# CS229 Final Exam - Summer 2019

Aug 16 2019 3:30pm PDT - Aug 17 2019 3:30pm PDT

## [0 points] The Stanford University Honor Code

At the top of your solution file, please write/type the following Honor Code statement and attest it (i.e. sign or print your name below it), thereby implying your consent to follow the code:

"*I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.*"

**Submissions without the attested Honor Code statement will NOT be graded.**

## 1 [10 points] True or False

Each question is for 1 point. To earn 1 point per question, provide the correct True or False answer along with the correct justification. Incorrect answers or incorrect justifications will earn 0 points. Justifications must be short (a couple of sentences). There will be no negative points.

1. **True or False**. Overfitting occurs when models are "too complex" for a given situation. One way to prevent it is by adding a regularization penalty. Your friend proposes an alternative: to train the model on fewer training data which is then less likely to overfit. Your friend is correct.

   **Answer:**

   Wrong.

   A specific classifier (with a fixed model complexity) will be more likely to overfit to noise in the training data when there is less training data, and is therefore more likely to overfit.

2. **True or False**. Given a model $y = \log\left(x_1^{\theta_1} x_2^{3\theta_2}\right) x_3^5 + \epsilon$, where $x_1, x_2 \in \mathbb{R}_+$, $y, x_3, \theta_1, \theta_2 \in \mathbb{R}$, $\epsilon \sim N(0,1)$ i.i.d., the Maximum Likelihood value of the model parameters ($\theta$) can be estimated with linear regression.

   **Answer:** True.

   Since $y$ is linear in $\theta_1$ and $\theta_2$, it can be learnt using linear regression.

$$\begin{aligned} y_i &= \log(x_1^{\theta_1} x_2^{3\theta_2}) x_3^5 + \epsilon_i \\ &= (\theta_1 \log(x_1) + 3\theta_2 \log(x_2)) x_3^5 + \epsilon_i \\ &= \theta_1 x_3^5 \log(x_1) + 3\theta_2 x_3^5 \log(x_2) + \epsilon_i \end{aligned}$$

3. **True or False**. In a binary classification problem, let's consider an example that is *correctly* classified and sufficiently far from the decision boundary. Claim: Logistic regression's decision boundary will be unaffected by this point but the one learnt by SVM will be affected.

**Answer:** False.

It is the reverse in fact. The hinge loss used by SVMs gives zero weight to these points while the log-loss used by logistic regression gives a little bit of weight to these points.

4. **True or False**. Consider a Bernoulli distribution with parameter $\theta \in [0, 1]$. The Maximum Likelihood estimate (MLE) of the parameter $\theta$ is equal to the Maximum-a-posteriori (MAP) estimate when the prior probability $p(\theta)$ is the Uniform distribution over [0,1].

   **Answer:** True.

$$\begin{aligned}\theta_{MLE} &= \arg\max_{\theta} \sum_i \log P(x_i|\theta) \\ \theta_{MAP} &= \arg\max_{\theta} \log P(X|\theta) + \log P(\theta) \\ &= \arg\max_{\theta} \left\{ \sum_i \log P(x_i|\theta) + \log P(\theta) \right\}\end{aligned}$$

   When prior is uniform, $\log P(\theta)$ is constant, which does not affect arg max. Hence true.

5. **True or False**. The more features that we use to represent our data, the better the learning algorithm will generalize to new data points.

   **Answer:** False (or not necessary).

   Adding more and more features is not always a good idea. It increases the dimension of the search space and thus makes the problem harder, the increased complexity outweighing value from the additional features.

6. **True or False**. Like the Logistic regression algorithm, the Perceptron algorithm does not converge if the training examples are perfectly separable.

   **Answer:** False.

   The hypothesis function of the perceptron algorithm is a threshold function:

$$\begin{aligned}h_\theta(x) &= g(\theta^T x) \\ &= \begin{cases} 1 \text{ if } \theta^T x \geq 0 \\ 0 \text{ if } \theta^T x < 0 \end{cases}\end{aligned}$$

   The update rule of the perceptron algorithm adjusts the parameter $\theta$ when there is misclassified point.

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

   If the model cannot classify all the training data correctly, it will never reach a stable point. On the other hand, it is guaranteed to converge if the two classes are linearly separable.

