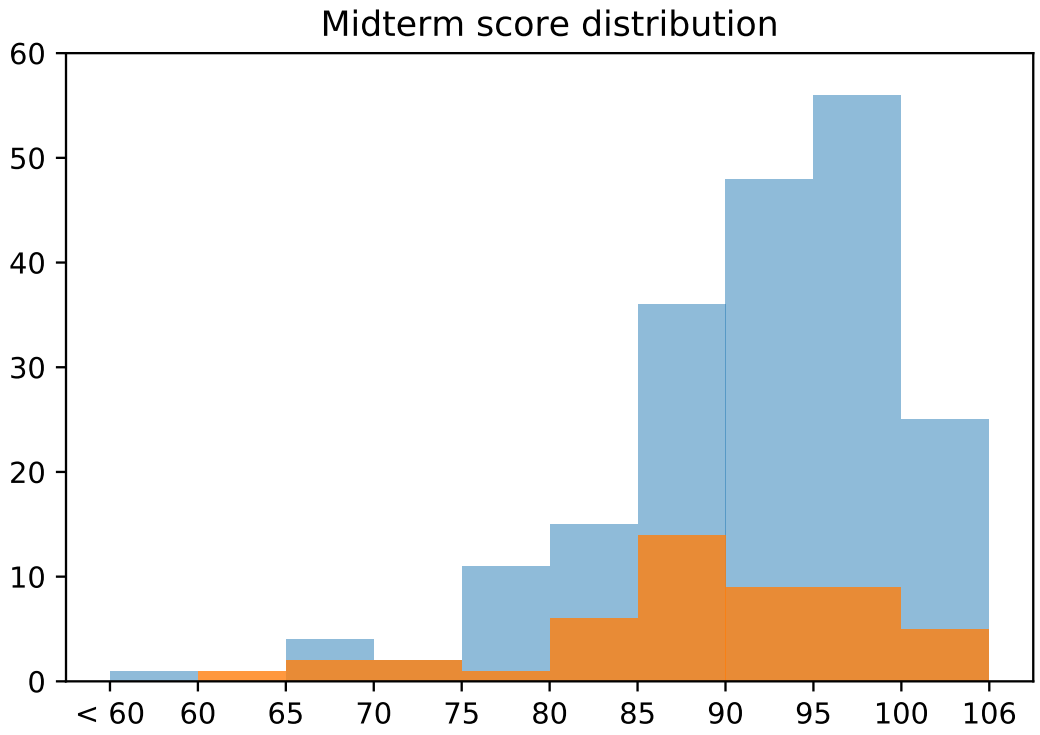
ECE C147/C247, Winter 2020 **Midterm Overview**
Department of Electrical Engineering Prof. J.C. Kao
UCLA ECE TAs W. Feng & M. Kleinman & J. Lee & K. Liang & C. Zheng

Midterm statistics:



Overall statistics
High score: 106
Mean score: 91.1
Median score: 92.5
Standard deviation: 8.6

C247
High score: 106
Mean score: 91.6
Median score: 93
Standard deviation: 8.3

C147
High score: 103
Mean score: 89.0
Median score: 89.5
Standard deviation: 9.4

Comments:

- This is the highest mean (91.1) I have ever seen on any exam in ECE. Congratulations on your excellent performance.
- If you have grading questions, please submit through Gradescope. Regrades should be submitted if you believe we applied the rubric mistakenly. Inquiries into particular questions should be directed to the person who graded that question.

  Questions 1a, 1b: Ken-Fu
  Questions 1c, 1d: Weixi
  Question 2: Johannes
  Question 3: Michael
  Question 4a: Cheng
  Question 4b: Jonathan
  Bonus: Jonathan

- Regrades close one week from today (March 2, 2020).

ECE C147/C247, Fall 2020 **Midterm**

Department of Electrical and Computer Engineering
University of California, Los Angeles

Prof. J.C. Kao
TAs: W. Feng, J. Lee,
K-F. Liang, M. Kleinman, C. Zheng

UCLA True Bruin academic integrity principles apply.
Open: Four cheat sheets allowed.
Closed: Book, computer, internet.
2:00pm-3:50am.
Wednesday, 19 Feb 2020.

State your assumptions and reasoning.
No credit without reasoning.
Show all work on these pages.

