AI Security

Neural Networks for Classification

Sangdon Park

POSTECH

March 4, 2026

Motivation

- Neural nets are practical models.
- Attacks (*i.e.*, generating adversarial examples) were conducted over neural nets.
- To understand attack mechanism, we need to understand neural nets first.

Outline

Classification Problem

Goal

Find a classifier $h : \mathcal{X} \to \mathcal{Y}$ such that a prediction $h(x)$ on an example x "matches" to a true label y.

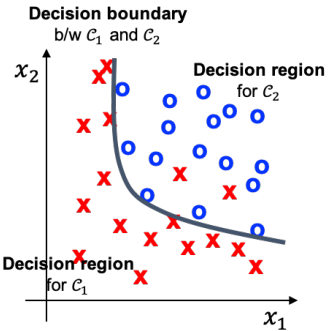
- The correctness metric, related to "matches", depends on learning setups.
- In offline supervised learning, solving the following is an objective:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{1}\left(h(x) \neq y\right)$$

  - $\mathcal{X}$: a set of examples
  - $\mathcal{Y}$: a set of labels
  - $\mathcal{D}$: a distribution over labeled examples $\mathcal{X} \times \mathcal{Y}$
  - $\mathcal{H}$: a set of classifiers defined over $\mathcal{X}$ and $\mathcal{Y}$
  - training samples are i.i.d.
- Use cases: handwriting recognition, speech recognition, biological classification, drug discovery, spam filtering, ...,

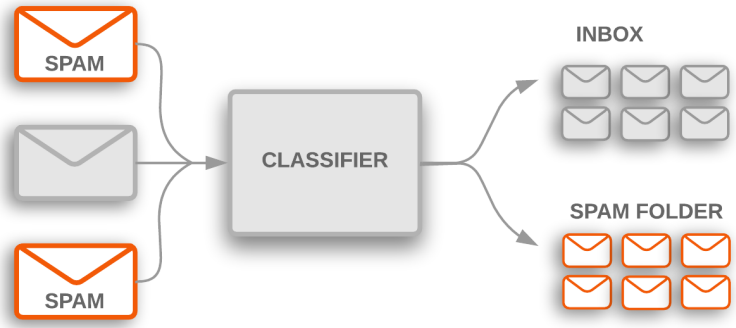
Implicit Goal: Learning A Decision Boundary

- Suppose that there are two classes $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{Y}$
- A classifier assigns a new data point x to $\mathcal{C}_1$ only if it is in the decision region $\mathcal{R}_1$ for $\mathcal{C}_1$, i.e., classification function $h(x) = \mathcal{C}_k$ if $x \in \mathcal{R}_k$.
- Decision boundaries (or surface) are boundaries between decision regions



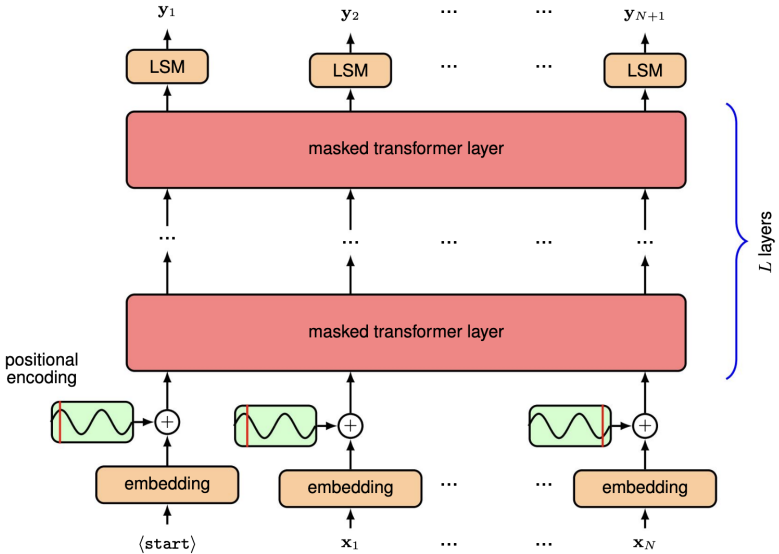
Learning a classifier = Drawing a decision boundary

Spam Filtering



https://medium.com/@naveeen.kumar.k/naive-bayes-spam-detection-7d087cc96d9d

Next-Token Prediction



What's Missing?

