WEEK 2 STUDY GUIDE

10-301/10-601 Introduction to Machine Learning (Summer 2025)
http://www.cs.cmu.edu/~hchai2/courses/10601
Released: Monday, May 19th, 2025
Quiz Date: Friday, May 23rd, 2025
TAs: Andy, Canary, Michael, Sadrishya, and Neural the Narwhal

Summary These questions are meant to prepare you for the upcoming quiz on MLE/MAP, Logistic Regression, Linear Regression and Neural Networks. You'll start with a few questions about MLE and MAP. Then, you'll work through an example of logistic regression and consider an adversarial attack scenario. Next, you'll apply gradient descent to optimize the parameters of a linear regression model, then consider the impact of various modifications to the training dataset and a new loss function. You'll revisit logistic regression to consider the impact of regularization. Finally, you'll answer a few questions about training and tuning neural networks, followed by an example that asks you to compute all of the relevant quantities for forward and backpropagation by hand, to give you some insight into these algorithms.

Note These questions are entirely optional; you do not need to submit your answers to these questions. However, at least 50% of the questions that will appear in your workshop quiz will be *identical or nearly identical* to questions in this document. Thus, we recommend you to at least attempt every question. Furthermore, we *highly encourage* you to work in groups to solve these questions: because you are not being directly assessed on your solutions, feel free to share solutions and discuss ideas with your peers.

We encourage you to work on this study guide regularly throughout the week; in particular, this study guide is organized in sections where each section corresponds to a particular day's lectures. Here is our recommended schedule for working on this study guide:

1. MLE/MAP - after Monday's (5/19) lectures
2. Logistic Regression: Example - after Tuesday's (5/20) lectures
3. Logistic Regression: Adversarial Attack - after Tuesday's (5/20) lectures
4. Linear Regression - after Tuesday's (5/20) lectures
5. Regularization - after Wednesday's (5/21) lectures
6. Neural Networks: Warm-up - after Wednesday's (5/21) lectures
7. Neural Networks: Example - after Thursday's (5/22) lectures

1 MLE/MAP

1. (a) True. As the number of training examples approaches infinity, the influence of the prior becomes negligible.

2. 1

   The posterior $p(w|Y)$ follows

   $$\begin{aligned} p(\gamma|Y) &\propto p(Y|\gamma)p(\gamma) \\ &= 2\gamma \times 3\gamma e^{-3\gamma \times \frac{2}{3}} \\ &= 6\gamma^2 e^{-2\gamma} \end{aligned}$$

   Take the first derivative of the posterior and set it to 0, we have

   $$12\gamma e^{-2\gamma}(1-\gamma) = 0$$

   Solve for γ, we obtain $\gamma = 0$ or $\gamma = 1$. Given that the second derivative of the posterior is not always less than 0, thus, $\gamma = 0$ and $\gamma = 1$ are only stationary points, not necessarily giving the global maximum. Further notice that when $\gamma = 0$, the posterior is 0. Thus, the MAP estimate of γ is 1.

3. MLE 0.75, MAP 0.5

   MLE:
   $w = p(\text{heads})$
   $w = 0 : f(w = 0) = \binom{3}{2} w^x (1-w)^{n-x} = 3(0)^2(1)^1 = 0$
   $w = .25 : f(w = .25) = \binom{3}{2} w^x (1-w)^{n-x} = 3(.25)^2(.75)^1 = 0.140625$
   $w = .5 : f(w = .5) = \binom{3}{2} w^x (1-w)^{n-x} = 3(.5)^2(.5)^1 = 0.375$
   $w = .75 : f(w = .75) = \binom{3}{2} w^x (1-w)^{n-x} = 3(.75)^2(.25)^1 = 0.421857$
   $w = 1 : f(w = 1) = \binom{3}{2} w^x (1-w)^{n-x} = 3(1)^2(0)^1 = 0$
   $w = 0.75$ has the highest estimate so we choose that as our MLE