   Partial credit was given to those who cited that the scaling logic for logistic regression does not apply with the perceptron (but fail to justify why, for instance, there can't be a different 'improvement' of the perceptron.)

   No credit was given for reasoning that would also apply to logistic regression.

7. **True or False.** Given an arbitrary Neural Network where we use Stochastic Gradient Descent to optimize the parameters, initializing all the weights to 0 is a good practice.

   **Answer:** False.

   If all weights are set to 0, their gradients will be the same, thus all weights in a given layer will have the same value in all the subsequent iterations.

Partial credit was given to those who said there will be zero learning, which is true only for certain (e.g., ReLU) activations.

8. **True or False.** The Exponential Family of probability distributions are obtained as a natural consequence of the Maximum Entropy principle. The entropy of any exponential family distribution is therefore $+\infty$.

   **Answer:** False. For instance, the Gaussian distribution has an entropy of the Gaussian distribution is $1/2 \ln(2\pi e \sigma^2)$

9. **True or False.** The Kernel Trick enables us to restructure learning algorithms and make them scalable to very large number of training examples, and with certain kernels even to an infinite number of examples.

   **Answer:** False .

10. **True or False.** Gaussian Processes enjoy a closed form solution once the kernel matrix has been evaluated. Alternatively, they can also be trained with Gradient Ascent/Descent (or Stochastic Gradient Ascent/Descent).

    **Answer:** Note: the first sentence is just a (true) statement; student were supposed to evluate the second statement.

    False. GPs are a Bayesian approach where we do not minimize a loss function during training. We condition on the observed data and construct a posterior distribution.

## 2 [10 points] Short Answers

Each sub-question is worth 2 points. Provide a short justification (one or two lines) for each answer.

1. **Short Answer**. How many parameters are necessary to specify a Naive Bayes classifier under the *Bernoulli Event Model* assumption where the size of the vocabulary is $d$ words, and there are **three classes**?

   **Answer:** $3d + 2$.

   We need 2 parameters for $p(y)$, and 3 parameters for each $x_i$, $\phi_{i|y=j} = p(x_i = 1|y = j), j \in \{0, 1, 2\}$.

2. **Short Answer**. How many parameters are necessary to specify a Naive Bayes classifier with the *Multinomial Event Model* assumption where the size of the vocabulary is $d$ words, and there are **three classes**?

   **Answer:** $3d - 1$.

   We need 2 parameter for $p(y)$, and $d - 1$ for each of the three Multinomial distributions.

3. **Short Answer**. Consider a Markov Decision Process (MDP) is defined as $(S, A, \{P_{sa}\}, \gamma, R)$, where $S$ is a finite set of states of size $|S|$, $A$ is a finite set of actions of size $|A|$, $\{P_{sa}\}$ are the state transition probabilities, $\gamma \in [0, 1)$ is the discount factor, and $R$ is the reward function. How many distinct policies $\pi : S \to A$ can possibly be defined for this MDP?

   **Answer:** $|A|^{|S|}$.

4. **Short Answer**. In Reinforcement Learning, unlike the Value Iteration algorithm, the Fitted Value Iteration algorithm does not have a guarantee of convergence to the true optimal value function $V^*$. This could be because $V^*$ may not be representable by our model class. Would you consider this as a High Bias or High Variance scenario?

   **Answer:** High Bias.

5. **Short Answer**. In the EM algorithm, we are given a training set $S = \{x^{(i)}\}_{i=1}^n$ that corresponds to a model $p(x, z; \theta)$. We do not observe the corresponding $z^{(i)}$'s, which are also called the latent variables. Our goal is to maximize $\sum_{i=1}^n \log p(x^{(i)}; \theta)$ w.r.t $\theta$.

   Suppose our dataset is not too large (i.e $n$ is not very big), and out of concern of overfitting, we decide to add regularization in the M-step, resulting in the following modified algorithm:

   - Repeat until convergence:
     - E-Step. For each $i = 1, \ldots, n$, set

       $$Q_i^{(t)}(z^{(i)}) = p(z^{(i)}|x^{(i)}; \theta^{(t)})$$

     - M-Step. Update the parameters with

       $$\theta^{(t+1)} = \arg\max_\theta \sum_{i=1}^n \text{ELBO}(x^{(i)}; Q_i^{(t)}, \theta) + \lambda\|\theta\|_2^2$$

   Is the above algorithm guaranteed to monotonically increase $\sum_{i=1}^n \log p(x^{(i)}; \theta)$ with each iteration? Provide an informal justification.

