Course Materials for GEN-AI

Northeastern University

These materials have been prepared and sourced for the course **GEN-AI** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

Instructor: Ramin Mohammadi
r.mohammadi@northeastern.edu

Thank you for your understanding and collaboration.

Content sourced from multiple references - see Reference section for details

Homework - 1

Problem 1 (5 points)

Find the derivative of the softmax function.

Problem 2 (10 points)

Consider a model where the prior distribution over the parameters is a normal distribution with mean zero and variance σ^2, so that

$$Pr(\phi) = \prod_{j=1}^{J} \text{Norm}(0, \sigma_j^2; \phi_j),$$

where j indexes the model parameters. We now maximize $\prod_{i=1}^{n} Pr(y_i \mid x_i, \phi) Pr(\phi)$. Show that the associated loss function of this model is equivalent to L2 regularization.

Problem 3 (10 points)

Show that the weight decay parameter update with decay rate λ:

$$\phi \leftarrow (1-\lambda)\phi - \alpha \frac{\partial L}{\partial \phi},$$

on the original loss function $L[\phi]$ is equivalent to a standard gradient update using L2 regularization so that the modified loss function $\tilde{L}[\phi]$ is:

$$\tilde{L}[\phi] = L[\phi] + \frac{\lambda}{2\alpha} \sum_k \phi_k^2,$$

where ϕ are the parameters, and α is the learning rate.

Problem 4 (8 points)

A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size three, stride one, and dilation one is applied. What size is the receptive field of the hidden units in the third layer?

Problem 5 (8 points)

A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size seven, stride one, and dilation one is applied. What size is the receptive field of hidden units in the third layer?

Problem 6 (8 points)

Consider a convolutional network with 1D input x. The first hidden layer H_1 is computed using a convolution with kernel size five, stride two, and a dilation rate of one. The second hidden layer H_2 is computed using a convolution with kernel size three, stride one, and a dilation rate of one. The third hidden layer H_3 is computed using a convolution with kernel size five, stride one, and a dilation rate of two. What are the receptive field sizes at each hidden layer?

Problem 7 (12 points)

A surface is guaranteed to be convex if the eigenvalues of the Hessian $H[\phi]$ are positive everywhere. In this case, the surface has a unique minimum, and optimization is easy. Find an algebraic expression for the Hessian matrix,

$$H[\phi] = \begin{bmatrix} \frac{\partial^2 L}{\partial \phi_0^2} & \frac{\partial^2 L}{\partial \phi_0 \partial \phi_1} \\ \frac{\partial^2 L}{\partial \phi_1 \partial \phi_0} & \frac{\partial^2 L}{\partial \phi_1^2} \end{bmatrix},$$

for the linear regression model. Prove that this function is convex by showing that the eigenvalues are always positive. This can be done by showing that both the trace and the determinant of the matrix are positive.

Linear Regression:

Consider applying gradient descent to the 1D linear regression model. The model $f(x_i, \phi)$ maps a scalar input x to a scalar output y and has parameters $\phi = (\phi_0, \phi_1)^T$, which represent the y-intercept and the slope:

$$y = f(x_i, \phi) = \phi_0 + \phi_1 x_i.$$

Given a dataset $\{(x_i, y_i)\}$ containing I input/output pairs, we choose the least squares loss function:

$$L[\phi] = \sum_{i=1}^{I} (f(x_i, \phi) - y_i)^2 = \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2,$$

Problem 8 (12 points)

The logistic regression model uses a linear function to assign an input to one of two classes $y \in \{0, 1\}$. For a 1D input and a 1D output, it has two parameters, ϕ_0 and ϕ_1, and is defined by:

$$Pr(y = 1|x) = \text{sig}(\phi_0 + \phi_1 x),$$

where sig is the logistic sigmoid function:

$$\text{sig}(z) = \frac{1}{1 + \exp(-z)}.$$

(i) Plot y against x for this model for different values of ϕ_0 and ϕ_1 and explain the qualitative meaning of each parameter.

(ii) What is a suitable loss function for this model?

(iii) Compute the derivatives of this loss function with respect to the parameters.

(iv) Generate ten data points from a normal distribution with mean -1 and standard deviation 1 and assign them the label $y = 0$. Generate another ten data points from a normal distribution with mean 1 and standard deviation 1 and assign these the label $y = 1$. Plot the loss as a heatmap in terms of the two parameters ϕ_0 and ϕ_1.

(v) Is this loss function convex? How could you prove this?

