
Deep Learning Assignment 3

Anonymous Author(s)

Affiliation

Address

email

1 General Questions

(a) Say if the first module is:

$$\max(W_1 X) \quad (1)$$

where the W input layer maybe doing summation and summation just like matrix mutiplication does WX , and the \max function is a non-linear active function modifying the valuse like a neuron does before entering the next module:

$$W_2(\max(W_1 X)) \quad (2)$$

If now we don't have the active function then the formula will looks like:

$$W_2(W_1 X) \rightarrow \bar{W} X \quad (3)$$

which eventually all W_i can become a single module \bar{W}

(b) For dictionary learning using sparse coding:

$$\min_{D, \alpha} \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{2} \|x_i - D\alpha\|^2 + \lambda \|\alpha_i\|_1 \right) \quad (4)$$

Which is a joint minimization problem with respect to dictionary D and α

And for autoencoders:

$$\min_{W_{de}, W_{en}} \frac{1}{n} \sum_{i=1}^n \|W_{de} \sigma(W_{en} x_i) - x_i\|^2 \quad (5)$$

Where σ is some non-linear function (e.g. shrinkage). Similar to dictionary learning, autoencoders is also a joint optimization problem respect to encoder and decoder matrix W_{en} , W_{de} . Indeed we can make the α to become $\sigma(W_{en} x)$ and this will make them very similar. However, there are two main factors making them different: One is autoencoder does not has the term for regulator, and therefore sparsity is not encouraged. Another one is autoencoder uses the model to find the code, while sparse coding approaching it by means of the optimizations.

2 Softmax regression gradient calculation

Given

$$\hat{y} = \sigma(Wx + b), \text{ where } x \in \mathbb{R}^d, W \in \mathbb{R}^{k \times d}, b \in \mathbb{R}^k \quad (6)$$

where d is the input dimension, k is the number of classes, σ is the softmax function:

$$\sigma(a)_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)} \quad (7)$$

Which means a given input x will output y with probability of each class

21 **(a)** Derive $\frac{\partial l}{\partial W_{ij}}$

22 If the given cross-entropy loss defined as followed:

$$l(y, \hat{y}) = - \sum_i y_i \log \hat{y}_i \quad (8)$$

23 As W_{ij} will affect the prediction of class i by multipling index j in x , therefore we can derive:

$$\frac{\partial l}{\partial W_{ij}} = \frac{\partial l}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial W_{ij}} \quad (9)$$

24 where:

$$l(y, \hat{y}) = - \sum_i y_i \log \hat{y}_i = -(y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \dots + y_i \log \hat{y}_i + \dots) \quad (10)$$

25 and therefore

$$\frac{\partial l}{\partial \hat{y}_i} = \frac{-y_i}{\hat{y}_i} \quad (11)$$

26 And we can rewrite for only for \hat{y}_i :

$$\hat{y}_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)} = \frac{\exp(a_i)}{C + \exp(a_i)}, \text{ where } C = \sum_{k \neq i} \exp(a_k) \quad (12)$$

27 Since

$$\frac{\partial \exp(a_i)}{\partial W_{ij}} = X_j \exp(a_i) \quad (13)$$

28 Therefore

$$\frac{\partial \hat{y}_i}{\partial W_{ij}} = X_j \hat{y}_i (1 - \hat{y}_i) \quad (14)$$

29 Finally, we will get the result of $\frac{\partial l}{\partial W_{ij}}$:

$$\frac{\partial l}{\partial W_{ij}} = \frac{\partial l}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial W_{ij}} = -X_j y_i (1 - \hat{y}_i) \quad (15)$$

30 **(b)** What happen when $y_{c_1} = 1, \hat{y}_{c_2} = 1, c_1 \neq c_2$

31 This means something like $y = [1, 0, 0]^T$ and $\hat{y} = [0, 0, 1]^T$, and the predict is far different from true
 32 label. This will cause the log part in loss (3) become negative infinity. We may not need to worry this
 33 because before one of the class predicted close to 1 and everything else close to 0, it will generate a
 34 great positive loss the the class that is miss-predicted trying to make the predict right to true label.