   **Answer:** No.

   Monotonic increase of likelihood requires $\text{ELBO}(Q^{(t)}, \theta^{(t+1)}) \geq \text{ELBO}(Q^{(t)}, \theta^{(t)})$ which is guaranteed by the original M-step. The above M-step does not provide that guarantee. The above M-step only guarantees that $\text{ELBO}(Q^{(t)}, \theta^{(t+1)}) + \lambda\|\theta^{(t+1)}\|_2^2 \geq \text{ELBO}(Q^{(t)}, \theta^{(t)}) + \lambda\|\theta^{(t)}\|_2^2$ which is insufficient.

   Remark: However, it can be shown that the above algorithm will *converge* even though the sequence of likelihood values does not monotonically increase.

# 3 [10 points] Adding features to Linear regression

In our study of Bias and Variance, we have seen that adding more features (and hence parameters) to a model can increase the variance. One way to think about variance is as the difference between test error and training error. So, generally, increasing the variance tends to reduce training error (and/or increase test error). In the case of linear regression, we can actually show that adding a new feature (it could be anything, even random numbers!) will always reduce the training error (or keep it the same).

Let $X \in \mathbb{R}^{n \times d}$ be a design matrix, and $\vec{y} \in \mathbb{R}^n$ the labels associated with it. Let $\theta \in \mathbb{R}^d$ be the parameters of the linear regression model. Now we will add a new random column $\vec{r} \in \mathbb{R}^n$ to the design matrix, resulting in an updated design matrix

$$\tilde{X} \in \mathbb{R}^{n \times (d+1)} = [X, \vec{r}\,].$$

Note that this new feature is not a random variable, but just a pre-computed vector of random numbers, that are unrelated to $X$. Correspondingly, we add a new parameter $t \in \mathbb{R}$ to the model, resulting in an updated parameter vector

$$\Theta \in \mathbb{R}^{d+1} = \begin{bmatrix} \theta \\ t \end{bmatrix}.$$

Let us define the projection and residual matrices of $X$ (assuming $X$ is full-rank, so $X^T X$ is invertible) as follows:

$$P = X(X^T X)^{-1} X^T,$$
$$R = P - I,$$

where for example for any vector $\vec{v} \in \mathbb{R}^n$, $P\vec{v}$ is the projection of the vector $\vec{v}$ onto the range of $X$, $I$ is the Identity matrix, and $R\vec{v}$ is the residual $P\vec{v} - \vec{v}$, i.e. the projection minus the original vector.

Recall that the cost function for linear regression is the squared error. Let us denote the cost functions of the two models as

$$J(\theta) = ||X\theta - \vec{y}||_2^2$$
$$\tilde{J}(\Theta) = ||\tilde{X}\Theta - \vec{y}||_2^2$$

We basically need to show that

$$\min_{\Theta} \tilde{J}(\Theta) = \tilde{J}^* \leq J^* = \min_{\theta} J(\theta)$$

We will break down the task into sub-problems and solve them in steps. The overall strategy will be to minimize $\tilde{J}(\Theta)$ in two stages, first with respect to a subset of its parameters $\theta$, and next with respect to the remaining parameter $t$. This will allow us to show that its minima is smaller than the minima of $J(\theta)$. i.e., across the following sub-questions, we will show

$$\min_{t} \left( \min_{\theta} \tilde{J}(\Theta) \right) = J^* - (\cdot)^2,$$

which will complete the proof.

1. **[2 points] Minimizer of $J(\theta)$**

   Calculate $J^*$, the lowest value achieved by $J(\theta)$ in the form of a closed form expression. Your expression can involve only $R$ and $\vec{y}$ terms.

