

Neural Networks: Backpropagation

Seung-Hoon Na¹

¹Department of Computer Science
Chonbuk National University

2018.10.25

Jacobian matrix

- Two functions $f(x, y)$, $g(x, y)$ with two parameters x , y

$$\begin{aligned}f(x, y) &= 3x^2y \\g(x, y) &= 5xy + y^3\end{aligned}$$

- **Jacobian matrix** (numerator layout):

$$\begin{aligned}J &= \begin{bmatrix} \nabla f(x, y) \\ \nabla g(x, y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} & \frac{\partial f(x, y)}{\partial y} \\ \frac{\partial g(x, y)}{\partial x} & \frac{\partial g(x, y)}{\partial y} \end{bmatrix} \\ &= \begin{bmatrix} 6yx & 3x^2 \\ 5y & 5x + 3y^2 \end{bmatrix}\end{aligned}$$

- **Jacobian matrix** (denominator layout):

$$J^T = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} & \frac{\partial g(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} & \frac{\partial g(x, y)}{\partial y} \end{bmatrix}$$

Jacobian: Generalization

- $\mathbf{y} = \mathbf{f}(\mathbf{x})$: a vector of m scalar-valued functions that each take a vector \mathbf{x}

$$y_1 = f_1(\mathbf{x})$$

$$\vdots$$

$$y_m = f_m(\mathbf{x})$$

- Jacobian matrix: has m rows for m equations.

$$\begin{aligned} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} &= \begin{bmatrix} \nabla f_1(\mathbf{x}) \\ \dots \\ \nabla f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \dots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \vdots & \dots & \vdots \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix} \end{aligned}$$

Jacobian: Generalization

			vector
	scalar	x	\mathbf{x}
scalar	f	$\frac{\partial f}{\partial x}$	$\frac{\partial f}{\partial \mathbf{x}}$
vector	\mathbf{f}	$\frac{\partial \mathbf{f}}{\partial x}$	$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$

- Jacobian is the multiplication of two other Jacobians

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{g}(\mathbf{x})) &= \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \\ &= \begin{bmatrix} \frac{\partial f_1}{\partial g_1} & \cdots & \frac{\partial f_1}{\partial g_k} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_m}{\partial g_1} & \cdots & \frac{\partial f_m}{\partial g_k} \end{bmatrix} \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial g_k}{\partial x_1} & \cdots & \frac{\partial g_k}{\partial x_n} \end{bmatrix}\end{aligned}$$

Vector chain rule: Example

- $\mathbf{y} = \mathbf{f}(x)$

$$\mathbf{y} = \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} \ln(x^2) \\ \sin(3x) \end{bmatrix}$$

- $\mathbf{y} = \mathbf{f}(\mathbf{g}(x))$: introduce two intermediate variables g_1, g_2 :

$$\begin{aligned} \mathbf{g} &= \begin{bmatrix} g_1(x) \\ g_2(x) \end{bmatrix} = \begin{bmatrix} x^2 \\ 3x \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} f_1(\mathbf{g}) \\ f_2(\mathbf{g}) \end{bmatrix} = \begin{bmatrix} \ln(g_1) \\ \sin(g_2) \end{bmatrix} \end{aligned} \quad (1)$$

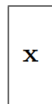
Jacobian: Generalization

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{g}(\mathbf{x})) = \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$$

