CS 446 / ECE 449 — Homework 2

your NetID here

Instructions.

- Homework is due **Tuesday, Feb. 21, at noon CDT**; no late homework accepted.
- Everyone must submit individually at gradescope under `hw2` and `hw2code`.
- The "written" submission at `hw2` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw2`, gradescope will ask you to mark out boxes around each of your answers; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw2code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `hw2code`, only upload `hw2_ResNet.py`. Additional files will be ignored.
- Gradescope coding questions are marked by "`[code]`."

1. Expectation Maximization

In this problem, you will implement an expectation-maximization (EM) algorithm to cluster samples $\mathcal{D} = \{x^{(i)}\}_{i=1}^n$, with $x^{(i)} \in \{0,1\}^D$ into groups. You will be using a mixture of Bernoullis model to tackle this problem.

(a) **Mixture of Bernoullis**.

i. Assume each variable x_d in the D-dimensional vector x is drawn from a Bernoulli(q_d) distribution, $P(x_d = 1) = q_d$. Let $q = [q_1, \cdots, q_D] \in [0,1]^D$ be the resulting vector of Bernoulli parameters. Write an expression for $P(x|q)$ as a function of q_d and x_d.

ii. Now suppose we have a mixture of K Bernoulli distributions: each vector $x^{(i)}$ is drawn from some vector of Bernoulli random variables with parameters $p^{(k)} = [p_1^{(k)}, \cdots, p_D^{(k)}]$, that we call Bernoulli($p^{(k)}$). Let $p \triangleq \{p^{(1)}, \cdots, p^{(K)}\}$. Let π_k denote the probability that the k^{th} set of Bernoulli parameters is chosen. Assume a distribution $\pi \triangleq \{\pi_1, \ldots, \pi_K\}$ over all possible Bernoulli parameters. Write an expression for $P(x^{(i)}|p, \pi)$, as a function of π_k and $P(x^{(i)}|p^{(k)})$.

iii. Using the above, write an expression for the log-likelihood of i.i.d. data $\mathcal{D}$, i.e., $\log P(\mathcal{D}|\pi, p)$.

(b) **Expectation step**.

i. Let $z^{(i)} \triangleq [z_1^{(i)}, \ldots, z_K^{(i)}] \in \{0,1\}^K$ be an indicator vector, such that $z_k^{(i)} = 1$ if $x^{(i)}$ was drawn from a Bernoulli($p^{(k)}$), and 0 otherwise. Write down the expression of $P(z^{(i)}|\pi)$ as a function of π_k and $z_k^{(i)}$.

ii. Write down the expression of $P(x^{(i)}|z^{(i)}, p, \pi)$ as a function of $P(x^{(i)}|p^{(k)})$ and $z_k^{(i)}$.

iii. Let $Z = \{z^{(i)}\}_{i=1}^n$. Using the two quantities above, derive the likelihood of the data and latent variables $P(Z, \mathcal{D}|\pi, p)$.

iv. Let $\eta(z_k^{(i)}) = \mathbb{E}[z_k^{(i)}|x^{(i)}, \pi, p]$. Show that

$$\eta(z_k^{(i)}) = \frac{\pi_k \prod_{d=1}^D (p_d^{(k)})^{x_d^{(i)}} (1 - p_d^k)^{1 - x_d^{(i)}}}{\sum_j \pi_j \prod_{d=1}^D (p_d^{(j)})^{x_d^{(i)}} (1 - p_d^j)^{1 - x_d^{(i)}}}$$

v. Let $\tilde{p}$, $\tilde{\pi}$ be the new parameters that we would like to maximize. p, π are from the previous iteration. Use this to derive the following final expression for the E-step in the EM algorithm:

$$\mathbb{E}[\log P(Z, \mathcal{D}|\tilde{p}, \tilde{\pi})|\mathcal{D}, p, \pi] = \sum_{i=1}^n \sum_{k=1}^K \eta(z_k^{(i)}) \Big[\log \tilde{\pi}_k + \sum_{d=1}^D (x_d^{(i)} \log \tilde{p}_d^{(k)} + (1 - x_d^{(i)}) \log(1 - \tilde{p}_d^{(k)})) \Big]$$

(c) **Maximization step**. In the following, we will find $\tilde{p}$ and $\tilde{\pi}$ that maximize the above expression.

i. Show that $\tilde{p}$ that maximizes the E-step is:

$$\tilde{p}^{(k)} = \frac{\sum_{i=1}^n \eta(z_k^{(i)}) x^{(i)}}{N_k},$$

where $N_k = \sum_{i=1}^n \eta(z_k^{(i)})$.

ii. Prove that the value of $\tilde{\pi}$ that maximizes the E-step is:

$$\tilde{\pi}_k = \frac{N_k}{\sum_{k'} N_{k'}}.$$

Hint: $\tilde{\pi}$ needs to be a distribution. Use Lagrange multiplier to include this constraint into your objective function and solve it.

