University of Illinois Fall 2023

ECE 544: Pattern Recognition
Problem Set 3

Due: Friday, October 13, 2023, 11:59 pm

1. **[Deep Net]**
   We want to train a simple deep net $f(w_3, w_2, w_1, x) = w_3\sigma_2(w_2\sigma_1(\underbrace{w_1 x}_{x_1}))$ where $w_1, w_2, w_3, x \in \mathbb{R}$ are real valued and $\sigma_1(x) = \sigma_2(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function.

   (a) Draw the computation graph that is specified by the function f.

   (b) Compute $\frac{\partial \sigma_1}{\partial x}$ and provide the answer (1) using only x, the exp-function and the square function, and (2) using only $\sigma_1(x)$.

   (c) Describe briefly what is meant by a 'forward pass' and a 'backward pass'?

   (d) Compute $\frac{\partial f}{\partial w_3}$. Which result should we retain from the forward pass in order to make computation of this derivative easy?

   (e) Compute $\frac{\partial f}{\partial w_2}$. Make use of the second option obtained in part (b). Which results should we retain from the forward pass in order to make computation of this derivative easy?

   (f) Compute $\frac{\partial f}{\partial w_1}$. Make use of the second option obtained in part (b). Which results should we retain from the forward pass in order to make computation of this derivative easy? In what order should we compute the derivatives $\frac{\partial f}{\partial w_3}$, $\frac{\partial f}{\partial w_2}$ and $\frac{\partial f}{\partial w_1}$ in order to obtain the result as early as possible and in order to reuse as many results as possible. How is this order related to the forward pass?

   (g) We now want to train a convolutional neural net for 10-class classification of MNIST images which are of size 28×28. As a first layer we use 20 2d convolutions each with a filter size of 5×5, a stride of 1 and a padding of 0. What is the output dimension after this layer? Subsequently we apply max-pooling with a size of 2×2. What is the output dimension after this layer?

   (h) After having applied the two layers (convolution + pooling) designed in part (g) we want to use a second convolution + max-pooling operation. The max-pooling operation has a filter size of 2×2. The desired output should have 50 channels and should be of size 4×4. What is the filter size, the stride, the padding and the channel dimension of the second convolution operation?

   (i) Complete `A5_DeepNet.py` by first implementing the two (convolution+pooling) operations. We also want to apply two linear layers. The first one maps from a $50 \cdot 4 \cdot 4$ dimensional space to a 500 dimensional one. After both convolutions and the first linear layer we also want to apply ReLU non-linearities. Also implement the two linear layers and the ReLU non-linearities. Provide the entire deep net class here. What is the best test set accuracy that you observed during training with this architecture? How many parameters does your network have (including biases)?

2. **[Representation of Deep Neural Networks]**

We will use the XOR dataset, $\mathcal{D}$, shown in Table 1. Each example $\mathbf{x}^{(i)} \in \mathbb{R}^2$ and the label $y \in \{0, 1\}$. We are interested in designing classification models for this dataset.

i	$\mathbf{x}_0^{(i)}$	$\mathbf{x}_1^{(i)}$	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Table 1: The XOR Dataset $\mathcal{D}$.

(a) Consider a model with the following parameterization:

$$p(y^{(i)}|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\intercal \mathbf{x}^{(i)} - b)}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$.

What is the highest accuracy for this model on the XOR dataset? **Note:** To compute accuracy, we use a threshold of 0.5, *i.e.*, the final prediction of the model is $\delta[p(y^{(i)}|\mathbf{x}) > 0.5]$, where δ denotes the indicator function.

(b) Consider another model with the parametrization shown below:

$$\tilde{y}^{(i)} = \frac{1}{1 + \exp(-a_2^{(i)})} \tag{2}$$

$$a_2^{(i)} = \theta^\intercal \max(\mathbf{a_1^{(i)}}, 0) + b \tag{3}$$

$$\mathbf{a_1^{(i)}} = \mathbf{W}\mathbf{x^{(i)}} + \mathbf{c} \tag{4}$$

where $\theta \in \mathbb{R}^2$, $b \in \mathbb{R}$, $\mathbf{W} \in \mathbb{R}^{2\times 2}$ and $\mathbf{c} \in \mathbb{R}^2$.