   **Answer:**

   We know that the lowest value of $J(\theta)$ is achieved at $\theta = (X^T X)^{-1} X^T \vec{y}$. Plugging this into the definition of $J(\theta)$, we get

$$
\begin{aligned}
J^* &= \|X(X^TX)^{-1}X^T\vec{y} - \vec{y}\|_2^2 \\
&= \|P\vec{y} - \vec{y}\|_2^2 \\
&\Rightarrow \boxed{J^* = \|R\vec{y}\|_2^2}
\end{aligned}
$$

2. **[2 points] Estimate $\theta$**

   We will now estimate parameters (closed form) that minimize the updated loss $\tilde{J}(\Theta)$. We will do that by separately estimating $\hat{\theta}$ and $\hat{t}$. In this sub-problem we will estimate $\hat{\theta}$. Calculate $\hat{\theta}$ by solving for $\theta$ in the equation $\nabla_\theta \tilde{J}(\Theta) = 0$. You may refer to lecture notes and cite the appropriate filename and page number to shorten your calculations. Your expression for $\hat{\theta}$ can only include $X$, $\vec{y}$, $\vec{r}$ and $t$ terms.

   **Answer:**

$$
\begin{aligned}
\nabla_\theta \tilde{J}(\Theta) &= \nabla_\theta \|\tilde{X}\Theta - \vec{y}\|_2^2 \\
&= \nabla_\theta \|X\theta + (\vec{r}t - \vec{y})\|_2^2 \\
&= 2X^TX\theta - 2X^T(\vec{y} - \vec{r}t) = 0 \\
&\Rightarrow \boxed{\hat{\theta} = (X^TX)^{-1}X^T(\vec{y} - \vec{r}t)}
\end{aligned}
$$

3. **[2 points] Updated $\tilde{J}$**

   Plug the estimated value of $\hat{\theta}$ into $\tilde{J}(\Theta)$, and call it $\tilde{J}_{\hat{\theta}}(t)$. This is because once we minimize $\tilde{J}(\Theta)$ with respect to $\theta$, we will be left with a function of only $t$. Now express $\tilde{J}_{\hat{\theta}}(t)$ as $J^* + f(t, \vec{r}, R, \vec{y})$. Clearly state the expression for $f$.

   **Answer:**

$$
\begin{aligned}
\tilde{J}_{\hat{\theta}}(t) &= ||\tilde{X}\Theta - \vec{y}||_2^2 \\
&= ||X\hat{\theta} + \vec{r}t - \vec{y}||_2^2 \\
&= ||P(\vec{y} - \vec{r}t) + \vec{r}t - \vec{y}||_2^2 \\
&= ||R\vec{y} - tR\vec{r}||_2^2 \\
&= \|R\vec{y}\|_2^2 + t^2\|R\vec{r}\|_2^2 - 2t\langle R\vec{y}, R\vec{r}\rangle \\
&= J^* + f(t, \vec{r}, R, \vec{y})
\end{aligned}
$$

   where $\boxed{f(t, \vec{r}, R, \vec{y}) = t^2\|R\vec{r}\|_2^2 - 2t\langle R\vec{y}, R\vec{r}\rangle}$

4. **[2 points] Estimate $t$**

   Now we will estimate $t$ by solving for $t$ in the equation $\nabla_t \tilde{J}_{\hat{\theta}}(t) = 0$. Come up with a closed form expression for $\hat{t}$. Your expression can only involve $R$, $\vec{y}$ and $\vec{r}$ terms.

   **Answer:**

$$
\nabla_t \tilde{J}_{\hat{\theta}}(t) = \nabla_t \left[J^* + f(t, \vec{r}, P, \vec{y})\right]
$$

$$= 0 + 2t\|R\vec{r}\|_2^2 - 2\langle R\vec{y}, R\vec{r}\rangle = 0$$

$$\Rightarrow \boxed{\hat{t} = \frac{\langle R\vec{y}, R\vec{r}\rangle}{\langle R\vec{r}, R\vec{r}\rangle}}$$

5. **[2 points] Minimizer $\tilde{J}^*$**

   Now we are ready to calculate the minimizer $\tilde{J}^*$ by plugging in the estimated $\hat{t}$ into $\tilde{J}_{\hat{\theta}}(t)$. Show that $\tilde{J}^* \leq J^*$

   **Remark:** Note, what we have shown is that adding a new feature decreases train error, and we have shown nothing about the test error. It is very likely that if we add a random feature as above, the test error is only going to get worse.

