1a.

The exponential family is a class of distributions with the following form: $p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$

By identifying the parameters of the Poisson distribution:

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

$$\exp\left[\log\left[e^{-\lambda}\lambda^y \frac{1}{y!}\right]\right]$$

$$\exp\left[-\lambda + y\log(y) + \log\frac{1}{y!}\right]$$

$$\frac{1}{y!}\exp\left[y\log(y) - \lambda\right]$$

So we can see: $\eta = \log(\lambda)$; $\lambda = e^{\eta}$; $b(y) = \frac{1}{y!}$; $T(y) = y$;

$a(\eta) = e^{\eta}$.

1. b

The canonical response function for the Poisson distribution is given by:

$$g(\eta) = E(y;\eta) = \lambda = e^{\eta}$$

This result can also be obtained by the ~~use~~ use of the exponential family:

$$g(x) = \frac{\partial}{\partial \eta} a(\eta) = \frac{\partial}{\partial \eta} e^{\eta} = e^{\eta}$$

1.c

The log-likelihood of a training example $(x^{(i)}, y^{(i)})$ is given by $\log p(y^{(i)} | x^{(i)}; \theta)$.

To derive the stochastic gradient descent update rule, we can start by computing the partial derivative of the log-likelihood with respect to parameter θ_j, with $\eta = \theta^T x^{(i)}$:

$$\frac{\partial}{\partial \theta_j} \log p(y^{(i)} | x^{(i)}; \theta) = \frac{\partial}{\partial \theta_j} \left((\theta^T x^{(i)})^T y^{(i)} - e^{\theta^T x^{(i)}} \right)$$

$$= (y^{(i)} - e^{\theta^T x^{(i)}}) x_j^i$$

The yields the update rule for stochastic gradient descent with learning rate α:

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

2a.

$$k(x,z) = k_1(x,z) + k(x,z)$$

is a kernel because the sum of two PSD matrices is PSD.

2b.

$$k(x,z) = k_1(x,z) - k(x,z)$$

is not necessarily a kernel. Counterexample:

$k_1(x,z) = 0$ and $k_2(x,z) = 1 \; \forall x, z$ are kernels, but $k(x,z) = -1$ is obviously not.

2.c

$$k(x,z) = a k_1(x,z)$$

is a kernel, because the set of PSD matrices is closed under multiplication with positive real numbers

2.d.

$$k(x,z) = -ak_1(x,z)$$

would only be a kernel if:

$$a \leq 0 \text{ or } k_1 = 0.$$

3. ai

The algorithm only makes one pass through the training set, it will always be the case that $\theta^{(i)}$ is a linear combination of $\phi(x^{(1)}), \ldots, \phi(x^{(i)})$ i.e.

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

for some real numbers $\alpha_k \in \mathbb{R}$.

Thus we could represent $\theta^{(i)}$ by the list $[\alpha_1, \ldots, \alpha_i]$. For technical reason we will represent $\theta^{(i)}$ as pair $([\alpha_1, \ldots, \alpha_i], [x^{(1)}, \ldots, x^{(i)}])$.

In particular $\theta^{(0)} = 0$ will be represented by $([\,], [\,])$, i.e pair

of two empty lists.

3. a.ii

For any vector $x \in \mathbb{R}^n$ and any vector θ with

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

we've got:

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

$$= \sum_{k=1}^{i} \alpha_k K(x^{(k)}, x).$$

For a new input x^{i+1} we can compute the prediction

efficiently by

$$h_{\theta^{(i)}}(x^{(i+1)}) = g(\theta^{(i)T} \phi(x^{(i+1)}))$$

$$= g\left(\sum_{k=1}^{i} \theta_k^{(i)} K(x^{(k)}, x^{(i+1)})\right).$$

3. a) iii)

The new update rule

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

can be derived by appending

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

resp. $x^{(i+1)}$ to the first resp. second component of our representation $([\theta_1, \ldots, \theta_i], [x^{(1)}, \ldots, x^{(i)}])$ of $\theta^{(i)}$.

3.C.

With the dot product kernel the perceptron performs poorly because the model attempts to identify a linear decision boundary but the dataset is far from linearly separable.