**Problem 1.**

**(a)**

Let's assume that the observation set uses the following notation:

$$X_{N\times p} = (x_1, x_2, \cdots, x_N)^T, x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$$

This notation indicates that there are $N$ samples, and each sample is a $p$-dimensional vector. Each observation is generated by $p(x|\theta)$.

For discriminative model, it assumes that The $\theta$ in $p(x|\theta)$ is a constant. For $N$ observations, the probability of the observation set is $p(X|\theta) \underset{iid}{=} \prod_{i=1}^{N} p(x_i|\theta))$. In order to find the size of $\theta$, we use the maximum log-likelihood MLE method:

$$\theta_{MLE} = \underset{\theta}{argmax} \log p(X|\theta) \underset{iid}{=} \underset{\theta}{argmax} \sum_{i=1}^{N} \log p(x_i|\theta)$$

For generative model, it assumes The $\theta$ in $p(x|\theta)$ is not a constant. This $\theta$ satisfies a preset prior distribution $\theta \sim p(\theta)$. So according to Bayes' theorem, the posterior dependent on the parameters of the observation set can be written as:

$$p(\theta|X) = \frac{p(X|\theta) \cdot p(\theta))}{p(X)} = \frac{p(X|\theta) \cdot p(\theta)}{\int\limits_{\theta} p(X|\theta) \cdot p(\theta) d\theta}$$

In order to find the value of $\theta$, we have to maximize this parameter posterior MAP:

$$\theta_{MAP} = \underset{\theta}{argmax}\, p(\theta|X) = \underset{\theta}{argmax}\, p(X|\theta) \cdot p(\theta)$$

In general, generative model learns the joint probability distribution p(X,$\theta$) and a discriminative model learns the conditional probability distribution p(X|$\theta$) - which is the probability of X given $\theta$.

**(b)**

I would choose Logistic regression.

The requirements of the generative model are relatively high, and the joint distribution of the input data x and the category needs to be learned. For the person who only has limited knowledge of the process generating the data, the Naive Bayes would be hard to compute. The generative model needs to

calculate the class conditional probability and the prior probability, which increases the amount of calculation. Furthermore, the generative model has higher bias,because it focuses on a smaller set of models

On the contrary, the discriminant model is to solve directly, learn the optimal classification surface between different categories, and reflect the differences between different types of data. The discriminant model focuses on a bigger set of models and has lower bias.

**Problem 2.**

**(1)**

False

**(2)**

True

**(3)**

False

**Problem 3.**

**(a)**

1.For gradient descent, we compute the full gradient. But for stochastic gradient descent, we compute the random gradient. This means that GD is O(n) slower than SGD.

2.Stochastic gradient descent is memory efficient, extended to mini-batches.

**(b)**

Total runtime of gradient descent: $O(\frac{n}{\epsilon}) + O(\frac{n}{\epsilon}) = O(\frac{n}{\epsilon})$

Total runtime of stochastic gradient descent: $O(\frac{1}{\epsilon^2}) + O(\frac{1}{\epsilon^2}) = O(\frac{1}{\epsilon^2})$

**(c)**

Total runtime of sub-gradient descent: $O(\frac{n}{\epsilon}) + O(\frac{n}{\epsilon^2}) = O(\frac{n}{\epsilon^2})$

Total runtime of stochastic sub-gradient descent: $O(\frac{1}{\epsilon^2}) + O(\frac{1}{\epsilon^2}) = O(\frac{1}{\epsilon^2})$

**Problem 4.**

**(a)**

For perceptron, the prediction on $x_i$ is incorrect if $y_i w^T x_i < 0$. Let the I(w) be the set of points on which prediction is incorrect. The objective function would be:

$E(w) = -\sum_{i \in I(w)} y_i w^T x_i = \sum_{i=1}^{n} max(0, -y_i w^T x_i)$

The objective function of perceptron is hinge loss, which has positive value when the data point is misclassified and 0 when the data point is correctly classified. So there exists a hinge point that makes the hinge loss non-smooth.