Solution.

2. Deep Net

We want to train a simple deep net $f(w_3, w_2, w_1, x) = w_3\sigma_2(\underbrace{w_2\sigma_1(\underbrace{w_1 x}_{x_1})}_{x_2})$ where $w_1, w_2, w_3, x \in \mathbb{R}$ are real valued and $\sigma_1(u) = \sigma_2(u) = \frac{1}{1+\exp(-u)}$ is the sigmoid function.

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

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

(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, and the channel dimension of the second convolution operation, assuming that padding is 0?

Solution.

3. ResNet.

In this problem, you will implement a simplified ResNet. You do not need to change arguments which are not mentioned here (but you of course could try and see what happens).

(a) `[code]` Implement a class `Block`, which is a building block of ResNet. It is described in Figure 2 of He et al. (2016), but also as follows.

The input to `Block` is of shape (N, C, H, W), where N denotes the batch size, C denotes the number of channels, and H and W are the height and width of each channel. For each data example $\boldsymbol{x}$ with shape (C, H, W), the output of `block` is

$$\texttt{Block}(\boldsymbol{x}) = \sigma_r\left(\boldsymbol{x} + f(\boldsymbol{x})\right),$$

where σ_r denotes the ReLU activation, and $f(\boldsymbol{x})$ also has shape (C, H, W) and thus can be added to $\boldsymbol{x}$. In detail, f contains the following layers.

i. A `Conv2d` with C input channels, C output channels, kernel size 3, stride 1, padding 1, and no bias term.
ii. A `BatchNorm2d` with C features.
iii. A ReLU layer.
iv. Another `Conv2d` with the same arguments as i above.
v. Another `BatchNorm2d` with C features.

Because 3×3 kernels and padding 1 are used, the convolutional layers do not change the shape of each channel. Moreover, the number of channels are also kept unchanged. Therefore $f(\boldsymbol{x})$ does have the same shape as $\boldsymbol{x}$.

Additional instructions are given in doscstrings in `hw2_ResNet.py`.

Library routines: `torch.nn.Conv2d and torch.nn.BatchNorm2d.`

Remark: Use `bias=False` for the `Conv2d` layers.

(b) `[code]` Implement a (shallow) `ResNet` consists of the following parts:

i. A `Conv2d` with 1 input channel, C output channels, kernel size 3, stride 2, padding 1, and no bias term.
ii. A `BatchNorm2d` with C features.
iii. A ReLU layer.
iv. A `MaxPool2d` with kernel size 2.
v. A `Block` with C channels.
vi. An `AdaptiveAvgPool2d` which for each channel takes the average of all elements.
vii. A `Linear` with C inputs and 10 outputs.

Additional instructions are given in doscstrings in `hw2_ResNet.py`.

Library routines: `torch.nn.Conv2d, torch.nn.BatchNorm2d, torch.nn.MaxPool2D, torch.nn.AdaptiveAvgPool2d and torch.nn.Linear.`

Remark: Use `bias=False` for the `Conv2d` layer.

(c) Train your `ResNet` implemented in (b) with different choices $C \in \{1, 2, 4\}$ on digits data and draw the training error vs the test error curves. To make your life easier, we provide you with the training script `utils_train_script.py` to load the digits data and train your ResNet with different choices for C. Therefore, you only need to plot the training and testing errors. Train your algorithms for 4000 epochs using SGD with mini batch size 128 and step size 0.1. See the docstrings in `hw2_ResNet.py` for more details. Include the resulting plot in your written handin.

For full credit, in addition to including the six train and test curves, include at least one complete sentence describing how the train and test error (and in particular their gap) change with C, which itself corresponds to a notion of model complexity as discussed in lecture.

Library routines: `torch.nn.CrossEntropyLoss, torch.autograd.backward, torch.no_grad, torch.optim.Optimizer.zero_grad, torch.autograd.grad, torch.nn.Module.parameters.`

(d) Train your `ResNet` implemented in (b) with $C = 64$ on digits data and draw the training error vs the test error curve. You may use the same training script provided for the previous question to train your ResNet with $C = 64$. Train your algorithms for 4000 epochs using SGD with mini batch size 128 and step size 0.1. See the docstrings in `hw2_ResNet.py` for more details. Include the resulting plot in your written handin.

For full credit, additionally include at least one complete sentence comparing the train and test error with those in part (c).

Library routines: `torch.nn.CrossEntropyLoss, torch.autograd.backward, torch.no_grad, torch.optim.Optimizer.zero_grad, torch.autograd.grad, torch.nn.Module.parameters.`

Solution.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.