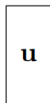
scalar



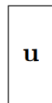
vector



vector



vector



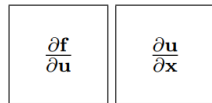
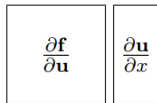
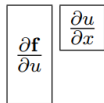
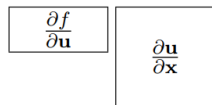
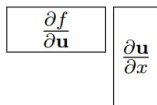
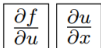
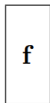
scalar



scalar

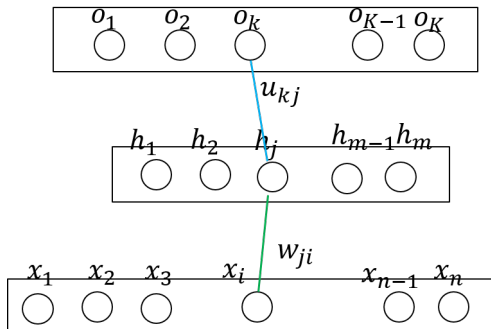


vector

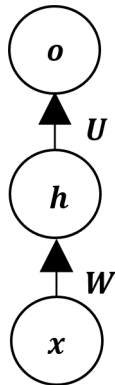
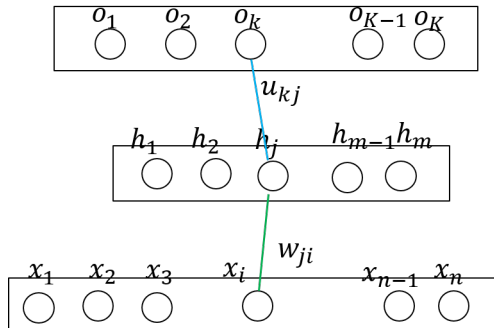


MLP with single hidden layer: Notation

- For simplicity, a network has single hidden layer only
 - o_k : k -th output unit, h_j : j -th hidden unit, x_i : i -th input
 - u_{kj} : weight b/w j -th hidden and k -th output
 - w_{ji} : weight b/w i -th input and j -th hidden
 - Bias terms are also contained in weights



MLP with single hidden layer: Matrix notation



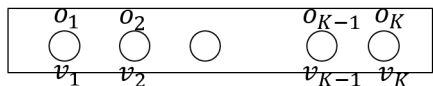
$$\mathbf{h} = \max(\mathbf{W}\mathbf{x}, 0)$$

$$\mathbf{o} = \text{softmax}(\mathbf{U}\mathbf{h})$$

Typical Setting for Classification

- K : the number of labels
- Input layer: Input values (raw features)
- Output layer: **Scores** of labels
- **Softmax** layer: $\mathbf{o} = \text{softmax}(\mathbf{v})$

$$o_k = \frac{\exp(v_k)}{\sum_i \exp(v_i)} = \frac{\exp(v_k)}{Z}$$



Output layer
(with softmax)



Hidden layer



Input layer

Learning as Optimization

- Training data: $\mathcal{T} := \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
 - \mathbf{x}_i : i -th input feature vector
 - y_i (or \mathbf{y}_i): i -th target label
- Parameter: $\theta := \{\mathbf{W}, \mathbf{U}\}$
 - Weight matrices: Input-to-hidden, and hidden-to-output
- Objective function (= Loss function)
 - Take Negative Log-likelihood (NLL) as Empirical risk

$$J(\theta) = \text{Loss}(\mathcal{T}, \theta) = - \sum_{(\mathbf{x}, y) \in \mathcal{T}} \log P(y|\mathbf{x})$$

- Training process
 - Known as Empirical risk minimization

$$\theta^* = \operatorname{argmin}_{\theta} J(\theta)$$

- Gradient Descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial J}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \mathbb{E}_{(\mathbf{x}, y)} [-\log P(y|\mathbf{x})]$$

- Batch algorithm
 - Expectations over the training set are required
 - But, computing expectations exactly is very expensive, as it evaluates on every example in the entire dataset
- **Minibatch** algorithm
 - In practice, we compute these expectations by randomly sampling a small number of examples from the dataset, then taking the average over only those examples
 - Using exact gradient using large examples does not significantly reduce the estimation error: Slow convergence