Name: _____

Signature: _____

ID#: _____

Problem 1	_____ / 30
Problem 2	_____ / 20
Problem 3	_____ / 20
Problem 4	_____ / 30
BONUS	_____ / 6 bonus points
Total	_____ / 100 points + 6 bonus points

1. **Short Answer on ML Basics** (30 points)

   (a) (5 points) Your colleague is tasked with designing the image recognition component for a self-driving vehicle. His goal is to classify images in real-time, identifying objects on the road (like a stop sign, pedestrian, traffic light, etc.) Each image is 1024×1024 pixels^2. He desires to use a k-nearest neighbors classifier, with distance measured by the Euclidean (L2) norm. What are two reasons he should avoid using the classifier? Justify your answer in no more than 4 sentences.

   Solution: 1. Testing is computationally expensive. 2. Distance measurements in a high dimensional space do not necessarily reflect similarity between two data points due to the curse of dimensionality).

   (b) (5 points) Batch normalization aims to normalize each artificial neuron so that its mean is 0 and its variance is 1. Why, intuitively, should this help with training deep neural networks? Justify your answer in no more than 4 sentences.

   Solution: When performing gradient updates, our gradient updates for a given weight matrix assume the other ones are held constant. This is not true in general. Therefore, gradient steps in earlier layers may dramatically change the statistics of the neurons for the next layer. Batch normalization normalizes these statistics, so that they aren't dramatically different after each gradient step. We also accepted that batch normalization prevents the outputs of each layer from being too large or too small.

   (c) (10 Points) The loss function of a softmax classifier can be written as:

$$\mathcal{L} = \sum_{i=1}^{m} \left(\log \sum_{j=1}^{c} e^{a_j(x^{(i)})} - a_{y^{(i)}}(x^{(i)}) \right)$$

   where $(\mathbf{x}^{(i)}, y^{(i)})$ are data pairs (of which there are m) and $a_j(x^{(i)}) = \mathbf{w}_j^T \mathbf{x}^{(i)} + b_j$ is the score of class j (there are c total classes). Calculate the gradient of $\mathcal{L}$ with respect to all the weights and biases.

   Solution:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_k} &= \sum_{i=1}^{m} \left(\frac{e^{a_k(x^{(i)})}}{\sum_{j=1}^{c} e^{a_j(x^{(i)})}} x^{(i)} - \mathbb{1}(k = y^{(i)}) \cdot x^{(i)} \right) \\ \frac{\partial \mathcal{L}}{\partial b_k} &= \sum_{i=1}^{m} \left(\frac{e^{a_k(x^{(i)})}}{\sum_{j=1}^{c} e^{a_j^{(x^{(i)})}}} - \mathbb{1}(k = y^{(i)}) \right) \end{aligned}$$

   (d) (10 Points) Assume that you have a handwritten digit image dataset for a classification task. All the images are of size 32×32. This dataset is split into training, validation and testing. Your CNN takes inputs of size 28×28. You resize the 32×32 images to 28×28 and then feed them into the network. During training, you observe that the training error is always decreasing while the validation error decreases first and then

increases. After training completes, the testing error is much higher than the training error.

i. (5 Points) What is the problem with your trained CNN? What are 2 potential reasons for this? Justify your answer in no more than 4 sentences.
Solution: Problem: Overfitting.
Reasons: The model is overtrained. The training data is not enough. The training data is enough but highly correlated. The model is too complex for the dataset. The data is too noisy (and other reasonable answers).

ii. (5 Points) Your colleague suggests that you need a larger training set with more images. However, no additional images are available. What can you do to the existing training images to increase the number of training set examples? Recall that the images are of digits. Justify your answer in no more than 3 sentences.

Solution: Randomly crop out several 28×28 regions from each image. Scaling up or down the images and then resize to 28×28. Horizontal or vertical shift by random values and then resize. Random rotation within a certain range of degrees and then resize (and other reasonable methods).
Not a valid answer: horizontal or vertical flip as most of the digits are not horizontally or vertically symmetric.