   MAP:
   $w = p(\text{heads})$
   $w = 0 : 0.9 * f(w = 0) = 0.9 * \binom{3}{2} w^x (1-w)^{n-x} = 0.9 * 3 * (0)^2(1)^1 = 0$
   $w = .25 : 0.04 * f(w = .25) = 0.04 * \binom{3}{2} w^x (1-w)^{n-x} = 0.04 * 3 * (.25)^2(.75)^1 = 0.005$
   $w = .5 : 0.03 * f(w = .5) = 0.03 * \binom{3}{2} w^x (1-w)^{n-x} = 0.03 * 3 * (.5)^2(.5)^1 = 0.011$
   $w = .75 : 0.02 * f(w = .75) = 0.02 * \binom{3}{2} w^x (1-w)^{n-x} = 0.02 * 3 * (.75)^2(.25)^1 = 0.008$
   $w = 1 : 0.01 * f(w = 1) = 0.01 * \binom{3}{2} w^x (1-w)^{n-x} = 0.01 * 3 * (1)^2(0)^1 = 0$
   $w = 0.5$ has the highest estimate so we choose that as our MAP

2 Logistic Regression: Example

1. 0.7975

$$\begin{aligned}
\ell(\mathbf{w}) = -\frac{1}{N}\log p\left(\mathbf{y} \mid \mathbf{X}, \mathbf{w}\right) &= -\frac{1}{N}\sum_{i=1}^{N}\log\left(p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\log\left(\sigma(\mathbf{w}^T\mathbf{x}^{(i)})^{y^{(i)}}(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})^{(1-y^{(i)})}\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\log(\sigma(\mathbf{w}^T\mathbf{x}^{(i)})) + (1-y^{(i)})\log(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})) \\
&= -\frac{1}{4}(\log(1-\sigma(1)) + \log(\sigma(2)) \\
&+ \log(\sigma(3)) + \log(1-\sigma(1.5))) \\
&= 0.797548...
\end{aligned}$$

2. 0.2044
   -0.0417
   0.1709

$$\begin{aligned}
\frac{\partial\ell(\mathbf{w})}{\partial w_j} &= \frac{1}{N}\sum_{i=1}^{N}(\sigma(\mathbf{w}^T\mathbf{x}^{(i)}) - y^{(i)})\mathbf{x}_j^{(i)} \\
\frac{\partial\ell(\mathbf{w})}{\partial w_1} &= \frac{1}{4}((\sigma(1)-0)(0) + (\sigma(2)-1)(0) + (\sigma(3)-1)(0) + (\sigma(1.5)-0)(1)) \\
&= 0.204396... = 0.2044 \\
\frac{\partial\ell(\mathbf{w})}{\partial w_2} &= \frac{1}{4}((\sigma(1)-0)(0) + (\sigma(2)-1)(1) + (\sigma(3)-1)(1) + (\sigma(1.5)-0)(0)) \\
&= -0.0416571987... = -0.0417 \\
\frac{\partial\ell(\mathbf{w})}{\partial w_3} &= \frac{1}{4}((\sigma(1)-0)(1) + (\sigma(2)-1)(0) + (\sigma(3)-1)(1) + (\sigma(1.5)-0)(0)) \\
&= 0.17090817... = 0.1709
\end{aligned}$$

3. 1.2956
   2.0417
   0.8291

$$\begin{aligned}
w_j &\leftarrow w_j - \eta\frac{\partial\ell(\mathbf{w})}{\partial w_j} \\
w_1 &= 1.5 - 1(0.2044) = 1.2956 \\
w_2 &= 2 - 1(-0.0417) = 2.0417 \\
w_3 &= 1 - 1(0.1709) = 0.8291
\end{aligned}$$

3 Logistic Regression: Adversarial Attack

1. $\frac{\partial \sigma(\mathbf{w}^T\mathbf{x})}{\partial \mathbf{x}} = \sigma(\mathbf{w}^T\mathbf{x})(1-\sigma(\mathbf{w}^T\mathbf{x}))\mathbf{w}$

   $\mathbf{x} \leftarrow \mathbf{x} + \eta\sigma(\mathbf{w}^T\mathbf{x})(1-\sigma(\mathbf{w}^T\mathbf{x}))\mathbf{w}$