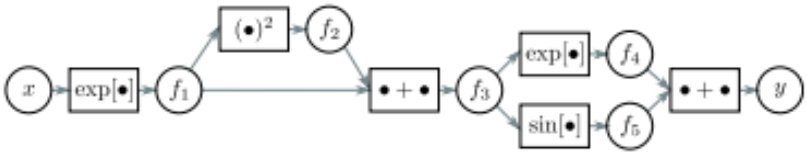


Figure 1: Image for question 9 and 10

Problem 9 (15 points)

This problem explores computing derivatives on general acyclic computational graphs. Consider the function:

$$y = \exp(\exp[x] + \exp[x]^2) + \sin(\exp[x] + \exp[x]^2).$$

We can break this down into a series of intermediate computations so that:

$$\begin{aligned} f_1 &= \exp[x], \\ f_2 &= f_1^2, \\ f_3 &= f_1 + f_2, \\ f_4 &= \exp[f_3], \\ f_5 &= \sin[f_3], \\ y &= f_4 + f_5. \end{aligned}$$

The associated computational graph is depicted in figure 1. Compute the derivative $\frac{\partial y}{\partial x}$ by reverse-mode differentiation. In other words, compute in order:

$$\frac{\partial y}{\partial f_5}, \frac{\partial y}{\partial f_4}, \frac{\partial y}{\partial f_3}, \frac{\partial y}{\partial f_2}, \frac{\partial y}{\partial f_1}, \text{ and } \frac{\partial y}{\partial x},$$

using the chain rule in each case to make use of the derivatives already computed.

Problem 10 (15 points)

For the same function as in problem 9, compute the derivative $\frac{\partial y}{\partial x}$ by forward-mode differentiation. In other words, compute in order:

$$\frac{\partial f_1}{\partial x}, \frac{\partial f_2}{\partial x}, \frac{\partial f_3}{\partial x}, \frac{\partial f_4}{\partial x}, \frac{\partial f_5}{\partial x}, \text{ and } \frac{\partial y}{\partial x},$$

using the chain rule in each case to make use of the derivatives already computed. Why do we not use forward-mode differentiation when we calculate the parameter gradients for deep networks?

Problem ① - Find the derivative of SOFTmax

For a vector $z = (z_1, z_2, \cdots, z_n)$ the softmax function can be defined as:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

For each element $s_i = \text{softmax}(z)_i$ represents the probability assigned to class "i".

Each output in a neural network using softmax as a classification function will depend on the inputs, that's why softmax's derivative is not a simple vector but a jacobian Matrix → $J_{ij} = \frac{\partial s_i}{\partial z_j}$ ∫ Probabilidad del elemento pertenezca a la clase I

↳ ~~p/ la probabilidad~~ logits.

This derivative has 2 cases

↳ (A) when $i = j$ // Diagonal

$$\frac{\partial s_i}{\partial z_i} = s_i \cdot (1 - s_i)$$

↳ (B) when $i \neq j$ // outside diagonal

$$\frac{\partial s_i}{\partial z_i} = -s_i \cdot s_j$$

Writting both cases:

$$\frac{\partial s_i}{\partial z_j} = s_i \cdot (\delta_{ij} - s_j) \quad \text{where} \quad \delta_{ij} = 1 \text{ if } i = j, \quad \delta_{ij} = 0 \text{ if } i \neq j$$

So,

$J = \text{diag}(s) - s.s^T$

$\text{diag}(s)$ = diagonal matrix with $[s_1, s_2 \dots s_n]$ in the diagonal

and $s.s^T$ is the outer product.

Problem 2 (10 points)

Consider a model where the prior distribution over the parameters is a normal distribution with mean zero and variance σ^2, so that

$$Pr(\phi) = \prod_{j=1}^{J} \text{Norm}(0, \sigma_j^2; \phi_j),$$

where j indexes the model parameters. We now maximize $\prod_{i=1}^{n} Pr(y_i \mid x_i, \phi) Pr(\phi)$. Show that the associated loss function of this model is equivalent to L2 regularization.

To avoid overFitting, we penalized big weights. In order to do that, two approaches can be used

- ↳ Regularization approach (L2): a penalty term is added to the loss function.

  $\text{Loss} = \text{Loss}_0 + \lambda (\|\Phi\|)^2$

  this term penalizes greater weights.

- ↳ Bayesian Approach: Assuming that the Φ comes from a normal distribution centered at zero.

  Prior: $Pr(\Phi) = \text{Norm}(0; \sigma^2; \Phi)$

With this Formula

$$Pr(\phi) = \prod_{j=1}^{J} \text{Norm}(0; \sigma_j^2; \phi_j)$$

and also considering that we want to maximize

$$\prod_{i=1}^{n} Pr(y_i | x_i, \phi) . Pr(\phi)$$

Taking logarithm (MAP)

$$\log \prod_{i=1}^{n} Pr(y_i|x_i, \phi) + \log Pr(\phi)$$

$$\downarrow$$

$$= \sum_{i=1}^{n} \log Pr(y_i|x_i, \phi) + \log \prod_{J=1}^{J} Norm(0, \sigma_J^2; \phi_j)$$