2. **Backpropagation** (20 points). Suppose you aim to predict $\mathbf{y}$ from $\mathbf{x}$ using the following neural network:

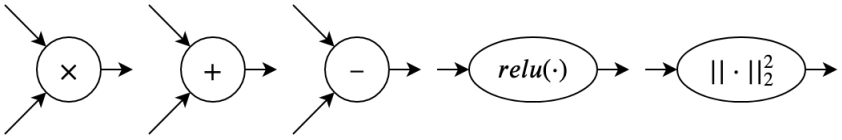
$$\mathbf{z} = \mathbf{W}_3 \text{relu}(\mathbf{W}_1 \mathbf{x}) + \mathbf{W}_2 \mathbf{x}.$$

Here, $\mathbf{z}$ is an estimate of $\mathbf{y}$ and the loss function is:

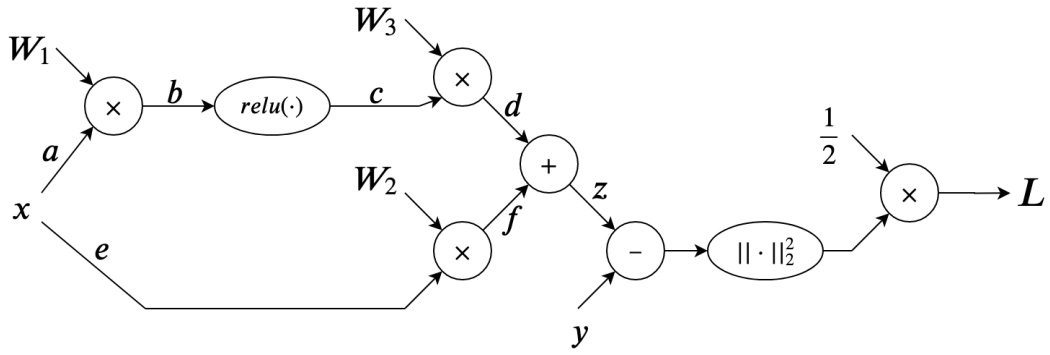
$$\mathcal{L} = \frac{1}{2}||\mathbf{z} - \mathbf{y}||_2^2 = \frac{1}{2}(\mathbf{z} - \mathbf{y})^T(\mathbf{z} - \mathbf{y}),$$

with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{z} \in \mathbb{R}^m$, $\mathbf{W}_1 \in \mathbb{R}^{p \times n}$, $\mathbf{W}_2 \in \mathbb{R}^{m \times n}$, and $\mathbf{W}_3 \in \mathbb{R}^{m \times p}$,

(a) (4 points) Draw a computational graph of this network to compute the loss $\mathcal{L}$. You may use the following elements:



Solution:



(b) (4 points) What is $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$ (i.e., $\nabla_{\mathbf{z}} \mathcal{L}$)? Your answer must be simplified to only contain the following allowed terms: $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3$? (To be clear, you do not need to simplify $\mathbf{z}$, since that is an allowed term.)

Solution:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}||\mathbf{z} - \mathbf{y}||_2^2 \\
&= \frac{1}{2}(\mathbf{z} - \mathbf{y})^T(\mathbf{z} - \mathbf{y}) \\
&= \frac{1}{2}(\mathbf{z}^T\mathbf{z} - 2\mathbf{y}^T\mathbf{z} + \mathbf{y}^T\mathbf{y}) \\
\frac{\partial \mathcal{L}}{\partial \mathbf{z}} &= \mathbf{z} - \mathbf{y}
\end{aligned}$$

(c) (12 Points) In terms of $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$, find:

i. (6 points) $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}$

ii. (6 points) $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$

You should not need to derive any local gradients, as we have derived all of the local gradients for these operations in class. In particular, recall the following:

- For $\mathbf{p} = \mathbf{C}\mathbf{q}$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{C}} = \frac{\partial \mathcal{L}}{\partial \mathbf{p}} \mathbf{q}^T$$
$$\frac{\partial \mathbf{p}}{\partial \mathbf{q}} = \mathbf{C}^T$$

- For $\mathbf{v} = \text{relu}(\mathbf{u})$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \mathbb{1}(\mathbf{u}) \odot \frac{\partial \mathcal{L}}{\partial \mathbf{v}}$$

  where $\mathbb{1}(\cdot)$ is the indicator function defined as:

$$\mathbb{1}(\mathbf{u}) = \begin{cases} 1, & \mathbf{u}_i \geq 0 \\ 0, & \mathbf{u}_i < 0 \end{cases}$$

  and $\odot$ is the Hadamard (elementwise) product. Be sure also to define any intermediate variables used or label them on the computational graph.