Stochastic Gradient Method

- 1 Randomly a minibatch of m samples $\{(\mathbf{x}, y)\}$ from training data
- 2 Define NLL for $\{(\mathbf{x}_i, y_i)\}$

$$J(\boldsymbol{\theta}) = \sum_{1 \leq i \leq m} \log(y_i | \mathbf{x}_i)$$

- 3 Compute derivatives $\frac{\partial J}{\partial \mathbf{W}}$ for $\mathbf{W} \in \boldsymbol{\theta}$
- 4 Update weight matrix for $\mathbf{W} \in \boldsymbol{\theta}$:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J}{\partial \mathbf{W}}$$

Iterate the above procedure until stopping criteria is satisfied

Logistic regression for binary classification

- (\mathbf{x}, y) : a training example for binary classification where $y \in \{0, 1\}$
- Logistic regression function:

$$o(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

which is rewritten to:

$$\begin{aligned} z &= \mathbf{w}^T \mathbf{x} + b \\ o &= \sigma(z) \end{aligned}$$

- J : the log-likelihood on (\mathbf{x}, y)

$$J = y \log(o) + (1 - y) \log(1 - o)$$

$$\frac{\partial J}{\partial o} = \frac{y}{o} - \frac{1 - y}{1 - o}$$

Logistic regression: Deriv J wrt \mathbf{w}

$$\frac{\partial z}{\partial \mathbf{w}} = \mathbf{x}^T$$

$$\frac{\partial o}{\partial z} = \sigma(z)(1 - \sigma(z)) = o(1 - o)$$

All together lead to:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= \frac{\partial J}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial \mathbf{w}} \\ &= \left(\frac{y}{o} - \frac{1-y}{1-o} \right) \cdot o(1-o) \mathbf{x}^T \\ &= (y - o) \mathbf{x}^T\end{aligned}$$

Logistic regression for multi-class classification

- K : the number of labels
- (\mathbf{x}, k) : a training example where $k \in \{1, \dots, K\}$
- Logistic regression function:

$$\mathbf{o}(\mathbf{x}) = \text{softmax}(\mathbf{W}^T \mathbf{x} + \mathbf{b})$$

which is rewritten to:

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$

$$\mathbf{o} = \text{softmax}(\mathbf{z})$$

where $\text{softmax}(\mathbf{z}) = \exp(\mathbf{z}) / \sum_i \exp(\mathbf{z}_i) = \exp(\mathbf{z}) / Z$

- J : the log-likelihood on (\mathbf{x}, k)

$$J = \mathbf{y}^T \log(\mathbf{o})$$

where \mathbf{y} is one-hot encoding for target label.

$$\mathbf{y} = [0 \dots 1 \dots 0]^T$$

where $y_i = \mathcal{I}(i = k)$ where k is the target label.

Logistic regression: Deriv J wrt \mathbf{W}

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{o}} &= \left[0 \cdots \frac{1}{o_k} \cdots 0 \right] \\ \frac{\partial \mathbf{o}}{\partial \mathbf{z}} &= \left[\frac{\partial o_j}{\partial z_i} \right]_{ij} = \left[\frac{\exp(z_j) (\mathcal{I}(i=j) \cdot \sum_k \exp(z_k) - \exp(z_i))}{(\sum_k \exp(z_k))^2} \right]_{ij} \\ &= \left[\frac{\exp(z_j) (\mathcal{I}(i=j) \cdot Z - \exp(z_i))}{(Z)^2} \right]_{ij}\end{aligned}$$

Thus, we have:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{z}} &= \frac{\partial J}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \mathbf{z}} \\ &= \left[0 \cdots \frac{1}{o_k} \cdots 0 \right] \left[\frac{\exp(z_j) (\mathcal{I}(i=j) \cdot Z - \exp(z_i))}{(Z)^2} \right]_{ij} \\ &= \left[\mathcal{I}(1=k) - o_1 \quad \cdots \quad 1 - o_k \quad \cdots \quad \mathcal{I}(K=k) - o_K \right]\end{aligned}$$

Logistic regression: Deriv J wrt \mathbf{W} (Cont.)

