1.a

From the course material, we have that exponential family models have probability distributions of the form

$$p(y;\eta) = b(y)\exp\left(\eta^T T(y) - a(\eta)\right)$$

where η is the natural parameter. We rearrange the given Poisson distribution to arrive at the desired form

$$\begin{aligned} p(y;\lambda) &= \frac{e^{-\lambda}\lambda^y}{y!} \\ &= \frac{1}{y!}\exp\left(-\lambda\right)\lambda^y \\ &= \frac{1}{y!}\exp\left(-\lambda\right)\exp\left(y\log(\lambda)\right) \\ &= \frac{1}{y!}\exp\left(y\log(\lambda) - \lambda\right). \end{aligned}$$

This results in $b(y) = 1/y!$, $\eta = \log(\lambda)$, $T(y) = y$, and $a(\eta) = \lambda = \exp\left(\eta\right)$. We conclude that the Poisson distribution is in the exponential family.

1.b

The canonical response function, $g(\eta)$, is given by

$$g(\eta) = \mathbb{E}[T(y);\eta].$$

In 1(a), we found that $T(y) = y$, $\eta = \log(\lambda)$, and $a(\eta) = \lambda = e^\eta$ for the given Poisson distribution parameterized by λ. Hence, we use the fact that a Poisson random variable with parameter λ has mean λ to say that

$$g(\eta) = \mathbb{E}[y;\eta] = \lambda = e^\eta.$$

Likewise, from the lecture material we know that inference is easy for exponential family models via

$$\begin{aligned} g(\eta) = \mathbb{E}[y,\eta] &= \frac{\partial}{\partial\eta}a(\eta) \\ &= e^\eta. \end{aligned}$$

1.c

The log-likelihood of an example $(x^{(i)}, y^{(i)})$ is defined as $\ell(\theta) = \log p(y^{(i)}|x^{(i)}; \theta)$. To derive the stochastic gradient ascent rule, use the results in part (a) and the standard GLM assumption that $\eta = \theta^T x$.

$$\begin{aligned}
\frac{\partial \ell(\theta)}{\partial \theta_j} &= \frac{\partial \log p(y^{(i)}|x^{(i)}; \theta)}{\partial \theta_j} \\
&= \frac{\partial \log \left(\frac{1}{y^{(i)}!} \exp(\eta^T y^{(i)} - e^{\eta}) \right)}{\partial \theta_j} \\
&= \frac{y!}{\exp\left(\eta^T y^{(i)} - e^{\eta}\right)} \cdot \frac{(x_j^{(i)} y^{(i)} - x_j^{(i)} e^{\eta}) \exp\left(\eta^T y^{(i)} - e^{\eta}\right)}{y!} \\
&= x_j^{(i)} (y^{(i)} - e^{\eta}).
\end{aligned}$$

Thus the stochastic gradient ascent update rule should be:

$$\theta_j := \theta_j + \alpha \frac{\partial \ell(\theta)}{\partial \theta_j},$$

which reduces here to:

$$\theta_j := \theta_j + \alpha x_j^{(i)} (y^{(i)} - e^{\theta^T x^{(i)}}).$$

2.a

Yes. $K(x,z) = K_1(x,z) + K_2(x,z)$ is a valid kernel.

Proof: By Mercer's Thm, we have that $K_1(x,z)$ and $K_2(x,z)$ are PSD. That is, for all $x \in \mathbb{R}$ we must have $x^T K_1 x \geq 0$ and $x^T K_2 x \geq 0$. By the properties of the real numbers then,

$$x^T K_1 x + x^T K_2 x \geq 0$$
$$x^T (K_1 + K_2) x \geq 0.$$

Hence, $K_1(x,z) + K_2(x,z)$ is also PSD by definition.

2.b

No. $K(x,z) = K_1(x,z) - K_2(x,z)$ is not necessarily a valid kernel.

As a counterexample, take the pseudo-trivial case where $K_1(x,z) = \overrightarrow{0}$ and $K_2(x,z) = \overrightarrow{1}$. These are both PSD because their traces are greater than or equal to zero. However, $K(x,z) = K_1(x,z) - K_2(x,z)$ is clearly not PSD because the trace would be negative. Since $K(x,z)$ is not PSD, it is not a valid kernel according to Mercer's Thm.

2.c

Yes. $K(x,z) = aK_1(x,z)$ for $a \in \mathbb{R}^+$ is a valid kernel.

Proof: In similar fashion as 2(a), we have that $K_1(x,z)$ is PSD by Mercer's Thm such that for all $x \in \mathbb{R}$ we must have $x^T K_1 x \geq 0$. It follows that we can distribute the scalar a to get $x^T aKx \geq 0$ since $a > 0$. $K(x,z)$ is therefore also PSD and therefore a valid kernel.

2.d

No. $K(x,z) = -aK_1(x,z)$ for $a \in \mathbb{R}^+$ is not a valid kernel.

As a counterexample, take any kernel for $K_1(x,z)$ other than the trivial solution, $K_1(x,z) = \overrightarrow{0}$. It follows that the trace of $-aK_1(x,z)$ would be negative and $K(x,z)$ would not be PSD or a valid kernel.

3.ai

We represent $\theta^{(i)}$ as a linear combination of the training examples when using the "kernel trick". In the high-dimensional space, this takes the form

$$\theta^{(i)} = \sum_{k=1}^{i} \theta_k \phi(x^{(k)})$$

where $\phi(x^{(k)})$ is the implicit representation of the training example $x^{(k)}$ and θ_k are the parameter coefficients. Coding the problem, we will represent $\theta^{(i)}$ as a list. Hence, $\theta^{(0)}$ will be defined as an empty list, [ ], given the lack of training examples.

3.aii

The perceptron algorithm is based on the sign of the dot product of $\theta^T \phi(x)$. Using our notation from 3(a.i) above, this take the form

$$\begin{aligned}
\theta^T \phi(x) &= \left(\sum_{k=1}^{i} \theta_k \phi(x^{(k)}) \right)^T \phi(x) \\
&= \sum_{k=1}^{i} \theta_k (\phi(x^{(k)})^T \phi(x)) \\
&= \sum_{k=1}^{i} \theta_k K(x^{(k)}, x)
\end{aligned}$$

where $K(x^{(k)}, x)$ is the kernel function corresponding to the high-dimensional space.

For a new input, $x^{(i+1)}$, we use the kernel trick to update the prediction via

$$\begin{aligned}
h_{\theta^{(i)}}(x^{(i+1)}) &= g({\theta^{(i)}}^T \phi(x^{(i+1)})) \\
&= g\left(\sum_{k=1}^{i} \theta_k K(x^{(k)}, x^{(i+1)}) \right).
\end{aligned}$$

3.aiii

Using the new update rule corresponding to the feature mapping,

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

we update the coefficients θ_k. This updates the parameter vector in the high-dimension space without explicitly computing it. In our code, we simply append using the prediction found in 3(a.ii) to get

$$\theta^{(i+1)} := \theta^{(i)} + \alpha \left(y^{(i+1)} - g\left(\sum_{k=1}^{i} \theta_k K(x^{(k)}, x^{(i+1)}) \right) \right) \phi(x^{(i+1)})$$

3.c

The dot-product kernel performs extremely poorly in classifying the points. This is largely due to the nature of the dataset. We observe that one class of data is nested within the other, so when the dot-product kernel attempts to identify a linear decision boundary, it fails to capture the nonlinearity of the cluster. The radial basis function kernel performs significantly better because it is suited to the quasi-circular clustering of the interior dataset.

Documentation: I did not collaborate with anyone on this assignment.