**(b)**

For this question, $\eta = 1$, $w_0 = 0$. Assuming we have n data points and let the I(w) be the set of points on which prediction is incorrect.

For perceptron, if prediction is correct, w does not change.

If prediction is wrong:

$w_{new} = w_{old} - \eta \bigtriangledown E_i(w) = w_{old} + \eta y_i x_i$

For the first round, n iterations of running perceptron:

$w_n = w_0 + \sum_{i \in I(w)} y_i x_i$

$= w_0 + \sum_{i=1}^{n} n_i y_i x_i$

$= \sum_{i=1}^{n} n y_i x_i$, where $n = 1\{i \in I(w)\}$

If we keep running the perceptron until it converges, $w_n$ would be updated every time the point $(x_i, y_i)$ is misclassified and the corresponding number of mistakes made on $(x_i, y_i)$ would increase each time. Since the data points are independent and identically distributed:

$w^* = \sum_{i=1}^{n} n_i y_i x_i$, where $n_i$ is the number of mistakes made on $(x_i, y_i)$.

Because we let the final vector after convergence $w^* = \sum_{i=1}^{n} \alpha_i y_i x_i$, $\alpha_i = n_i$ = the number of mistakes made on $(x_i, y_i)$ during the perceptron iterations before convergence.

**(c)**

Linear separability means that a linear function can be used to separate two classes without error, such as a straight line in two-dimensional space, a plane in three-dimensional space, and a linear function in high-dimensional space. Linear inseparable refers to the situation where some samples are divided by linear classification planes that will produce classification errors.

For the classes are not linearly separable, there must exist a data point $(x_i, y_i)$ make $y_i w^T x_i < 0$ for all w. Thus, the perceptron with a fixed learning rate $\eta$ cannot converge.

I cannot modify the learning rate $\eta$ so that the iterations converge in expectation. Once the iterations can converge, it means we can find a w that can clearly separate the two classes for all data points, which contradict the assumption that the classes are not linearly separable.

**Extra Credit 1:**

$\theta^* = argmax_\theta P(x_1, x_2, \cdots, x_n|\theta)$

$= argmax_\theta \Pi_{i=1}^n P(x_i|\theta)$ (i.i.d. samples)

$= argmax_\theta \sum_{i=1}^n ln(P(x_i|\theta))$ (take the logarithm)

$= argmax_\theta \sum_{i=1}^n ln(2\theta x_i e^{-\theta x_i^2})$

$= argmax_\theta \sum_{i=1}^n ln(2\theta x_i) + ln(e^{-\theta x_i^2})$

$= argmax_\theta \sum_{i=1}^n ln(2\theta x_i) - \theta x_i^2$

$= argmin_\theta \sum_{i=1}^n \theta x_i^2 - ln(2\theta x_i)$

To get the optimal $\theta$, we take the derivative of $\theta$ for the above equation and let the derivative equals to 0.

$\frac{\partial(\sum_{i=1}^n \theta x_i^2 - ln(2\theta x_i))}{\partial\theta} = 0$

$\frac{\partial(\sum_{i=1}^n \theta x_i^2)}{\partial\theta}$ - $\frac{\partial(\sum_{i=1}^n ln(2\theta x_i)}{\partial\theta} = 0$

$(\sum_{i=1}^n x_i^2) - (\sum_{i=1}^n \frac{2x_i}{2\theta x_i}) = 0$

$(\sum_{i=1}^n x_i^2) - (\sum_{i=1}^n \frac{1}{\theta}) = 0$

$(\sum_{i=1}^n x_i^2) = \frac{n}{\theta}$

$\theta = \frac{n}{(\sum_{i=1}^n x_i^2)}$

**Extra Credit 2:**

$\frac{\partial L(y_i|x_i,w)}{\partial w_j} = \frac{\partial y_i log(\frac{1}{1+e^{-w^T x_i}}) + (1-y_i) log(\frac{1}{1+e^{w^T x_i}})}{\partial w_j}$

For the first part:

$\frac{\partial log(\frac{1}{1+e^{-w^T x_i}})}{\partial w_j}$

$= -\frac{\partial log(1+e^{-w^T x_i})}{\partial w_j}$ using $log(a^b) = blog(a)$

Now, using the chain rule, set $1 + e^{-w^T x_i} = y$

$= -\frac{\partial log y}{\partial y} \frac{\partial y}{\partial w_j}$

$= -\frac{1}{y} \frac{\partial y}{\partial w_j}$

Now, for $\frac{\partial 1+e^{-w^T x_i}}{\partial w_j}$, we do not need to consider the constant term. So,using the chain rule again, set $-w^T x_i = y$

$= -\frac{1}{y} \frac{\partial e^y}{\partial y} \frac{\partial y}{\partial w_j}$

$= -\frac{1}{y} * (e^{-w^T x_i} * (-x_{ij}))$

$= \frac{e^{-w^T x_i} * x_{ij}}{1+e^{-w^T x_i}}$

$= (1 - \frac{1}{1+e^{-w^T x_i}}) x_{ij}$

For the second part:

$\frac{\partial log(\frac{1}{1+e^{w^T x_i}})}{\partial w_j}$

$= -\frac{\partial log(1+e^{w^T x_i})}{\partial w_j}$ using $log(a^b) = blog(a)$

Now, using the chain rule, set $1 + e^{w^T x_i} = y$

$= -\frac{\partial log y}{\partial y} \frac{\partial y}{\partial w_j}$

$= -\frac{1}{y} \frac{\partial y}{\partial w_j}$

Now, for $\frac{\partial 1+e^{w^T x_i}}{\partial w_j}$, we do not need to consider the constant term. So,using the chain rule again, set $w^T x_i = y$

$= -\frac{1}{y}\frac{\partial e^y}{\partial y}\frac{\partial y}{\partial w_j}$

$= -\frac{1}{y} * (e^{w^T x_i} * (x_{ij}))$

$=-\frac{e^{w^T x_i} * x_{ij}}{1+e^{w^T x_i}}$

$=-\frac{x_{ij}}{1+e^{-w^T x_i}}$

The original formula:

$=y_i((1-\frac{1}{1+e^{-w^T x_i}})x_{ij}) + (1-y_i)(-\frac{x_{ij}}{1+e^{-w^T x_i}})$

$=(y_i - \frac{y_i}{1+e^{-w^T x_i}} - \frac{1}{1+e^{-w^T x_i}} + \frac{y_i}{1+e^{-w^T x_i}})x_{ij}$

$=(y_i - \frac{1}{1+e^{-w^T x_i}})x_{ij}$

$=(y_i - \sigma(w^T x_i))x_{ij}$

**Extra Credit 3:**

Suppose K is a gram matrix which $K_{ij} = K(x_i, x_j) \in R^{n \times n}$.

For any vector $\alpha \in R^n$:

$\alpha^T K \alpha$

$= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j (K(x_i, x_j))$

$= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j)$

$\because K(x, y) = min\{x, y\}$, x and y are positive numbers.

$\therefore \forall K(x_i, x_j)$, it is a positive number. Let's assume it is $m_i j$.

$= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \Phi(\sqrt{m_i j})^T \Phi(\sqrt{m_i j})$

$\because \alpha_i$ and $\alpha_j$ are real numbers

$=\sum_{i=1}^n \alpha_i \Phi(\sqrt{m_i j})^T \sum_{j=1}^n \alpha_j \Phi(\sqrt{m_i j})$

$= \sum_{k=1}^m \{\sum_{i=1}^n \alpha_i \Phi(\sqrt{m_i j})\}^T \sum_{j=1}^n \alpha_j \Phi(\sqrt{m_i j})$

$\because$ inner product always $\geq 0$

$\therefore \alpha^T K \alpha \geq 0$ and K is positive semi-definite function.

For symmetry:

$K(x, y)^T$

$= (min\{x, y\})^T$

$= (min\{x, y\})$

$= K(x, y)$

K(x,y) = $min\{x, y\}$ is a valid kernel function because it is symmetric and positive semi-definite function.