   **Answer:**

$$\begin{aligned}
\tilde{J}^* &= \tilde{J}_{\hat{\theta}}(\hat{t}) \\
&= J^* + f(\hat{t}, \vec{r}, P, \vec{y}) \\
&= J^* + \hat{t}^2\|R\vec{r}\|_2^2 - 2\hat{t}\langle R\vec{y}, R\vec{r}\rangle \\
&= J^* + \left(\frac{\langle R\vec{y}, R\vec{r}\rangle}{\langle R\vec{r}, R\vec{r}\rangle}\right)^2 \|R\vec{r}\|_2^2 - 2\left(\frac{\langle R\vec{y}, R\vec{r}\rangle}{\langle R\vec{r}, R\vec{r}\rangle}\right)\langle R\vec{y}, R\vec{r}\rangle \\
&= J^* - \left(\frac{\langle R\vec{y}, R\vec{r}\rangle}{\|R\vec{r}\|_2}\right)^2 \\
&\leq J^*. \quad \blacksquare
\end{aligned}$$

# 4 [10 points] Regression with weak supervision

In this question we will derive a method to learn a regression model, i.e. predict $y \in \mathbb{R}$ given $x \in \mathbb{R}^d$, with only weak supervision. By weak supervision we mean, rather than observing a point value for $y$ of an example, we are instead given an interval $[l, u]$ such that $l \leq y \leq u$, where $u, l \in \mathbb{R}$ are the upper and lower ends of the interval. Our training set consists of $n$ 3-tuples $\{(x^{(i)}, l^{(i)}, u^{(i)})\}_{i=1}^n$ where each $x^{(i)} \in \mathbb{R}^d$ is the input, and the corresponding unknown output lies in the range $[l^{(i)}, u^{(i)}]$ where $l^{(i)}, u^{(i)} \in \mathbb{R}$. Note that the width of each interval can be different for different examples, and the true unobserved $y^{(i)}$ could lie anywhere in that interval, not necessarily at the mid-point. We are also provided a feature map $\psi : \mathbb{R}^d \to \mathbb{R}^p$ that we will use for learning.

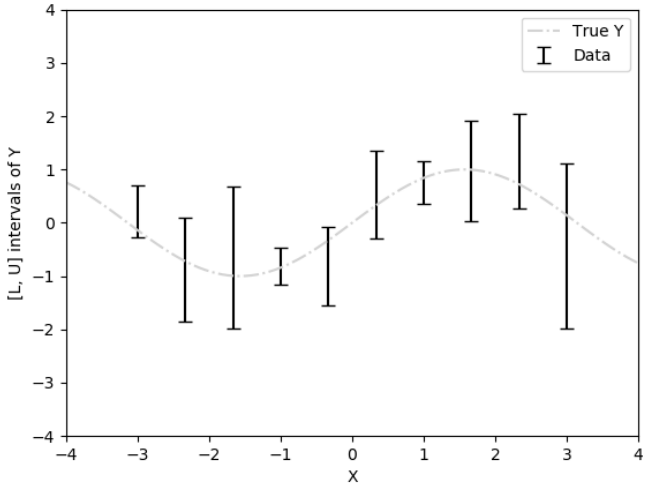



Figure 1: Training Dataset. In place of a point estimate of $y$, we instead have $[l, u]$ intervals for each example where the $y$ value resides in. The true $y$ is unknown.

We make the assumption that for each example, $y|x$ is distributed according a Normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}_+$. For simplicity we will assume $\sigma^2 = 1$ throughout this problem. We make the linearity assumption

$$\mu = \psi(x)^T \theta,$$

where $\theta \in \mathbb{R}^p$ are the learnable parameters. The predictions from our model on a new example will be

$$h_\theta(x) = \mathbb{E}[y|x; \theta].$$

In order to fit our model to the data, and recognizing the fact that the true $y$ lies in the interval $[l, u]$, we define the likelihood function of $\theta$ given the training set as:

$$\ell(\theta) = \log \prod_{i=1}^{n} P\left(l^{(i)} \leq y^{(i)} \leq u^{(i)} \mid x^{(i)}; \theta\right).$$

Based on the above likelihood objective, we will use gradient ascent to perform maximum likelihood estimation of the parameters $\theta$.