Find a θ and b that achieve 100 % accuracy on the XOR dataset, given $\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{c} = [0, -1]^\intercal$. *To show your work, write out* $\mathbf{a_1^{(i)}}$ *and* $a_2^{(i)}$ *for the four datapoints in the XOR dataset and your choice of* θ*, and* b*.*

Note: To compute accuracy, we use a threshold of 0.5, *i.e.*, the final prediction of the model is $\delta[\tilde{y}^{(i)} > 0.5]$, where δ denotes the indicator function.

(c) To learn the parameters of the model in (b), the learning problem is formulated as the following program:

$$\min_{\theta, b, \mathbf{W}, \mathbf{c}} \mathcal{L} := \sum_{(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D}} ||y^{(i)} - \tilde{y}^{(i)}||_2^2. \tag{5}$$

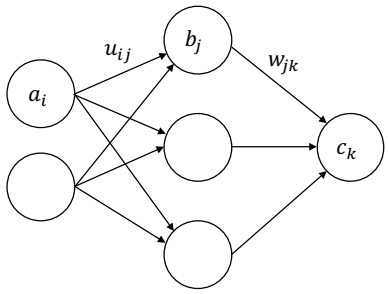
Write out $\frac{\partial \mathcal{L}}{\partial \tilde{y}^{(i)}}$. You may use the terms $y^{(i)}$ and $\tilde{y}^{(i)}$.

(d) Write out $\frac{\partial \tilde{y}^{(i)}}{\partial a_2^{(i)}}$. You may use the terms $a_2^{(i)}$.

(e) Write out $\frac{\partial \mathcal{L}}{\partial \mathbf{c}}$. You may use the terms $\frac{\partial \mathcal{L}}{\partial \tilde{y}^{(i)}}$, $\frac{\partial \tilde{y}^{(i)}}{\partial a_2^{(i)}}$, $\mathbf{W}$, $\mathbf{x}^{(i)}$, $\mathbf{c}$, $\theta, b, \mathbf{a}_1^{(i)}$ and $\delta[\cdot]$ as the indicator function.

3. [**Backpropagation**]

Consider the deep net in the figure below consisting of an input layer, an output layer, and a hidden layer. The feed-forward computations performed by the deep net are as follows: every input a_i is multiplied by a set of fully-connected weights u_{ij} connecting the input layer to the hidden layer. The resulting weighted signals are then summed and combined with a bias e_j. This results in the activation signal $z_j = e_j + \sum_i a_i u_{ij}$. The hidden layer applies activation function g on z_j resulting in the signal b_j. In a similar fashion, the hidden layer activation signals b_j are multiplied by the weights connecting the hidden layer to the output layer w_{jk}, a bias f_k is added and the resulting signal is transformed by the output activation function g to form the network output c_k. The loss between the desired target t_k and the output c_k is given by the MSE: $E = \frac{1}{2}\sum_k (c_k - t_k)^2$, where t_k denotes the ground truth signal corresponding to c_k. Training a neural network involves determining the set of parameters $\theta = \{U, W, e, f\}$ that minimize E. This problem can be solved using gradient descent, which requires determining $\frac{\partial E}{\partial \theta}$ for all θ in the model.



(a) For $g(x) = \sigma(x) = \frac{1}{1+e^{-x}}$, compute the derivative $g'(x)$ of $g(x)$ as a function of $\sigma(x)$.

(b) Compute $\frac{\partial E}{\partial w_{jk}}$. Use c_k, t_k, g', f_k, b_j, w_{jk}. Note the use of g' to simplify the expression.

(c) Compute $\frac{\partial E}{\partial f_k}$. Use c_k, t_k, g', f_k, b_j, w_{jk}. Note the use of g' to simplify the expression.

(d) Compute $\frac{\partial E}{\partial u_{ij}}$. Use c_k, t_k, g', f_k, b_j, w_{jk}, a_i. Note the use of g' to simplify the expression.