35 **3 Chain rule**

36 Without explicitly deriving the formula of $f(x, y)$, can we apply layers of functions to represent
 37 function f , which is similar to build deep learning architecture.

$$\begin{aligned} f &= \frac{x^2 + \sigma(y)}{3x + y - \sigma(x)} = \frac{a}{b} \\ \implies \frac{\partial f}{\partial x} &= \frac{\partial a}{\partial x} \frac{1}{b} - \frac{a}{b^2} \frac{\partial b}{\partial x} \\ \implies \frac{\partial f}{\partial y} &= \frac{\partial a}{\partial y} \frac{1}{b} - \frac{a}{b^2} \frac{\partial b}{\partial y} \\ \implies \frac{\partial a}{\partial x} &= 2x \\ \implies \frac{\partial a}{\partial y} &= \sigma(y)(1 - \sigma(y)) \\ \implies \frac{\partial b}{\partial x} &= 3 - \sigma(x)(1 - \sigma(x)) \\ \implies \frac{\partial b}{\partial y} &= 1 \end{aligned} \quad (16)$$

38 **(b)** As $x = 1$ and $y = 0$, then for each of value from the function listed above:

$$\begin{aligned}
 a &= 1 + \sigma(0) = 1.5 \\
 b &= 3 + 0 + \sigma(1) = 2.269 \\
 \frac{\partial a}{\partial x} &= 2 \cdot 1 = 2 \\
 \frac{\partial a}{\partial y} &= 0.5(1 - 0.5) = 0.25 \\
 \frac{\partial b}{\partial x} &= 3 - 0.731(1 - 0.731) = 2.803 \\
 \frac{\partial b}{\partial y} &= 1
 \end{aligned} \tag{17}$$

39 Therefore, applying each of the gradient at $(x, y) = (1, 0)$ to the chain rule, we will get:

$$\begin{aligned}
 \frac{\partial f}{\partial x} &= \frac{\partial a}{\partial x} \frac{1}{b} - \frac{a}{b^2} \frac{\partial b}{\partial x} = 2 \cdot \frac{1}{2.269} - \frac{1.5}{(2.269)^2} \cdot 2.803 = 0.0647 \\
 \frac{\partial f}{\partial y} &= \frac{\partial a}{\partial y} \frac{1}{b} - \frac{a}{b^2} \frac{\partial b}{\partial y} = 0.25 \cdot \frac{1}{2.269} - \frac{1.5}{(2.269)^2} \cdot 1 = -0.1811
 \end{aligned} \tag{18}$$

40 **4 Variants of pooling**

41 **5 Convolution**

42 **(a)** As it is using 3x3 kernel along x and y axis of input, which is 5 and 5 respectively. The output
 43 of this layer will be $(5 - 3 + 1) \times (5 - 3 + 1)$ which is 3x3.

44

45 **(b)** Assuming the kernel operation is point-point multiplication and summation, then the output of
 46 this layer is:

$$\begin{pmatrix} 109 & 92 & 72 \\ 108 & 85 & 74 \\ 110 & 74 & 79 \end{pmatrix}$$

48

$$\begin{pmatrix} 4 & 7 & 10 & 6 & 3 \\ 9 & 17 & 25 & 16 & 8 \\ 11 & 23 & 34 & 23 & 11 \\ 7 & 16 & 24 & 17 & 8 \\ 2 & 6 & 9 & 7 & 3 \end{pmatrix}$$

50

51 **6 Optimization**

52 **(a)** say the encoder and decoder is defined as:

$$\begin{aligned}
 z &= W_1 x + b_1 \\
 \tilde{x} &= W_2 z + b_2
 \end{aligned} \tag{19}$$

53 And therefore the reconstruction loss J will be:

$$J(W_1, b_1, W_2, b_2) = (\tilde{x} - x)^2 = (W_2(W_1 x + b_1) + b_2 - x)^2 \tag{20}$$

54 **(b)** To have the gradient of reconstruction loss respective to the parameters, we take the derivative
 55 of each parameters:

$$\begin{aligned}
 \frac{\partial J}{\partial W_1} &= W_2 x \\
 \frac{\partial J}{\partial W_2} &= W_1 x + b_1
 \end{aligned} \tag{21}$$

56 (c) Say now we are at stage t and would like to compute W_1^{t+1} and W_2^{t+1} :

$$\begin{aligned} W_1^{t+1} &= W_1^t - \mu_1^t \frac{\partial J}{\partial W_1^t} = W_1^t - \mu_1^t (W_2^t x) \\ W_2^{t+1} &= W_2^t - \mu_2^t \frac{\partial J}{\partial W_2^t} = W_2^t - \mu_2^t (W_1^t x + b_1) \end{aligned} \quad (22)$$

57 where μ_1^t and μ_2^t are the step size at stage t

58 (d) The updates during stochastic gradient descent usually involves Move-Forward and Correction
59 stages and this oscillation may delay the efficiency of convergence, and therefore adding a momentum
60 term may make the update toward the good direction as well as with the previous update history
61 considered:

$$\begin{aligned} W_1^{t+1} &= W_1^t - \mu_1^t \frac{\partial J}{\partial W_1^t} + \Delta W_1^t \\ W_2^{t+1} &= W_2^t - \mu_2^t \frac{\partial J}{\partial W_2^t} + \Delta W_2^t \end{aligned} \quad (23)$$