1. **[5 points]** Derive the full-batch gradient ascent update rule for the above log-likelihood objective. You may use the notation $\phi_\mu(y)$ and $\Phi_\mu(y)$ to be the probability density function and cumulative

distribution function of a Normal distribution with mean $\mu$ (and $\sigma^2 = 1$) evaluated at $y$. You may also leave the update rule in terms of these functions.

**Answer:** We start with the likelihood:

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{n} \log P(l^{(i)} \leq y^{(i)} \leq u^{(i)} \mid x^{(i)}; \theta) \\ &= \sum_{i=1}^{n} \log \left[ P(y^{(i)} \leq u^{(i)} \mid x^{(i)}; \theta) - P(y^{(i)} \leq l^{(i)} \mid x^{(i)}; \theta) \right] \\ &= \sum_{i=1}^{n} \log \left[ \Phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \Phi_{\theta^T \psi(x^{(i)})}(l^{(i)}) \right]\end{aligned}$$

From this we derive the gradient:

$$\begin{aligned}\nabla_\theta \ell(\theta) &= \sum_{i=1}^{n} \nabla_\theta \log \left[ \Phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \Phi_{\theta^T \psi(x^{(i)})}(l^{(i)}) \right] \\ &= -\sum_{i=1}^{n} \frac{\phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \phi_{\theta^T \psi(x^{(i)})}(l^{(i)})}{\Phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \Phi_{\theta^T \psi(x^{(i)})}(l^{(i)})} \cdot \psi(x^{(i)})\end{aligned}$$

And thus the gradient update rule:

$$\theta := \theta - \alpha \sum_{i=1}^{n} \frac{\phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \phi_{\theta^T \psi(x^{(i)})}(l^{(i)})}{\Phi_{\theta^T \psi(x^{(i)})}(u^{(i)}) - \Phi_{\theta^T \psi(x^{(i)})}(l^{(i)})} \cdot \psi(x^{(i)})$$

2. **[2 points]** Assuming we have found the model parameters $\hat{\theta}$ (using gradient ascent), provide an expression for the hypothesis to make a prediction on an unseen example as $h_{\hat{\theta}}(x) = \mathbb{E}[y|x; \hat{\theta}] = \ldots$

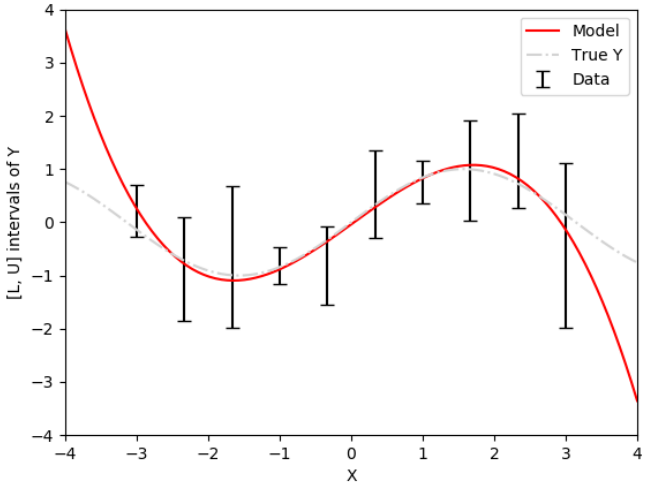
   **Answer:** Being a symmetric distribution around $\mu$, we readily observe

$$\begin{aligned}h_{\hat{\theta}}(x) &= \mathbb{E}[y|\mu] \\ &= \mu \\ &= \psi(x)^T \hat{\theta}.\end{aligned}$$

3. **[3 points]** Use the provided starter code to fill in the update rule and the prediction rule derived above. The starter code already uses a suitable feature map. Run the starter code and obtain a plot of the learned hypothesis against the training set. **Include only the obtained plot as your answer**. You do not need to submit your code.

   You may use the `norm` module from the `scipy.stats` package (already imported in the starter code), in particular, the methods `norm.cdf()` and `norm.pdf()` to evaluate the cumulative distribution and probability density function of the Standard Normal distribution.

   **Answer:**


Model
True Y
Data
[L, U] intervals of Y
X
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4