2. The update rule is for each pixel j in the pixel vector x, set $x_j = 1$ if $w_j > 0$ and set $x_j = 0$ otherwise.

3. B

4 Linear Regression

1. (a) 209.2000

   The gradient of this objective function is:

$$\frac{\partial \ell}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} 2 \cdot (wx_i + b - y_i) \cdot x_i$$
$$= \frac{1}{5}\sum_{i=1}^{5} 2x_i(3x_i - y_i)$$
$$= \frac{1}{5}[468 + 24 + 180 + 6 + 368]$$
$$= 209.2$$

   (b) 29.2000

   The gradient of this objective function is:

$$\frac{\partial \ell}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} 2 \cdot (wx_i + b - y_i)$$
$$= \frac{1}{5}\sum_{i=1}^{5} 2(3x_i - y_i)$$
$$= \frac{1}{5}[52 + 12 + 30 + 6 + 46]$$
$$= 29.2$$

   (c) Weight: 0.9080

   Intercept: -0.2920

$$w \leftarrow w - \alpha\frac{\partial \ell}{\partial w} = 3.0 - 0.01 \cdot 209.2 = 0.9080$$
$$b \leftarrow b - \alpha\frac{\partial \ell}{\partial w} = 0.0 - 0.01 \cdot 29.2 = -0.2920$$

2. (a) C. In this problem, we are looking at linear regression in a one-dimensional case. We are looking at what happens when we added a positive constant α to all the x's and a constant β to all the y's to see how the solution to the linear regression problem will change. The constraint of $w_1\alpha \neq \beta$ says that the constants we are adding will not simply "slide" the points along the line $y = w_1x + \beta_1$. Since this is the case, the intercept of the solution will change (as an example think of just a perfect line where you shift by α and β). The slope of the solution will not change since all points are shifting uniformly (again the simplest case is a perfect line).

   (b) In general, by duplicating the data, the coefficients will generally change. This can actually be seen as a form of weighted linear regression, the duplicated rows are considered more important, as they will appear multiple times in objective function to be minimized.

3. (a) $\ell(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell^{(i)}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \log(\cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)}))$

   (b) $\frac{\partial \ell^{(i)}(\boldsymbol{w})}{\partial w_j} = \frac{1}{\cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} \cdot \frac{\partial \cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}{\partial w_j} = \frac{\sinh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}{\cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} \cdot \frac{\partial \boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)}}{\partial w_j} = \frac{\sinh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}{\cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} \cdot x_j^{(i)}$

   $\big(\tanh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)}) \cdot x_j^{(i)}$ and $\frac{e^{(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} - e^{-(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}}{e^{(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} + e^{(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}} \cdot x_j^{(i)}$ are also correct$\big)$ since $\sinh(x) = \frac{e^x - e^{-x}}{2}$

   (c) $\frac{\sinh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})}{\cosh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})} \cdot x^{(i)} = \big(\tanh(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})\big) \cdot x^{(i)}$

   (d)
```
w = zeros(k)
grad = zeros(k)
for i in range(N):
    grad = grad + gradient[i] / N
w = w - 0.1 * grad
```

5 Regularization

1. C

2. For a feature dimension on which the training data is separable, if positive feature values increase the probability of a positive class label, then we can always increase the log-likelihood by increasing the corresponding weight because this will increase the probability (likelihood) of positive labels and decrease the probability of negative labels. If negative feature values increase the probability of a positive class label, then we can similarly always increase the log-likelihood by driving the weight to negative infinity.

3. Option 2, 3 are correct. 1 refers to ℓ_0 regularization.

6 Neural Network Warm-up

1. False
2. (a) A, B, and C; linear layers or layers with no activation function can be collapsed to make equivalent neural networks.

   (b) B

   (c) True; even though these two models are effectively the same (the linear layer does not add any expressiveness to g or any classifier that g learns, f can also learn), there's no guarantee that at convergence, they will learn the same or equally good classifiers so it very well can be the case that g achieves a lower training loss.
3. A; trimming is not guaranteed to help or hurt: it tends to decrease overfitting (removing edges makes the model simpler) but it could be that your original neural network was appropriately or even underfit so trimming could increase the validation loss. Note the key caveat that you trimmed at least one edge; if you were not forced to trim any edges, then the trimming procedure could not increase the validation loss as the original neural network would be a valid option.