Recall our concise objective for classification:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{1}\left(h(x) \neq y\right)$$

- How to define $\mathcal{H}$?
- How to solve min?

Outline

Logistic Function and Its Properties

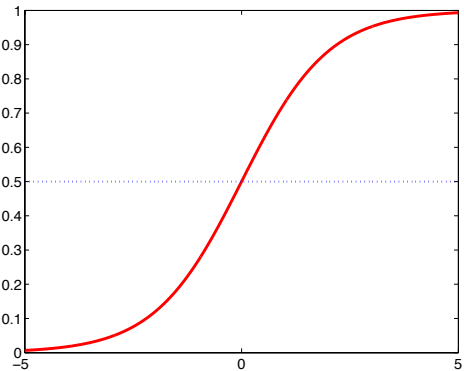


Figure: $\sigma(t) = \frac{1}{1+e^{-t}}$

- $\sigma(t) \to 0$ as $t \to -\infty$
- $\sigma(t) \to 1$ as $t \to \infty$
- $\sigma(-t) = 1 - \sigma(t)$ (a.k.a. a symmetry property)
- $\frac{d}{dt}\sigma(t) = \sigma(t)\sigma(-t) = \sigma(t)(1 - \sigma(t))$

Logistic Regression Model in Binary Classification

Definition (a logistic regression model)

$$f(x) = \sigma(w^T x + b) = \sigma(\mathbf{w}^T \mathbf{x}) \in [0, 1],$$

where $w, x \in \mathbb{R}^D$, $b \in \mathbb{R}$, $\mathbf{w} = [w^T, b^T]^T$, and $\mathbf{x} = [x^T, 1]^T$.

- A classifier is defined as $h(x) = \begin{cases} +1 & \text{if } \sigma(\mathbf{w}^T \mathbf{x}) \geq 0.5 \\ -1 & \text{otherwise} \end{cases} = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases}$.
- It has probablicstic interpretaion, as follows:

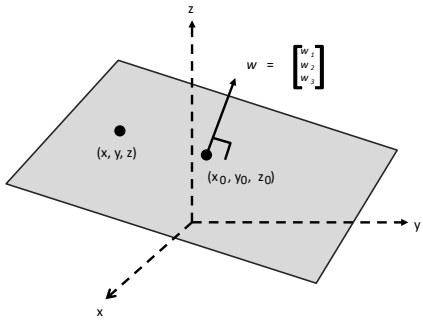
$$\hat{p}(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = f(x) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})} \text{ and } \hat{p}(y = -1 \mid \boldsymbol{x}, \boldsymbol{w}) = 1 - f(x) = \frac{1}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})}$$

or equivalently,

$$\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + \exp(-y \boldsymbol{w}^\top \boldsymbol{x})}$$

Decision Boundary of Logistic Regression Models

The decision boundary of the logistic regression is $\{x \in \mathbb{R}^D \mid w^T x + b = 0\}$.



- Recall that the classifier $h(x)$ for the logistic regression: $h(x) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases}$
- This means that $\boldsymbol{x}$ is classified by the sign of $\boldsymbol{w}^T\boldsymbol{x}$.

Main Learning Objective for Classification

Learning Objective: Log-likelihood Maximization

Maximize the likelihood of a data set Z given a parameterized probabilistic model $p(\cdot \mid \mathbf{w})$, *i.e.*,

$$\hat{w} = \arg\max_{\boldsymbol{w}} p(Z \mid \mathbf{w})$$

- We assume the labeled examples in Z are i.i.d. samples from a fixed but hidden distribution $\mathcal{D}$.
- Then, we decompose the likelihood of a data set Z as follows:

$$p(Z \mid \mathbf{w}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n \mid \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) p(\boldsymbol{x}_n \mid \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) p(\boldsymbol{x}_n),$$

  where the last equality hold if the parameter does not depend on $\mathbf{x}_n$.

- How to model $p(y_n \mid \mathbf{x}_n, \mathbf{w})$? Use a logistic regression model.