62 7 Top-k error

63 For image classification, sometime the class is ambiguous, and the loss during is being modified to
64 consider multiple label. The top-k error rate is the fraction of test images for which the correct label
65 is not among the top-k labels considered most probable. The reason why ImageNet using both top-5
66 and top-1 is due to sometimes only looking at top-1 error cannot be objective enough to evaluate the
67 model because the image itself contains multi-label, and therefore evaluating top-5 error is important
68 too.

69 8 t-SNE

70 9 Proximal gradient descent

71 (a) Since Proximal operator is defined as:

$$prox_{h,t}(x) = argmin_z \frac{1}{2} \|z - x\|_2^2 + th(z) \quad (24)$$

72 which the optimal condition is to have the gradient w.r.t z equal to 0:

$$0 \in z - x + t\partial h(z) \quad (25)$$

73 if function $h(z) = \|z\|_1$ and $z_i \neq 0$, then:

$$\partial h(z) = sign(z) \quad (26)$$

74 And therefore the optimal solution z^* will be:

$$z^* = x - t \cdot sign(z^*) \quad (27)$$

75 Noted that if $z_i^* < 0$, then $x_i < -t$, and if $z_i^* > 0$, then $x_i > t$. This implies $|x_i| > t$ and
76 $sign(z_i^*) = sign(x_i)$, and we can rewrite formula to:

$$z_i^* = x_i - t \cdot sign(x_i) \quad (28)$$

77 Then if the solution $z_i^* = 0$, the subgradient of l1-norm is in the interval of $[-1, 1]$, and we can write:

$$0 \in -x_i + t \cdot [-1, 1] \implies x_i \in [-t, t] \implies |x_i| \leq t \quad (29)$$

78 Therefore the solution of Proximal operator will be:

$$z_i^* = \begin{cases} 0 & \text{if } |x_i| \leq t \\ x_i - t \cdot sign(x_i) & \text{if } |x_i| > t \end{cases} \quad (30)$$

79 which is

$$prox_{h,t}(x) = S_t(x) = (|x| - t)_+ \odot sign(x) \quad (\text{element-wise}) \quad (31)$$

80 which is a soft-threshold function with t as threshold value

81

82 **(b)** In the field of signal processing, the true signal usually will be blurred as followed:

$$Ax = b \quad (32)$$

83 where A is the blur operation, b is the known observed blurred-signal. The way to solve true signal x
84 is called deblurring problem:

$$\min_x \{F(x) \equiv \frac{1}{2} \|b - Ax\|_2^2 + \lambda \|x\|_1\} \quad (33)$$

85 This is ISTA problem, and as we can see the first term is convex and differentiable, and the second
86 term is convex and simple l1-norm function. Then the ISTA is become one example of proximal
87 gradient descent
88

89 **(c)** From the definition of Proximal operator the optimal solution is where $\frac{\partial \text{prox}_{h,t}}{\partial z} = 0$, and
90 therefore we will have:

$$0 \in z - x + t\partial h(z) \quad (34)$$

91 After we rewrite the function and replace z by u which is the optimal result from Proximal function:

$$\frac{x - u}{t} \in \partial h(u) \quad (35)$$

92 which means the calculated result from proximal function will be within the interval proportional to
93 the subgradient of the simple-nonDerentiable function $h(x)$
94

95 **(d)** From definition of Proximal operator, the optimal solution x_{k+1} will be:

$$x_{k+1} = \text{prox}_{h,\alpha_k}(x_k - \alpha_k \nabla g(x_k)) = x_k - \alpha_k \nabla g(x_k) - \alpha_k \partial h(x_{k+1}) \quad (36)$$

96 and from definition:

$$G_{\alpha_k}(x_k) = \frac{x_k - \text{prox}_{h,\alpha_k}(x_k - \alpha_k \nabla g(x_k))}{\alpha_k} \quad (37)$$

97 after rewrite:

$$x_k - \alpha_k \nabla g(x_k) - \alpha_k \partial h(x_{k+1}) = x_k - \alpha_k G_{\alpha_k}(x_k) \quad (38)$$

98 Therefore

$$G_{\alpha_k}(x_k) - \nabla g(x_k) \in \partial h(x_{k+1}) \quad (39)$$

99 which is because h is not differentiable and the result will within the range of subgradient of $\partial h(x_{k+1})$