Expanding the prior gaussian:

$$\log Norm(0, \sigma_j^2; \phi_J) = \log \left(\frac{1}{\sqrt{2\pi\sigma_J^2}} \cdot e^{-\frac{\phi_J^2}{2\sigma_j^2}} \right)$$

$$= -\frac{\phi_J^2}{2\sigma_j^2} - 1/2 \log(2\pi\sigma_j^2)$$

Summing up the parameters

$$\log Pr(\phi) = \sum_{j=1}^{J} \left(-\frac{\phi^2 j}{2\sigma^2 j} \right) + \text{constant}$$

The complete MAP objective is:

$$\sum_{i=1}^{n} \log . Pr(y_i|x_i, \phi) - \sum_{J=1}^{J} \frac{\phi^2 j}{2\sigma^2 j}$$

$$\downarrow$$

considering a minimization problem

$$\mathcal{L} = -\sum_{i=1}^{n} \log \Pr(y_i | x_i, \phi) + \sum_{J=1}^{J} \phi_J^2 / 2\sigma_J^2$$

$$= \underbrace{-\sum_{i=1}^{n} \log \Pr(y_i | x_i, \phi)}_{\text{this is a negative log-likelihood + } L_2 \text{ regularization with } \lambda = \frac{1}{2\sigma^2}} + \frac{1}{2\sigma^2} \| \phi \|^2$$

Problem 3 (10 points)

Show that the weight decay parameter update with decay rate λ:

$$\phi \leftarrow (1 - \lambda)\phi - \alpha \frac{\partial L}{\partial \phi},$$

on the original loss function $L[\phi]$ is equivalent to a standard gradient update using L2 regularization so that the modified loss function $\tilde{L}[\phi]$ is:

$$\tilde{L}[\phi] = L[\phi] + \frac{\lambda}{2\alpha} \sum_k \phi_k^2,$$

where ϕ are the parameters, and α is the learning rate.

we have two ways of regularizing

↳ ① weight decay

$$\phi \leftarrow (1 - \underbrace{\lambda}_{\text{decay rate}}) \cdot \phi - \alpha \cdot \frac{\partial L}{\partial \phi}$$

At each iteration, we multiply the weights * $(1 - \lambda)$ before substracting the gradient.

↳ ② L_2 regularization.

$$\tilde{L}[\phi] = L[\phi] + \frac{\lambda}{2\alpha} \sum_K \phi_K^2$$

we add a penalty term to the loss and gradient descent is used on this new loss.

so, considering the standard gradient with L_2

↳ $\varphi \leftarrow \varphi - \alpha \cdot \partial \tilde{L} / \partial \varphi$

we compute the $\tilde{L}$ gradient

↳ $\frac{\partial \tilde{L}}{\partial \varphi} = \frac{\partial L}{\partial \varphi} + \left(\frac{\lambda}{2\alpha}\right) \cdot 2\varphi = \frac{\partial L}{\partial \varphi} + (\lambda/2)\varphi$

Sustituying in the update

↳ $\varphi \leftarrow \varphi - \alpha \cdot [\partial L / \partial \varphi + (\lambda/\alpha)\varphi]$

$\varphi \leftarrow \varphi - \alpha \cdot \frac{\partial L}{\partial \varphi} - \lambda\varphi$

$\boxed{\varphi \leftarrow (1-\lambda)\varphi - \alpha \cdot \frac{\partial L}{\partial \varphi}}$ // weight decay.

Problem 4 (8 points)

A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size three, stride one, and dilation one is applied. What size is the receptive field of the hidden units in the third layer?

Receptive field → we want to compute how many pixeles from the original image a neuron in the 3rd convolutional layer "sees".

↓

Kernel size = 3

stride "s" = 1

dilation (spacing between kernel elements) = 1

Padding "p" = 0

The Formula for calculating RF for a single convolutional layer → $RF_{layer} = \underbrace{K}_{\text{Kernel}} + \underbrace{(K-1) \times (d-1)}_{\text{dilation effect}}$

Replacing with dilation = 1 :

$RF\_layer = K + (K-1) \cdot (0) = K = 3$

Layer 1

$RF\_layer\,1 = 3$

Layer 2

$RF_2 = RF_1 + (K-1) \cdot stride_1$

$RF_2 = 3 + (3-1) \cdot 1$

$RF_2 = 5$

Layer 3

$RF_3 = RF_2 + (K-1) \cdot stride_2$

$RF_3 = 5 + (3-1) \cdot 1$

$RF_3 = 7$ // The receptive field of units in the 3rd layer is 7 (each neuron in layer 3 sees and is influenced by 7 consecutive pixels from the original input).

Problem 5 (8 points)

A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size seven, stride one, and dilation one is applied. What size is the receptive field of hidden units in the third layer?

Kernel size = 7
stride = 1
dilation = 1
padding = zero-padded

Computing RF

- $RF_1 = 7$ // each neuron in layer 1 sees 7 pixels.