Learning Objective for Logistic Regression

Negative Log-likelihood Minimization

Minimize the negative log-likelihood of a data set Z by modeling $p(y_n \mid \mathbf{x}_n, \mathbf{w})$ via a logistic regression model, *i.e.,*

$$\hat{w} = \arg\min_{\boldsymbol{w}} -\sum_{n=1}^{N} y_n \ln \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \ln(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$$

- The decomposed likelihood is rewritten as follows, assuming $\mathcal{Y} = \{0, 1\}$ instead of $\mathcal{Y} = \{-1, 1\}$:

$$-\ln p(Z \mid \mathbf{w}) \propto -\ln \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) = -\ln \prod_{n=1}^{N} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)\right)^{(1-y_n)}$$

- Recall *cross-entropy*, which measures the difference between two probability distributions, *i.e.,*

$$-\sum_{y \in \mathcal{Y}} p(y) \ln q(y) \quad \text{for a discrete set } \mathcal{Y}.$$

- Letting $\mathcal{Y} = \{0, 1\}$, $p(y)$ be a true label distribution (with one-hot encoding), and $q(y) \coloneqq \sigma(\mathbf{w}^\top \mathbf{x})$, our objective is the same as minimizing the average cross entropy. Check out!

Connection to Expected Classification Error Minimization I

Relation between the zero-one loss and cross-entropy loss

The cross-entropy loss with some constant factor is an upper bound of the zero-one loss, *i.e.*,

$$\mathbb{1}\left(h(x) \neq y\right) \leq C \cdot \ell_{\mathsf{CE}}(x, y, f)$$

- Recall that we wish to find a classifier h that minimizes the expected error, *i.e.*,

$$\min_{h \in \mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{1}\left(h(x) \neq y\right)$$

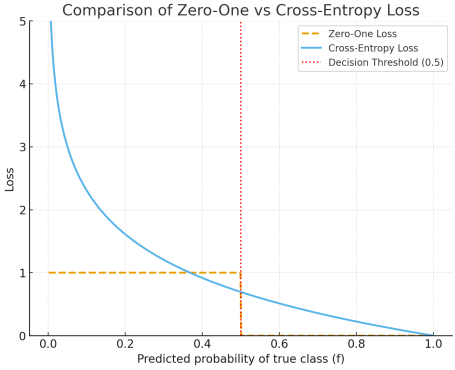
- Given the i.i.d. samples $\left((x_n, y_n)\right)_{n\in[N]} \sim \mathcal{D}^N$, consider the following empirical approximation and the relation with the cross-entropy loss:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{1}\left(h(x) \neq y\right) \approx \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left(h(x_n) \neq y_n\right) \leq \frac{1}{N}\sum_{n=1}^{N} C \cdot \ell_{\mathsf{ce}}(x_n, y_n, f),$$

  where $\ell_{\mathsf{ce}}(x, y, f) = -\Big(y \ln f(x) + (1-y)\ln\big(1 - f(x)\big)\Big)$

- Thus, minimizing the average cross-entropy loss minimizes the empirical classification error.

Connection to Expected Classification Error Minimization II



Assuming binary classification, define loss functions as follows:

$$\text{(Zero-One Loss)} = \begin{cases} 0 & \text{if } \mathbb{1}(f(x) > 0.5) = y \\ 1 & \text{otherwise} \end{cases} \qquad \text{(Cross-Entropy Loss)} = \begin{cases} -\ln f(x) & \text{if } y = 1 \\ -\ln(1 - f(x)) & \text{if } y = 0 \end{cases}$$

Outline

Outline

How to solve the following minimization?

Negative Log-likelihood Minimization

$$\hat{w} = \arg\min_{\boldsymbol{w}} -\sum_{n=1}^{N} y_n \ln \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \ln(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$$

1. Gradient Decent Method – the first-order method
2. Newton Method – the second-order method but we will not touch this.

Gradient

Definition (Gradient)

If $f : \mathbb{R}^D \to \mathbb{R}$ is differentiable, the Gradient $\nabla f(\mathbf{x})$ of f at $\mathbf{x} \in \mathbb{R}^D$ is a $D \times 1$ vector with $\frac{\partial f(\mathbf{x})}{\partial x_i}$ at the i-th element, *i.e.*,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_D} \end{bmatrix}$$

- "Change of function values"
- The Derivative of f at $\mathbf{x}$ is the same as $(\nabla f(\mathbf{x}))^\top$.
- Derivative and Gradient are equivalent and the only difference is the vector shape.