7 Example Feed Forward and Backpropagation

1. (a) 4

   (b) 3

   (c) $\boldsymbol{\alpha}\mathbf{x}^{(1)}$

   (d) B. Append a value of 1 to be the first entry of $\mathbf{z}$

   (e) $\hat{\mathbf{y}} = \begin{bmatrix} \frac{e^{\mathbf{b}_1}}{e^{\mathbf{b}_1}+e^{\mathbf{b}_2}+e^{\mathbf{b}_3}} \\ \frac{e^{\mathbf{b}_2}}{e^{\mathbf{b}_1}+e^{\mathbf{b}_2}+e^{\mathbf{b}_3}} \\ \frac{e^{\mathbf{b}_3}}{e^{\mathbf{b}_1}+e^{\mathbf{b}_2}+e^{\mathbf{b}_3}} \end{bmatrix}$
2. (a)

$$\frac{\partial \ell}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

(b) The chain rule gives $\frac{\partial \ell}{\partial b_k} = \sum_l \frac{\partial \ell}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial b_k} = -\sum_l y_l(\mathbb{I}[k=l] - \hat{y}_k) = -(\sum_l y_l \mathbb{I}[k=l] - \hat{y}_k \sum_l y_l) = -(y_k - \hat{y}_k)$. Note that $\mathbf{y}$ is a one-hot encoding vector, and thus $\sum_l y_l = 1$. We know to include the sum over y_l because, from examining the computation graph and tracing back the arrows, b_k depends on all y_l due to the definition of the softmax.

$$\begin{aligned}
\frac{\partial \ell}{\partial b_k} &= \sum_l \frac{\partial \ell}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial b_k} && \frac{\partial \ell}{\partial \hat{y}_l} = -\frac{y_l}{\hat{y}_l} \\
&= -\sum_l y_l \left(\mathbb{I}[k=l] - \hat{y}_k\right) \\
&= -\left(y_k - \sum_l y_l \hat{y}_k\right) \\
&= -\left(y_k - \hat{y}_k \sum_l y_l\right) && \sum_l y_l = 1 \\
&= \hat{y}_k - y_k
\end{aligned}$$

(c)

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_{kj}} &= \frac{\partial \ell}{\partial b_k} \frac{\partial b_k}{\partial \beta_{kj}} = \frac{\partial \ell}{\partial b_k} z_j \\
\frac{\partial \ell}{\partial \beta} &= \frac{\partial \ell}{\partial \mathbf{b}} \mathbf{z}^T
\end{aligned}$$

(d) C. The forward pass does not compute $\boldsymbol{\beta}_{k,0}$ as a function of α, so the backward pass does not need to compute $\frac{\partial \ell}{\partial \alpha}$ as a function of $\frac{\partial \ell}{\partial \boldsymbol{\beta}_{k,0}}$.

(e)

$$\begin{aligned}
\frac{\partial \ell}{\partial z_j} &= \sum_k \frac{\partial \ell}{\partial b_k} \frac{\partial b_k}{\partial z_j} = \sum_k \frac{\partial \ell}{\partial b_k} \beta^*_{kj} \\
\frac{\partial \ell}{\partial \mathbf{z}} &= (\beta^*)^T \frac{\partial \ell}{\partial \mathbf{b}}
\end{aligned}$$

(f)

$$\frac{\partial \ell}{\partial z_j} \frac{\partial z_j}{\partial a_j} = \frac{\partial \ell}{\partial z_j} z_j(1 - z_j)$$

(g)

$$\begin{aligned}
\frac{\partial \ell}{\partial \alpha_{ji}} &= \frac{\partial \ell}{\partial a_j} \frac{\partial a_j}{\partial \alpha_{ji}} = \frac{\partial \ell}{\partial a_j} x_i^{(1)} \\
\frac{\partial \ell}{\partial \alpha} &= \frac{\partial \ell}{\partial \mathbf{a}} \mathbf{x}^{(1)T}
\end{aligned}$$