- $RF_2 = RF_1 + (K-1) \cdot \text{stride}_1$
$RF_2 = 7 + (7-1) \cdot 1$
$RF_2 = 13$

- $RF_3 = RF_2 + (K-1) \cdot \text{stride}_2$
$RF_3 = 13 + (7-1) \cdot 1$
$RF_3 = 19$ // The receptive field of units in the 3rd layer is 19.

Problem 6 (8 points)

Consider a convolutional network with 1D input x. The first hidden layer H_1 is computed using a convolution with kernel size five, stride two, and a dilation rate of one. The second hidden layer H_2 is computed using a convolution with kernel size three, stride one, and a dilation rate of one. The third hidden layer H_3 is computed using a convolution with kernel size five, stride one, and a dilation rate of two. What are the receptive field sizes at each hidden layer?

H_1: Kernel = 5 stride = 2 dilation = 1

H_2: Kernel = 3 stride = 1 dilation = 1

H_3: Kernel = 5 stride = 1 dilation = 2

input $\longrightarrow$ $RF_0 = 1$

$stride_0 = 1$

$H_1 \longrightarrow$ Effective Kernel size

$$K\_effective_1 = K + (K-1)\cdot(d-1)$$
$$= 5 + (5-1)\cdot(1-1)$$
$$= 5$$

RF_1

$$RF_1 = RF_0 + (K\_effective_1 - 1)\cdot stride_0$$
$$= 1 + (5-1)\cdot 1$$
$$= 5$$

$RF_1 \longrightarrow 5$

Layer 2 - H_2

Effective Kernel size 2

$$K\_effective_2 = K + (K-1) \cdot (d-1)$$
$$= 3 + (3-1) \cdot (1-1)$$
$$= 3$$

RF_2

$$RF_2 = RF_1 + (K\_effective_2 - 1) \cdot stride_1$$
$$= 5 + (3-1) \cdot 2$$
$$= 9$$

$RF_2 = 9$ //

Layer 3 → H_3

Effective Kernel size H_3

$$K\_effective_3 = K + (K-1) \cdot (d-1)$$
$$= 5 + (5-1) \cdot (2-1)$$
$$= 9$$

RF_3

$$RF_3 = RF_2 + (K\_effective_3 - 1) \cdot stride_2$$
$$RF_3 = 9 + (9-1) \cdot 1$$

$RF_3 = 17$

$RF_1 = 5$
$RF_2 = 9$
$RF_3 = 17$ //

Problem 7 (12 points)

A surface is guaranteed to be convex if the eigenvalues of the Hessian $H[\phi]$ are positive everywhere. In this case, the surface has a unique minimum, and optimization is easy. Find an algebraic expression for the Hessian matrix,

$$H[\phi] = \begin{bmatrix} \frac{\partial^2 L}{\partial \phi_0^2} & \frac{\partial^2 L}{\partial \phi_0 \partial \phi_1} \\ \frac{\partial^2 L}{\partial \phi_1 \partial \phi_0} & \frac{\partial^2 L}{\partial \phi_1^2} \end{bmatrix},$$

linear regression

for the linear regression model. Prove that this function is convex by showing that the eigenvalues are always positive. This can be done by showing that both the trace and the determinant of the matrix are positive.

Linear Regression:

Consider applying gradient descent to the 1D linear regression model. The model $f(x_i, \phi)$ maps a scalar input x to a scalar output y and has parameters $\phi = (\phi_0, \phi_1)^T$, which represent the y-intercept and the slope:

$$y = f(x_i, \phi) = \phi_0 + \phi_1 x_i.$$

Given a dataset $\{(x_i, y_i)\}$ containing I input/output pairs, we choose the least squares loss function:

$$L[\phi] = \sum_{i=1}^{I} (f(x_i, \phi) - y_i)^2 = \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2,$$

For this problem I have to proof that the Loss function of linear regression is convex, that means that the surface is convex and positive in both ways so it is easily optimizable and has a unique solution.

Model $y = \underbrace{\varphi_0}_{\text{interc.}} + \underbrace{\varphi_1 x_i}_{\text{slope * input}}$

Parameters $\varphi = (\varphi_0, \varphi_1)^T$

Loss Function of linear regression $L(\varphi) = \sum_{i=1}^{I} (\varphi_0 + \varphi_1 x_i - y_i)^2$ // sum of squares errors

First derivatives

* with respect to φ_0

$$\frac{\partial L}{\partial \varphi_0} = \frac{\partial}{\partial \varphi_0} \cdot [\Sigma_i (\varphi_0 + \varphi_1 x_i - y_i)^2]$$

Applying chain rule

$$= \Sigma_i 2(\varphi_0 + \varphi_1 x_i - y_i) \cdot \underbrace{\frac{\partial}{\partial \varphi_0}(\varphi_0 + \varphi_1 x_i - y_i)}_{=1}$$

$$\frac{\partial L}{\partial \varphi_0} = 2 \sum_i (\varphi_0 + \varphi_1 x_i - y_i) \quad // \text{ residual error}$$