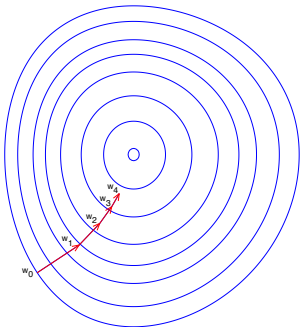
Gradient Descent Method: A First-order Method

Gradient Descent Method

Given a learning rate γ and learning objective $\mathcal{L}$, the gradient descent method update parameters

$$\boldsymbol{w}^{\text{new}} \leftarrow \boldsymbol{w}^{\text{old}} - \gamma \nabla_{\mathbf{w}^{\text{old}}} \mathcal{L}(\mathbf{w}).$$

- A generic learning algorithm for almost all cases, specially for no closed-form solution exists.



Gradient Descent Method for Logistic Regression Models I

Update rule for logistic regression models

$$\boldsymbol{w}^{\text{new}} \leftarrow \boldsymbol{w}^{\text{old}} + \gamma \sum_{n=1}^{N} \left[y_n - \sigma \left(\left(\boldsymbol{w}^{\text{old}} \right)^{\top} \mathbf{x}_n \right) \right] \mathbf{x}_n$$

- Recall the learning objective for logistic regression models:

$$\mathcal{L}(\mathbf{w}) \coloneqq -\sum_{n=1}^{N} \left[y_n \ln \sigma(\boldsymbol{w}^{\top} \boldsymbol{x}_n) + (1 - y_n) \ln(1 - \sigma(\boldsymbol{w}^{\top} \boldsymbol{x}_n)) \right]$$

- Generally no closed-form solution of $\mathbf{w}$ to minimize this objective.
- Consider the gradient of $\mathcal{L}(\mathbf{w})$ with respect to $\mathbf{w}$, *i.e.*, $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}$.

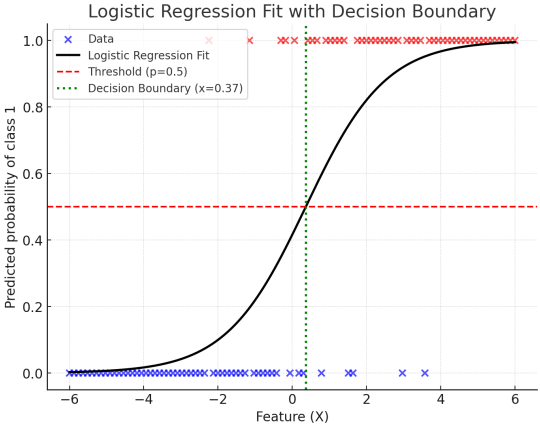
Gradient Descent Method for Logistic Regression Models II

- Derivation: letting $\hat{y}_n \coloneqq \sigma(\mathbf{w}^\top \mathbf{x}_n)$, we have

$$
\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}} &= -\frac{\partial}{\partial \mathbf{w}} \sum_{n=1}^{N} [y_n \ln \hat{y}_n + (1-y_n) \ln(1-\hat{y}_n)] \\
&= -\sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n'}{\hat{y}_n} \mathbf{x}_n + (1-y_n) \frac{-\hat{y}_n'}{1-\hat{y}_n} \mathbf{x}_n \right] \\
&= -\sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n(1-\hat{y}_n)}{\hat{y}_n} - (1-y_n) \frac{\hat{y}_n(1-\hat{y}_n)}{1-\hat{y}_n} \right] \mathbf{x}_n \\
&= -\sum_{n=1}^{N} [y_n(1-\hat{y}_n) - (1-y_n)\hat{y}_n] \mathbf{x}_n \\
&= -\sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n.
\end{aligned}
$$

Gradient Descent Method for Logistic Regression Models III

- Fitting example:



Logistic Regression for Multiclass Classification

Almost straightforward to extend logistic regression for binary to multi-class classification.

- Model the input-output mapping by a softmax transformation of linear functions of examples:

$$p(y = k \mid x) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x})}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^\top \boldsymbol{x})}$$

- The negative log-likelihood is given by

$$\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N} \underbrace{\sum_{k=1}^{K} y_{kn} \log p(y_n = k \mid x_n)}_{\text{cross-entropy loss}},$$

  where $y_{kn} = \mathbb{1}[y_n = k]$.