(e) Compute $\frac{\partial E}{\partial e_j}$. Use c_k, t_k, g', f_k, b_j, w_{jk}, a_i. Note the use of g' to simplify the expression.

4. [**Structured Prediction**]

We are interested in jointly predicting/modeling two discrete random variables $y = (y_1, y_2) \in \mathcal{Y}$ with $y_i \in \mathcal{Y}_i = \{0, 1\}$ for $i \in \{1, 2\}$ and $\mathcal{Y} = \prod_{i\in\{1,2\}} \mathcal{Y}_i$. We define the joint probability distribution to be $p(y) = p(y_1, y_2) = \frac{1}{Z} \exp F(y)$.

(a) What is the value of Z (in terms of $F(y)$) and what is Z called? How many configurations do we need to sum over? Provide the expression using $\mathcal{Y}_i$.

(b) Next we want to solve (for any hyperparameter ϵ)

$$\max_{\hat{p}\in\Delta_{\mathcal{Y}}} \sum_{y\in\mathcal{Y}} \hat{p}(y)F(y) - \sum_{y\in\mathcal{Y}} \epsilon \hat{p}(y) \log \hat{p}(y), \tag{6}$$

where $\Delta_{\mathcal{Y}}$ denotes the probability simplex, *i.e.*, $\hat{p}$ is a valid probability distribution over its domain $\mathcal{Y}$. Using general notation, write down the Lagrangian and compute its derivative w.r.t. $\hat{p}(y)$ $\forall y \in \mathcal{Y}$. Subsequently, find the optimal $\hat{p}^*$. What is the resulting optimal cost function value for the program given in Eq. (6)? How does this result relate to part (a)?

(c) For the program in Eq. (6) assume now $\epsilon = 0$, *i.e.*, we are searching for that configuration $y^* = \arg\max_{\hat{y} \in \mathcal{Y}} F(\hat{y})$ which maximizes $F(y)$. Assume $F(y) = f_1(y_1) + f_2(y_2) + f_{1,2}(y_1, y_2)$. How many different values can the functions f_1, f_2 and $f_{1,2}$ result in?

(d) As discussed in class, finding the global maximizer can be equivalently written as the following integer linear program:

$$\max_{b} \sum_{r, y_r} b_r(y_r) f_r(y_r) \qquad \text{s.t.} \qquad \begin{cases} b_r(y_r) \in \{0, 1\} & \forall r, y_r \\ \sum_{y_r} b_r(y_r) = 1 & \forall r \\ \sum_{p \in P(r)} b_p(y_p) = b_r(y_r) & \forall r, p \in P(r), y_r \end{cases} . \tag{7}$$

Using the decomposition $F(y) = f_1(y_1) + f_2(y_2) + f_{1,2}(y_1, y_2)$, *i.e.*, for $r \in \{\{1\}, \{2\}, \{1, 2\}\}$, explicitly state the integer linear program and all its constraints for the special case that $\mathcal{Y}_i = \{0, 1\}$ for $i \in \{1, 2\}$. (**Hint:** The parent sets are as follows: $P(\{1\}) = \{1, 2\}$ and $P(\{2\}) = \{1, 2\}$. Use notation such as $f_1(y_1 = 0)$ and $b_1(y_1 = 0)$.)

(e) Let b be the vector

$$\begin{aligned} b = [\quad & b_1(y_1 = 0), b_1(y_1 = 1), b_2(y_2 = 0), b_2(y_2 = 1), \\ & b_{1,2}(y_1 = 0, y_2 = 0), b_{1,2}(y_1 = 1, y_2 = 0), b_{1,2}(y_1 = 0, y_2 = 1), b_{1,2}(y_1 = 1, y_2 = 1)]^\top . \end{aligned}$$

Specify all but the integrality constraints of part (d) using matrix vector notation, *i.e.*, provide A and c for $Ab = c$.

(f) Complete `A6_Structure.py` where we approximately solve the integer linear program using the linear programming relaxation. Implement the constraints. Why do we provide $-f$ as input to the solver? What is the obtained result b for the relaxation of the program given in Eq. (7) and its cost function value? Is this the configuration y^* which has the largest score?