Solution:
Using the chain rule and the computational graph:
(i)

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{W_1}} &= \frac{\partial \mathbf{b}}{\partial \mathbf{W_1}} \frac{\partial \mathbf{c}}{\partial \mathbf{b}} \frac{\partial \mathbf{d}}{\partial \mathbf{c}} \frac{\partial \mathbf{z}}{\partial \mathbf{d}} \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \\
\frac{\partial \mathbf{z}}{\partial \mathbf{d}} &= \mathbf{I} \\
\frac{\partial \mathbf{d}}{\partial \mathbf{c}} &= \mathbf{W}_3^T \\
\frac{\partial \mathbf{c}}{\partial \mathbf{b}} &= diag(\mathbb{1}(\mathbf{b})) \\
\frac{\partial \mathcal{L}}{\partial \mathbf{b}} &= \frac{\partial \mathbf{c}}{\partial \mathbf{b}} \frac{\partial \mathbf{d}}{\partial \mathbf{c}} \frac{\partial \mathbf{z}}{\partial \mathbf{d}} \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \\
&= \mathbb{1}(\mathbf{b}) \odot (\mathbf{W}_3^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}) \\
\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} &= \frac{\partial \mathbf{b}}{\partial \mathbf{W}_1} \frac{\partial \mathcal{L}}{\partial \mathbf{b}} \\
&= \frac{\partial \mathcal{L}}{\partial \mathbf{b}} \mathbf{x}^T \\
&= \left(\mathbb{1}(\mathbf{b}) \odot (\mathbf{W}_3^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}) \right) \mathbf{x}^T \\
&= \left(\mathbb{1}(\mathbf{W_1 x}) \odot (\mathbf{W}_3^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}) \right) \mathbf{x}^T
\end{aligned}$$

(ii) To account for the law of total derivatives, we define $\mathbf{a} = \mathbf{e} = \mathbf{x}$. This results in:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{a}} + \frac{\partial \mathcal{L}}{\partial \mathbf{e}} \\
&= \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \frac{\partial \mathcal{L}}{\partial \mathbf{b}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathcal{L}}{\partial \mathbf{f}} \\
&= \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \frac{\partial \mathcal{L}}{\partial \mathbf{b}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{f}} \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \\
&= \mathbf{W}_1^T \left(\mathbb{1}(\mathbf{b}) \odot (\mathbf{W}_3^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}) \right) + \mathbf{W}_2^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \\
&= \mathbf{W}_1^T \left(\mathbb{1}(\mathbf{W_1 x}) \odot (\mathbf{W}_3^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}) \right) + \mathbf{W}_2^T \frac{\partial \mathcal{L}}{\partial \mathbf{z}}
\end{aligned}$$

with

$$\begin{aligned}
\frac{\partial \mathbf{z}}{\partial \mathbf{f}} &= \mathbf{I} \\
\frac{\partial \mathbf{f}}{\partial \mathbf{x}} &= \mathbf{W}_2^T
\end{aligned}$$

3. **Regularization and Optimization** (20 points)

For each statement below, determine whether it is true or false. You must justify your answer to receive full credit.

(a) (5 points) Regularization is helpful to achieve lower loss on the training set.

Solution: False. Helpful for generalization/ test set performance.

(b) (5 points) When performing (minibatch) stochastic gradient descent, you are calculating the gradient of the training set exactly.

Solution: False. Noisy approximation because you're using fewer samples.

(c) (5 points) An optimizer using Adagrad will always have an equal or smaller learning rate in successive iterations.

Solution: True. The gradient history stays the same or increases in each dimension for every iteration of Adagrad.

(d) (5 points) You are performing stochastic gradient descent with a learning rate $\epsilon = 5 \times 10^{-6}$. You believe your network is correctly implemented, but when training your network, the value of the loss remains approximately constant across many training iterations. You should try increasing ϵ.

Solution: True. This likely reflects that the learning rate is too small. Some points were also given for False, if the question was interpreted that the loss had plateaued, assuming proper justification.