Given $\mathbf{z} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$,

$$\frac{\partial z_i}{\partial \mathbf{W}_{i*}} = \mathbf{x}^T$$

where \mathbf{W}_{i*} is i -th row vector of \mathbf{W} .

$$\frac{\partial J}{\partial \mathbf{W}_{i*}} = \frac{\partial J}{\partial z_i} \frac{\partial z_i}{\partial \mathbf{W}_{i*}} = (\mathcal{I}(i = k) - o_i) \mathbf{x}^T$$

Finally, this leads to:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}} &:= \begin{bmatrix} \frac{\partial J}{\partial \mathbf{W}_{1*}} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{W}_{K*}} \end{bmatrix} = \begin{bmatrix} (\mathcal{I}(1 = k) - o_1) \mathbf{x}^T \\ \vdots \\ (\mathcal{I}(K = k) - o_K) \mathbf{x}^T \end{bmatrix} \\ &= \begin{bmatrix} (\mathcal{I}(1 = k) - o_1) \\ \vdots \\ (\mathcal{I}(K = k) - o_K) \end{bmatrix} \mathbf{x}^T = \frac{\partial J}{\partial \mathbf{z}} \mathbf{x}^T \end{aligned}$$

MLP with single hidden layer: Log-likelihood

$$\mathbf{h} = \max(\mathbf{W}\mathbf{x}, 0)$$

$$\mathbf{o} = \text{softmax}(\mathbf{U}\mathbf{h})$$

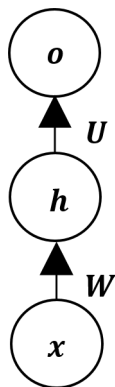
- J : the log-likelihood on single example (\mathbf{x}, \mathbf{y})

$$J = \mathbf{y}^T \log(\mathbf{o})$$

- \mathbf{y} : one-hot encoding for target label.

$$\mathbf{y} = [0 \cdots 1 \cdots 0]^T$$

where $y_i = \mathcal{I}(i = k)$ where k is a target label.



Derivative of J wrt Output layer

$$\mathbf{v} = \mathbf{U}\mathbf{h}$$

$$\mathbf{o} = \text{softmax}(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_i \exp(v_i)} = \frac{\exp(\mathbf{v})}{Z}$$

$$J = \mathbf{y}^T \log(\mathbf{o})$$

$$\frac{\partial J}{\partial \mathbf{o}} = \left[0 \cdots \frac{1}{o_k} \cdots 0 \right]$$

$$\frac{\partial \mathbf{o}}{\partial \mathbf{v}} = \left[\frac{\partial o_j}{\partial v_i} \right]_{ij} = \left[\frac{\exp(v_j) (\mathcal{I}(i=j) \cdot \sum_k \exp(v_k) - \exp(v_i))}{(\sum_k \exp(v_k))^2} \right]_{ij}$$

$$= \left[\frac{\exp(v_j) (\mathcal{I}(i=j) \cdot Z - \exp(v_i))}{(Z)^2} \right]_{ij}$$

Derivative of J wrt Output layer (Cont.)

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{v}} &= \frac{\partial J}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \mathbf{v}} \\ &= \left[0 \cdots \frac{1}{o_k} \cdots 0 \right] \left[\frac{\exp(v_j) (\mathcal{I}(i=j) \cdot Z - \exp(v_i))}{(Z)^2} \right]_{ij} \\ &= \left[0 \cdots \frac{Z}{\exp(v_k)} \cdots 0 \right] \left[\frac{\exp(v_j) (\mathcal{I}(i=j) \cdot Z - \exp(v_i))}{(Z)^2} \right]_{ij} \\ &= \left[-\frac{\exp(v_1)}{Z} \quad \cdots \quad 1 - \frac{\exp(v_k)}{Z} \quad \cdots \quad -\frac{\exp(v_K)}{Z} \right] \\ &= \left[-o_1 \quad \cdots \quad 1 - o_k \quad \cdots \quad -o_K \right]\end{aligned}$$

- Let $\delta^{(o)}$ be the error signal for output layer:

$$\delta^{(o)} := \frac{\partial J}{\partial \mathbf{v}} = \left[-o_1 \quad \cdots \quad 1 - o_k \quad \cdots \quad -o_K \right]$$

Here, note that $\delta^{(o)}$ is a **row vector**.

