)$

(g) [3 points] $K(x,z) = K_3(\phi(x), \phi(z))$

(h) [3 points] $K(x,z) = p(K_1(x,z))$

(a) Yes, symmetric & PSD

$z^T K z = z^T (K_1 + K_2) z = z^T K_1 z + z^T K_2 z \geq 0 \Rightarrow$ PSD

(b) No, not necessarily PSD

$z^T K z = z^T (K_1 - K_2) z$, let $K_2 = 2K_1$,

$\Rightarrow z^T K z = -z^T K_1 z \leq 0$.

(c) Yes, symmetric & PSD

$z^T K z = z^T a K_1 z = a z^T K_1 z \geq 0$

(d) No, symmetric but not PSD.

$z^T K z = z^T (-a K_1) z = -a z^T K_1 z \leq 0$

(e) Yes, symmetric & PSD

① $z_1^T K_1 z_1 \geq 0$ where $z_1, z_2 \in \mathbb{R}^n$, s.t. $z_1 z_2^T = I$

② $z_2^T K_2 z_2 \geq 0$ $K_1, K_2 \in \mathbb{R}^{n \times n}$ $z_2 z_1^T = I$

$z_2 z_1^T z_1 = I z_1$

$z_2 = c \cdot I z_1$, where $c \geq 0$

① × ② $\Rightarrow$ $z_1^T K_1 z_1 z_2^T K_2 z_2 \geq 0$

$z_1^T K_1 K_2 z_2 \geq 0$

$z_1^T K_1 K_2 z_1 \geq 0$

Alternatively:

$$z^T K z = \sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij} z_i z_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} k_1(x^{(i)}, x^{(j)})\, k_2(x^{(i)}, x^{(j)})\, z_i z_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \phi_1(x^{(i)})^T \phi_1(x^{(j)})\, \phi_2(x^{(i)})^T \phi_2(x^{(j)})\, z_i z_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \sum_{\alpha=1}^{d} \phi_{1\alpha}(x^{(i)})\phi_{1\alpha}(x^{(j)}) \sum_{\beta=1}^{d} \phi_{2\beta}(x^{(i)})\phi_{2\beta}(x^{(j)})\, z_i z_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \sum_{\alpha=1}^{d}\sum_{\beta=1}^{d} \phi_{1\alpha}(x^{(i)})\phi_{1\alpha}(x^{(j)})\phi_{2\beta}(x^{(i)})\phi_{2\beta}(x^{(j)})\, z_i z_j$$

$$= \sum_{i=1}^{n} \sum_{\alpha=1}^{d}\sum_{\beta=1}^{d} \left(\phi_{1\alpha}(x^{(i)})\phi_{2\beta}(x^{(i)})\, z_i\right)^2 \geq 0$$

(f) Yes.

$$z^T K z = \sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij} z_i z_j$$

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

$$= \sum_{i=1}^{n} \left(f(x^{(i)}) z_i\right)^2 \geq 0$$

(g) Yes. K_3 is a valid kernel

(h) Yes

Accounting to (e), any power of $k_1(x,z)$ should be kernel, also, the sum with positive coefficients should be kernel if individuals are kernels, from (a), (c)

$$K(x,z) = \sum_{k=0}^{n} c_k (k_1(x,z))^k$$

$(k_1(x,z))^k$ → kernel. from (e)

$K(x,z)$ ↓ kernel, from (a), (c)

5.

(a) [9 points] Let K be a Mercer kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, ∞-dimensional) that it's infeasible to ever represent $\phi(x)$ explicitly. Describe how you would apply the "kernel trick" to the perceptron to make it work in the high-dimensional feature space ϕ, but without ever explicitly computing $\phi(x)$.

[**Note:** You don't have to worry about the intercept term. If you like, think of ϕ as having the property that $\phi_0(x) = 1$ so that this is taken care of.] Your description should specify:

i. [3 points] How you will (implicitly) represent the high-dimensional parameter vector $\theta^{(i)}$, including how the initial value $\theta^{(0)} = 0$ is represented (note that $\theta^{(i)}$ is now a vector whose dimension is the same as the feature vectors $\phi(x)$);

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

since $\theta^{(1)} := \vec{0} + C_1 x^{(1)}$, $\theta^{(2)} = C_1 x^{(1)} + C_2 x^{(2)}$ ------

where C_k are constants

as $y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}) \in \{0, \pm 1\}$

$$\therefore \theta^{(i)} = \sum_{j=1}^{i} C_j x^{(j)} \qquad \theta^{(0)} = \sum_{j=1}^{0} C_j x^{(j)} = \vec{0}$$

when kernel function applied:

$$\theta^{(i)} = \sum_{j=1}^{i} C_j \varphi(x^{(j)}) \qquad \theta^{(0)} = \sum_{j=1}^{0} C_j \varphi(x^{(j)}) = \vec{0}$$

ii. [3 points] How you will efficiently make a prediction on a new input $x^{(i+1)}$. I.e., how you will compute $h_{\theta^{(i)}}(x^{(i+1)}) = g(\theta^{(i)^T}\phi(x^{(i+1)}))$, using your representation of $\theta^{(i)}$; and

$$h_{\theta^{(i)}}(x^{(i+1)}) = g(\theta^{(i)T}\varphi(x^{(i+1)}))$$
$$= \text{sign}\left(\sum_{j=1}^{i} c_j \varphi^T(x^{(j)}) \cdot \varphi(x^{(i+1)})\right)$$
$$= \text{sign}\left(\sum_{j=1}^{i} c_j K(x^{(j)}, x^{(i+1)})\right)$$

iii. [3 points] How you will modify the update rule given above to perform an update to θ on a new training example $(x^{(i+1)}, y^{(i+1)})$; *i.e.*, using the update rule corresponding to the feature mapping ϕ:

$$\theta^{(i+1)} := \theta^{(i)} + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}))x^{(i+1)}$$

$$\theta^{(i+1)} := \sum_{j=1}^{i} c_j \varphi(x^{(j)}) + \alpha\left[y^{(i+1)} - \text{sign}\left(\sum_{j=1}^{i} c_j K(x^{(j)}, x^{(i+1)})\right)\right]\varphi(x^{(i+1)})$$

$$\Rightarrow$$

$$\theta^{(i+1)} := \sum_{j=1}^{i} c_j \varphi(x^{(j)}) + c_{i+1}\varphi(x^{(i+1)})$$

(c) [2 points] Run `src/p05_percept.py` to train kernelized perceptrons on `data/ds5_train.csv`. The code will then test the perceptron on `data/ds5_test.csv` and save the resulting predictions in the `src/output` folder. Plots will also be saved in `src/output`. We provide two kernels, a dot product kernel and an radial basis function (rbf) kernel. One of the provided kernels performs extremely poorly in classifying the points. Which kernel performs badly and why does it fail?

Dot kernel perform badly, as $\varphi(x) = x$ which did not change to a higher space. Also, the current space cannot be linearly seperatable.