* with respect to φ_1

$$\frac{\partial L}{\partial \varphi_1} = \frac{\partial}{\partial \varphi_1} \left[\sum_i (\varphi_0 + \varphi_1 x_i - y_i)^2 \right]$$

chain rule:

$$\frac{\partial L}{\partial \varphi_1} = \Sigma_i 2(\varphi_0 + \varphi_1 x_i - y_i) \cdot \left(\frac{\partial}{\partial \varphi_1} (\varphi_0 - \varphi_1 x_i - \right.$$

$$\left. x_i \right.$$

$$\frac{\partial L}{\partial \varphi_1} = \underbrace{\sum_i 2(\varphi_0 + \varphi_1 x_i - y_i) \cdot x_i}_{\text{residual} * \text{input}}$$

Now, I have to compute the 2nd derivatives since these are the Hessian Elements that ∋ to the Hessian Matrix.

- $H[1,1] : \frac{\partial^2 \ell}{\partial \theta_0^2}$

$$\frac{\partial^2 L}{\partial \theta_0^2} = \frac{\partial}{\partial \theta_0}\left[2\Sigma_i(\theta_0 + \theta_1 x_i - y_i)\right]$$

$$= 2\Sigma_i \frac{\partial}{\partial \theta_0}(\theta_0 + \theta_1 x_i - y_i)$$

$$= 2\Sigma_i \cdot 1$$

$$= 2I \quad // \text{ 2 times the number of data}$$

since $I > 0 \rightarrow$ Positive ✓

- $H[2,2] : \frac{\partial^2 \ell}{\partial^2 \theta_1}$

$$\frac{\partial^2 L}{\partial \theta_1^2} = \frac{\partial}{\partial \theta_1}\left[2\Sigma_i(\theta_0 + \theta_1 x_i - y_i) \cdot x_i\right]$$

$$= 2\Sigma_i x_i \cdot \frac{\partial}{\partial \theta_1}(\theta_0 + \theta_1 x_i - y_i)$$

$$= 2\Sigma_i x_i \cdot x_i$$

$$= 2\Sigma_i x_i^2 \quad // \text{ 2 times the sum of inputs}^2$$

since $x_i^2 \geq 0$ always $\rightarrow$ Positive ✓

- $H[1,2] = H[2,1] = \frac{\partial^2 \ell}{\partial \theta_0 \partial \theta_1}$

$$\frac{\partial^2 L}{\partial \theta_0 \theta_1} = \frac{\partial}{\partial \theta_0}\left[2\Sigma_i(\theta_0 + \theta_1 x_i - y_i)\cdot x_i\right]$$

$$= 2\Sigma_i x_i \cdot \frac{\partial}{\partial \theta_0}(\theta_0 + \theta_1 x_i - y_i)$$

$$= 2\Sigma_i x_i \cdot 1$$

$$= 2\Sigma_i x_i \quad // \quad 2 \times \text{sum of the inputs}$$

Hessian Matrix

$$H = \begin{bmatrix} 2I & 2\Sigma_i x_i \\ 2\Sigma_i x_i & 2\Sigma_i x_i^2 \end{bmatrix} = 2\begin{bmatrix} I & \Sigma_i x_i \\ \Sigma_i x_i & \Sigma_i x_i^2 \end{bmatrix}$$

Now, I have to prove positive eigenvalues (Trace method). (sum of diagonal elements)

$\text{Trace}(H) = 2(I + \Sigma_i x_i^2) > 0$ ✓

$I > 0$ (we have datapoints) → trace is positive

$\text{Trace} = \lambda_1 + \lambda_2 > 0$ ✓

For the 2nd condition : prove positive eigenvalues (Determinant)

└ Determinant = (diagonal product) − (OFF diagonal product)^2

$$\text{Det}(H) = (2I)(2\Sigma_i x_i^2) - (2\Sigma_i x_i)^2$$

$$= 4I \cdot \Sigma_i x_i^2 - 4(\Sigma_i x_i)^2$$

$$= 4\underbrace{\left[I \cdot \Sigma_i x_i^2 - (\Sigma_i x_i)^2\right]}$$

I have to show that this is > 0
(Cauchy - Inequality)

IF I consider real data :

$$I \cdot \Sigma_i x_i^2 - (\Sigma_i x_i)^2 > 0$$

Therefore :

$$\text{Det}(H) = 4\left[I \cdot \Sigma_i x_i^2 - (\Sigma_i x_i)^2\right] > 0 \quad \checkmark$$

| is positive

IF det > 0 AND trace > 0, both eigenvalues > 0

Therefore H is positive definite $\therefore$ $L(\phi)$ is convex $\therefore$
Linear regression has a unique global minimum.