4. **Fully Connected and Convolutional Neural Networks** (30 points)

(a) **Residual Fully connected networks.** In this problem, we modify the two-layer FC network with residual connections, according to the equation below. Given the input vector $\mathbf{x} \in \mathbb{R}^n$, the forward propagation is,

$$\text{First layer: } \mathbf{h}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{W}_s\mathbf{x}$$
$$\text{Second (Output) layer: } \mathbf{z} = \mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2$$

where $\mathbf{b}_1 \in \mathbb{R}^m$ and $\mathbf{b}_2 \in \mathbb{R}^l$. $\mathbf{W}_s$ is the linear mapping for the "shortcut" connection. f is the non-linear activation function like ReLU.

i. (3 points) Write the dimension of the following variables: $\mathbf{W}_1, \mathbf{W}_s, \mathbf{W}_2$.
Solution: $\mathbf{W}_1 \in \mathbb{R}^{m\times n}$, $\mathbf{W}_s \in \mathbb{R}^{m\times n}$, $\mathbf{W}_2 \in \mathbb{R}^{l\times m}$.

ii. (3 points) How many trainable parameters does this residual FC network have? Your answer should be expressed in terms of l, m, n.
Solution: Total number of trainable parameters:$= mn + m + mn + ml + l = 2mn + ml + m + l$.

iii. (9 points) To select a proper activation function f, we are considering whether it: (1) saturates (and if so, for what values does it saturate), (2) is differentiable everywhere, (3) approximates identity near the origin (activation functions where $f(0) = 0$ and $f'(0) = 1$ and f' is continuous at 0 have this property.). For the following candidate activation functions, explain from the three aspects and **Briefly** justify each solution you give.

A. Hyperbolic tangent: $\tanh(x)$.
Solution: Saturates as x becomes very positive or negative. Is differentiable everywhere as it has no discontinuities. Approximates identity near the origin.

B. Leaky ReLU: $\max(0.01x, x)$.
Solution: Does not saturate. Is not differentiable at $x = 0$. Does not approximates identity near the origin.

C. Gaussian Error Linear Unit (GELU): it is popular in recent transformer structures like BERT. This activation function can be approximated as $\text{GELU}(x) = 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)]$

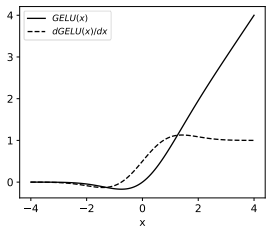


Figure 1: Gaussian Error Linear Unit (GELU)

Solution: Saturates when x is very negative. Is differentiable everywhere as it has no discontinuities. Does not approximate identity near the origin.

(b) (15 points) **Convolutional neural networks.** Consider a convolutional neural network (CNN) that is used to classify CIFAR-10 images that are $32 \times 32 \times 3$. The first layer of the CNN is a convolutional layer, whose output is $26 \times 26 \times 16$.

i. (5 points) How many filters are in the first convolutional layer, and what are the dimensions of each filter? Assume there is no zero padding and the stride is equal to 1.

Solution: There are 16 filters, each of which is $7 \times 7 \times 3$.

ii. (5 points) There is a second convolutional layer immediately after the first convolutional layer. In the second convolutional layer, there are 32 filters that are 3×3. What is the depth of each filter, and how many total trainable parameters are there in this layer? (Do not include biases in the number of parameters.)

Solution: Their depth is 16. Therefore, the number of trainable parameters is $32 \times 3 \times 3 \times 16$.

iii. (5 points) In general, why do convolutional layers have less parameters than fully connected layers? Justify your answer in no more than 3 sentences.

Solution: Convolutional layers have sparse connectivity and share parameters.