Error propagation to hidden layer

- Hidden-to-output: $\mathbf{v} = \mathbf{U}\mathbf{h}$

$$\frac{\partial \mathbf{v}}{\partial \mathbf{h}} = \mathbf{U}$$

- Let $\delta^{(h)}$ be the error signal for hidden layer

$$\begin{aligned}\delta^{(h)} &:= \frac{\partial J}{\partial \mathbf{h}} = \frac{\partial J}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{h}} \\ &= \delta^{(o)} \mathbf{U}\end{aligned}$$

Deriv of J wrt hidden-output weight matrix \mathbf{U}

- Hidden-to-output: $\mathbf{v} = \mathbf{U}\mathbf{h}$

$$\frac{\partial v_i}{\partial \mathbf{U}_{i*}} = \mathbf{h}^T$$

where \mathbf{U}_{i*} is i -th row vector of \mathbf{U} .

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{U}_{i*}} &= \frac{\partial J}{\partial v_i} \frac{\partial v_i}{\partial \mathbf{U}_{i*}} = \delta_i^{(o)} \mathbf{h}^T \\ \frac{\partial J}{\partial \mathbf{U}} &:= \begin{bmatrix} \frac{\partial J}{\partial \mathbf{U}_{1*}} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{U}_{K*}} \end{bmatrix} = \boldsymbol{\delta}^{(o)T} \mathbf{h}^T \end{aligned}$$

Error prop to input layer

- Input-to-hidden layer: $\mathbf{z} = \mathbf{W}\mathbf{x}$ $\mathbf{h} = \max(\mathbf{z}, 0)$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \text{diag}(\mathcal{I}(z_i > 0))$$
$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathbf{W}$$

- Let $\delta^{(z)}$ be the error signal for the pre-activated hidden layer

$$\delta^{(z)} := \frac{\partial \mathbf{J}}{\partial \mathbf{z}} = \frac{\partial \mathbf{J}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \delta^{(h)} \text{diag}(\mathcal{I}(z_i > 0))$$

- Let $\delta^{(x)}$ be the error signal for input layer

$$\delta^{(x)} := \frac{\partial \mathbf{J}}{\partial \mathbf{x}} = \frac{\partial \mathbf{J}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \delta^{(h)} \text{diag}(\mathcal{I}(z_i > 0)) \mathbf{W}$$

Deriv of J wrt input-hidden weight matrix \mathbf{W}

- Input-to-hidden layer: $\mathbf{z} = \mathbf{W}\mathbf{x}$

$$\frac{\partial z_i}{\partial \mathbf{W}_{i*}} = \mathbf{x}^T$$

where \mathbf{W}_{i*} is i -th row vector of \mathbf{W} .

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}_{i*}} &= \frac{\partial J}{\partial z_i} \frac{\partial z_i}{\partial \mathbf{W}_{i*}} = \delta_i^{(z)} \mathbf{x}^T \\ \frac{\partial J}{\partial \mathbf{W}} &:= \begin{bmatrix} \frac{\partial J}{\partial \mathbf{W}_{1*}} \\ \vdots \\ \frac{\partial J}{\partial \mathbf{W}_{m*}} \end{bmatrix} = \boldsymbol{\delta}^{(z)T} \mathbf{x}^T \end{aligned}$$

- Here, error messages such as $\delta^{(o)}$, $\delta^{(h)}$ are row vectors
- But, we can define $\delta^{(o)}$, $\delta^{(h)}$ as column vectors and derive backprop again.
- In this case, only the slight modification on error prop is necessary