Problem 8 (12 points)

The logistic regression model uses a linear function to assign an input to one of two classes $y \in \{0, 1\}$. For a 1D input and a 1D output, it has two parameters, ϕ_0 and ϕ_1, and is defined by:

$$Pr(y = 1|x) = \text{sig}(\phi_0 + \phi_1 x),$$

where sig is the logistic sigmoid function:

$$\text{sig}(z) = \frac{1}{1 + \exp(-z)}.$$

(i) Plot y against x for this model for different values of ϕ_0 and ϕ_1 and explain the qualitative meaning of each parameter.

(ii) What is a suitable loss function for this model?

(iii) Compute the derivatives of this loss function with respect to the parameters.

(iv) Generate ten data points from a normal distribution with mean -1 and standard deviation 1 and assign them the label $y = 0$. Generate another ten data points from a normal distribution with mean 1 and standard deviation 1 and assign these the label $y = 1$. Plot the loss as a heatmap in terms of the two parameters ϕ_0 and ϕ_1.

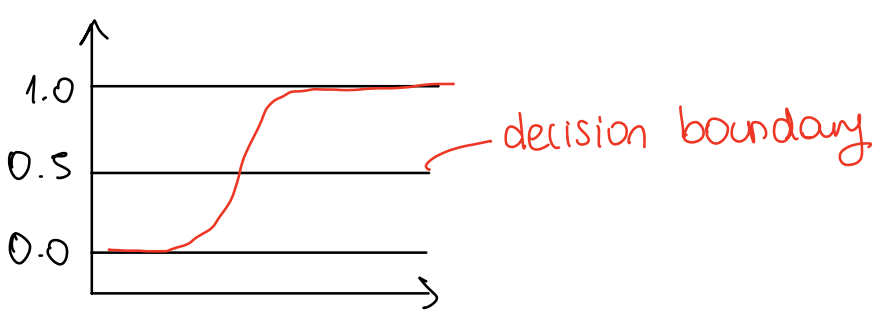
(v) Is this loss function convex? How could you prove this?

① Model $\quad Pr(y=1|x) = sig(\phi_0 + \phi_1 x)$

where $\quad sig(z) = 1/(1 + exp(-z))$

Parameters
- ϕ_0 = bias (shifts decision boundary)
- ϕ_1 = slope (transition)

i)



Parameters Effects
- $\phi_0 = 0$ centered the curve at $x=0$
- $\phi_0 > 0$ curve shifts left.
- $\phi_0 < 0$ curve shifts right

↳ { θ_1 controls curve steepness
θ_1 large → steep transition, ergo, confident predictions
θ_1 small → gradual transition
$\theta_1 < 0$ → Flipped curve

ii) suitable Loss function

↳ Log-loss (binary cross entropy) since measures how well predicted probabilities match the labels, penalizes confident wrong predictions and is standard for binary classification as well as convex for logistic regression.

iii) Derivative of Log-Loss

Considering the sigmoid property

$$\frac{\partial}{\partial z}[sig(z)] = sig(z) \cdot (1 - sig(z))$$

Let $pr_i = sig(\phi_0 + \phi_1 x_i)$

. Derivative with respect to ϕ_0:

$$L = -\Sigma_i [y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)]$$

chain rule to each term

$$\frac{\partial}{\partial \phi_0} [y_i \cdot \log(p_i)] = y_i \cdot \left(\frac{1}{p_i}\right) \cdot \frac{\partial p_i}{\partial \phi_0}$$

Find $\partial p_i / \partial \phi_0$

$$\partial p_i / \partial \phi_0 = p_i(1-p_i) \cdot 1 = p_i \cdot (1-p_i)$$

1st term $y_i \cdot \left(\frac{1}{p_i}\right) \cdot p_i(1-p_i) = y_i \cdot (1-p_i)$

2nd term $\rightarrow -(1-y_i)\, p_i$

Combining

$$\frac{\partial L}{\partial \theta_0} = -\Sigma_i \left[y_i (1-p_i) - (1-y_i) p_i \right]$$

$$\frac{\partial L}{\partial \theta_0} = -\Sigma_i \left[y_i - p_i \right]$$

$$\boxed{\frac{\partial L}{\partial \theta_0} = \Sigma_i (p_i - y_i)}$$

Derivative w.r.t θ_1:

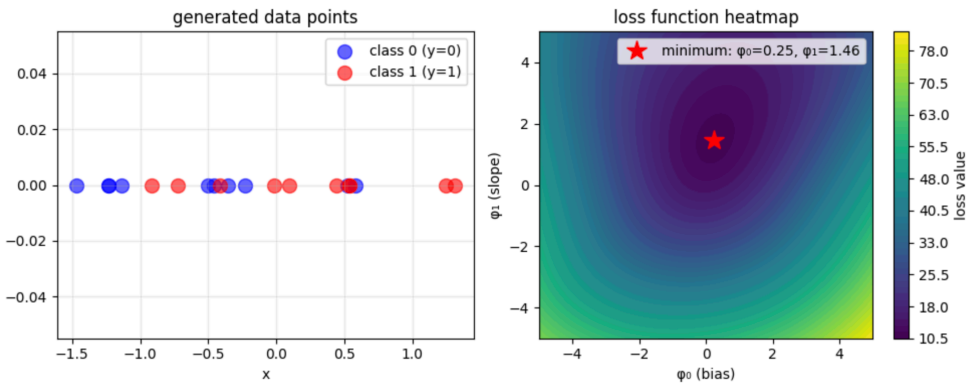
Same than before but:

$$\frac{\partial p_i}{\partial \theta_1} = p_i (1-p_i) \cdot x_i$$

Following same steps

$$\frac{\partial L}{\partial \theta_1} = \Sigma_i (p_i - y_i) \cdot x_i$$

IV



(V) As we can see from the heapmap, the Loss Function is convex($\theta_0 = 0.25$, $\theta_1 = 1.46$; I did all this with code). This can be proven by showing that the Hessian matrix is positive semi-definite. The 2nd derivatives $\frac{\partial^2 L}{\partial \theta_0^2} = \sum_i p_i(1-p_i)$ and $\frac{\partial^2 L}{\partial \theta_1}^2 =$

$\sum_i p_i(1-p_i) \cdot x_i^2$ are always positive since $0 < p_i < 1$ for sigmoid. Therefore the loss has a unique global minimum and gradient descent will always converge.

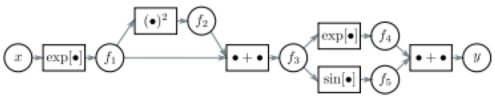


Figure 1: Image for question 9 and 10

Problem 9 (15 points)

This problem explores computing derivatives on general acyclic computational graphs. Consider the function:

$$y = \exp(\exp[x] + \exp[x]^2) + \sin(\exp[x] + \exp[x]^2).$$

We can break this down into a series of intermediate computations so that:

$$\begin{aligned} f_1 &= \exp[x], \\ f_2 &= f_1^2, \\ f_3 &= f_1 + f_2, \\ f_4 &= \exp[f_3], \\ f_5 &= \sin[f_3], \\ y &= f_4 + f_5. \end{aligned}$$

The associated computational graph is depicted in figure 1. Compute the derivative $\frac{\partial y}{\partial x}$ by reverse-mode differentiation. In other words, compute in order:

$$\frac{\partial y}{\partial f_5}, \frac{\partial y}{\partial f_4}, \frac{\partial y}{\partial f_3}, \frac{\partial y}{\partial f_2}, \frac{\partial y}{\partial f_1}, \text{ and } \frac{\partial y}{\partial x},$$

using the chain rule in each case to make use of the derivatives already computed.

I need to compute $\frac{\partial y}{\partial x}$ For the nested Function using backpropaga= tion. My approach is to break the complex function into simple intermediate steps.

$$y = \exp(\exp[x] + \exp[x]^2) + \sin(\exp[x] + \exp[x]^2)$$

Breaking down the computational graph:

$f_1 = \exp[x]$

$f_2 = f_1^2$

$f_3 = f_1 + f_2$

$f_4 = \exp[f_3]$

$f_5 = \sin[f_3]$

$y = f_4 + f_5$

(a) $\frac{\partial y}{\partial f_5} \rightarrow y = f_4 + f_5 \quad \frac{\partial y}{\partial f_5} = 1$

(b) $\frac{\partial y}{\partial f_4}$

$y = f_4 + f_5$

$\frac{\partial y}{\partial f_4} = 1$

(c) $\partial y / \partial f_3$

$f_3 \rightarrow$ 2 childrens $\begin{cases} f_4 \\ f_5 \end{cases} \rightarrow$ Accumulate gradients from both parts.

$$\frac{\partial y}{\partial f_3} = \frac{\partial y}{\partial f_4} \cdot \frac{\partial f_4}{\partial f_3} + \frac{\partial y}{\partial f_5} \cdot \frac{\partial f_5}{\partial f_3}$$

since $f_4 = \exp[f_3] \rightarrow \frac{\partial f_4}{\partial f_3} = \exp[f_3] = f_4$