5. **(Bonus, 6 points)**. Binary cross-entropy loss. In class, we have said that maximizing the likelihood of the observed data is equivalent to minimizing the "cross-entropy" loss in machine learning. Consider that we have data: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$. The data comes from one of two classes, class 0 or class 1, i.e., $y^{(j)} \in \{0, 1\}$. For each input sample, $\mathbf{x}^{(j)}$, we calculate $\hat{y}^{(j)}$, which is the probability that $y^{(j)} = 1$, i.e.,

$$\hat{y}^{(j)} = \Pr(y^{(j)} \text{ belongs to class } 1 | \mathbf{x}^{(j)}, \theta)$$

Though not necessary for this question, this probability could be given by, for example, the sigmoid function:

$$\begin{aligned} \hat{y}^{(j)} &= \sigma(\theta, \mathbf{x}^{(j)}) \\ &= \frac{1}{1 + \exp(-\theta^T \mathbf{x}^{(j)})} \end{aligned}$$

The cross-entropy loss is:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \left[y^{(j)} \log \hat{y}^{(j)} + (1 - y^{(j)}) \log(1 - \hat{y}^{(j)}) \right]$$

Show that minimizing the cross-entropy loss is equivalent to maximizing the likelihood of the data,

$$\arg\max_{\theta} p((\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)}) | \theta).$$

Assume that each data point $(\mathbf{x}^{(j)}, y^{(j)})$ is independent given the parameters θ (i.e., the same assumption we made in class to simplify the joint probability).

Hint: When $y^{(j)}$ is binary, show that $p(y^{(j)} | \mathbf{x}^{(j)}, \theta)$, which is the probability that $\mathbf{x}^j$ belongs to class $y^{(j)}$ under a model with parameters θ, can be written as

$$p(y^{(j)} | \mathbf{x}^j, \theta) = (\hat{y}^{(j)})^{y^{(j)}} \cdot (1 - \hat{y}^{(j)})^{1 - y^{(j)}}$$

Solution: First, we show the hint. Note that:

$$\begin{aligned} \Pr(y^{(j)} = 1 | \mathbf{x}^j, \theta) &= \hat{y}^{(j)} \\ \Pr(y^{(j)} = 0 | \mathbf{x}^j, \theta) &= 1 - \hat{y}^{(j)} \end{aligned}$$

And therefore, when $y^{(j)} = 1$, then $p(y^{(j)} | \mathbf{x}^j, \theta) = \hat{y}^{(j)}$ and when $y^{(j)} = 0$, then $p(y^{(j)} | \mathbf{x}^j, \theta) = 1 - \hat{y}^{(j)}$. The single expression

$$p(y^{(j)} | \mathbf{x}^j, \theta) = (\hat{y}^{(j)})^{y^{(j)}} \cdot (1 - \hat{y}^{(j)})^{1 - y^{(j)}}$$

captures both of these cases, since when $y^{(j)} = 1$, then $1 - y^{(j)} = 0$, leaving just the $\hat{y}^{(j)}$ term. Likewise, when $y^{(j)} = 0$, then $1 - y^{(j)} = 1$, leaving just the $(1 - \hat{y}^{(j)})$ term.

The likelihood of the data is:

$$\begin{aligned}
\mathcal{L}(\theta) &= \prod_{j=1}^{N} p(\mathbf{x}^{(j)}, y^{(j)}|\theta) \\
&= \prod_{j=1}^{N} p(\mathbf{x}^{(j)}|\theta) p(y^{(j)}|\mathbf{x}^{(j)}, \theta) \\
&= \prod_{j=1}^{N} p(\mathbf{x}^{(j)}|\theta) p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta)^{y^{(j)}} (1 - p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta))^{(1-y^{(j)})}
\end{aligned}$$

We take the log of both sides, and also recognize that $p(\mathbf{x}^{(j)})$ is independent of θ. This gives that:

$$\begin{aligned}
\arg\max_{\theta} \log \mathcal{L}(\theta) &= \arg\max_{\theta} \sum_{j=1}^{N} \log p(y^{(j)}|\mathbf{x}^{(j)}, \theta) \\
&= \arg\max_{\theta} \sum_{j=1}^{N} \log p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta)^{y^{(j)}} + (\log(1 - p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta)))^{(1-y^{(j)})} \\
&= \arg\max_{\theta} \sum_{j=1}^{N} y^{(j)} \cdot \log p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta) + (1 - y^{(j)}) \cdot (\log(1 - p(y^{(j)} = 1|\mathbf{x}^{(j)}, \theta))) \\
&= \arg\max_{\theta} \sum_{j=1}^{N} y^{(j)} \cdot \log \hat{y}^{(j)} + (1 - y^{(j)}) \cdot \log(1 - \hat{y}^{(j)})
\end{aligned}$$