$f_5 = \sin[f_3] \rightarrow \frac{\partial f_5}{\partial f_3} = \cos[f_3]$

$$\frac{\partial y}{\partial f_3} = 1 \cdot f_4 + 1 \cdot \cos[f_3]$$

$$= f_4 + \cos[f_3]$$

(D) $\frac{dy}{df_2}$ | f_2 only affects f_3 ; so:

$$\frac{\partial y}{\partial f_2} = \frac{\partial y}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_2}$$

since $f_3 = f_1 + f_2 \longrightarrow \frac{\partial f_3}{\partial f_2} = 1$

$$\frac{\partial y}{\partial f_2} = (f_4 + \cos[f_3]) \cdot 1$$

$$= f_4 + \cos[f_3]$$

(E) $\partial y / \partial f_1$

$f_1 \longrightarrow$ 2 childrens
- to f_3
- to f_2

$$\frac{\partial y}{\partial f_1} = \frac{\partial y}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_1} + \frac{\partial y}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_1}$$

considering $f_3 = f_1 + f_2 \longrightarrow \frac{\partial f_3}{\partial f_1} = 1$

$$f_2 = f_1^2 \longrightarrow \frac{\partial f_2}{\partial f_1} = 2f_1$$

$$\frac{\partial y}{\partial f_1} = (f_4 + \cos[f_3]) \cdot 1 + (f_4 + \cos[f_3]) \cdot 2f_1$$
$$= (f_4 + \cos[f_3])(1 + 2f_1)$$

so, $\frac{\partial y}{\partial x} \longrightarrow \frac{\partial y}{\partial f_1} \cdot \frac{\partial f_1}{\partial x}$

since $f_1 = \exp[x] \rightarrow \frac{\partial f_1}{\partial x} = \exp[x] = f_1$

$$\therefore \frac{\partial y}{\partial x} = (f_4 + \cos[f_3])(1 + 2f_1) \cdot f_1$$
$$= f_1 (f_4 + \cos[f_3]) \cdot (1 + 2f_1)$$

Problem 10 (15 points)

For the same function as in problem 9, compute the derivative $\frac{\partial y}{\partial x}$ by forward-mode differentiation. In other words, compute in order:

$$\frac{\partial f_1}{\partial x}, \frac{\partial f_2}{\partial x}, \frac{\partial f_3}{\partial x}, \frac{\partial f_4}{\partial x}, \frac{\partial f_5}{\partial x}, \text{ and } \frac{\partial y}{\partial x},$$

using the chain rule in each case to make use of the derivatives already computed. Why do we not use forward-mode differentiation when we calculate the parameter gradients for deep networks?

For this problem, I need to compute the same derivative $\frac{\partial y}{\partial x}$ than problem (9) but using forward mode differentiation. The difference is in the direction: I start at x and propagate derivatives forward to the graph until I reach y.

In Forward mode, I compute how changes in x affect each intermediate variable as I go for = ward. This constrasts with reverse mode where I computed how Y depends on each variable working backwards.

computational graph

$f_1 = \exp[x]$

$f_2 = f_1^2$

$f_3 = f_1 + f_2$

$f_4 = \exp[f_3]$

$f_5 = \sin[f_3]$

$y = f_4 + f_5$

Applying the forward differentiation:

$\frac{\partial f_1}{\partial x}$ | $f_1 = \exp[x]$

$\frac{\partial f_1}{\partial x} = \exp[x] = f_1$

$\frac{\partial f_2}{\partial x}$ | $f_2 = f_1^2 \rightarrow$ using chain rule

$\rightarrow \frac{\partial f_2}{\partial x} = \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial x} = 2f_1 \cdot f_1 = 2f_1^2$

$\frac{\partial f_3}{\partial x}$ | $f_3 = f_1 + f_2$

$$\frac{\partial f_3}{\partial x} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial x}$$

$$= f_1 + 2f_1^2$$

$$= f_1 (1 + 2f_1)$$

$\partial f_4 / \partial x$ |

$f_4 = \exp [f_3]$

$$\frac{\partial f_4}{\partial x} = \frac{\partial f_4}{\partial f_3} \cdot \frac{\partial f_3}{\partial x}$$

$$= \exp [f_3] \cdot f_1 (1 + 2f_1)$$

$$= f_4 \cdot f_1 (1 + 2f_1)$$

$\frac{\partial f_5}{\partial x}$ | $f_5 = \sin [f_3]$

$$\frac{\partial f_5}{\partial x} = \frac{\partial f_5}{\partial f_3} \cdot \frac{\partial f_3}{\partial x}$$

$$\frac{\partial f_5}{\partial x} = \cos (f_3) \cdot f_1 (1 + 2 f_1)$$

Final answer

$y = f_4 + f_5$ // f_4 and f_5 depends on x

$$\frac{\partial y}{\partial x} = \frac{\partial f_4}{\partial x} + \frac{\partial f_5}{\partial x}$$

$$= f_4 \cdot f_1 (1 + 2 f_1) + \cos[f_3] \cdot f_1 (1 + 2 f_1)$$

$$= f_1 (1 + 2 f_1)(f_4 + \cos[f_3]).$$

Forward is inefficient for deep neural networks since it implies one pass per variable, so you can just think that if we have 1.000.000 parameters, we will need 1000000 forward passes and, of course, the computational cost of this is really high; In backpropagation (reverse-mode) one backward pass all the parameters (RT $\rightarrow O(1)$).