- That's it!

Outline

Neural Network for Function Approximation

Neural networks: a powerful set of function models which consist of

- Perceptrons
- Multi-layer perceptrons
- Convolutions
- Activation function
- Maximum/Average pooling
- Soft-max layer
- Dropout
- ...

Toward Deep Learning

Remarks

1. The representation power of an MLP is from function composition.
2. Training is nothing but optimization and the gradient method is heuristic but efficient.

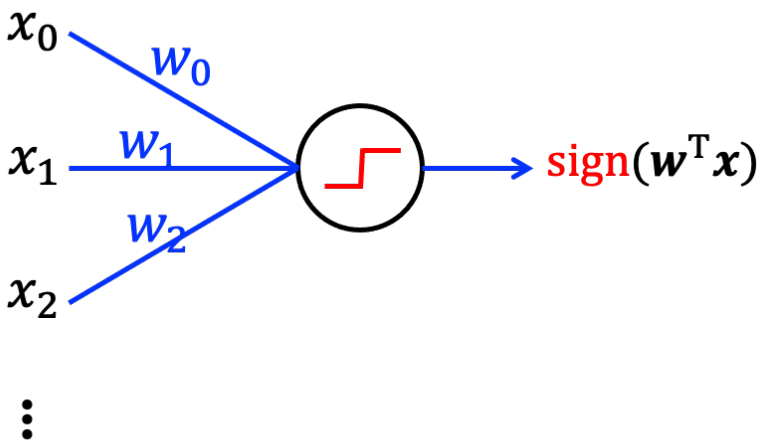
- Deep learning: a composition of differentiable functions to approximate function F, *i.e.*,

$$F(\boldsymbol{x}) \approx f_{\boldsymbol{w}_L}(f_{\boldsymbol{w}_{L-1}}(...f_{\boldsymbol{w}_1}(\boldsymbol{x})...))$$

- "Perceptron" is a basic building block.
- Then, how to make "Perceptron" differentiable?

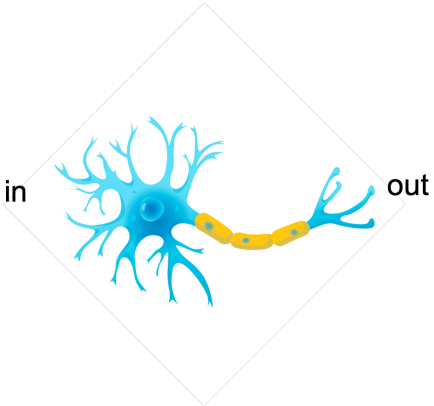
Perceptron

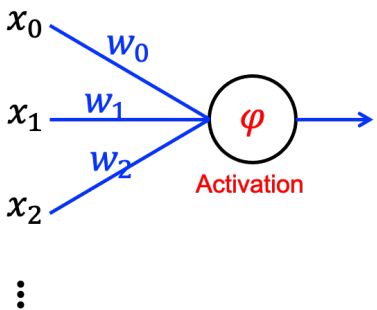
Although a single perceptron with `sign` is trainable, an MLP with `signs` is not since the derivative of `sign` is 0 almost everywhere



A Neuron: McCulloch-Pitts Model (1943)

The sign function can be generalized and approximated by other activation functions





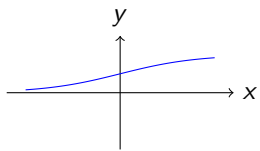
~~$\text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$~~

$\max\{0, \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\}$

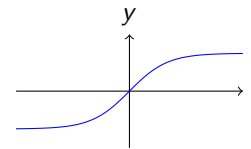
$\exp(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$

Activation Functions

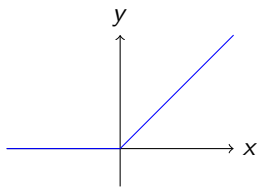
Sigmoid: $\varphi(x) = \frac{1}{1+\exp(-x)}$



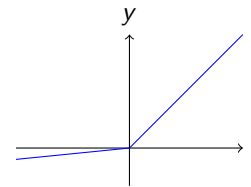
tanh: $\varphi(x) = \tanh(x)$



ReLU (*): $\varphi(x) = \max\{0, x\}$



Leaky ReLU: $\varphi(x) = \max\{0.1x, x\}$



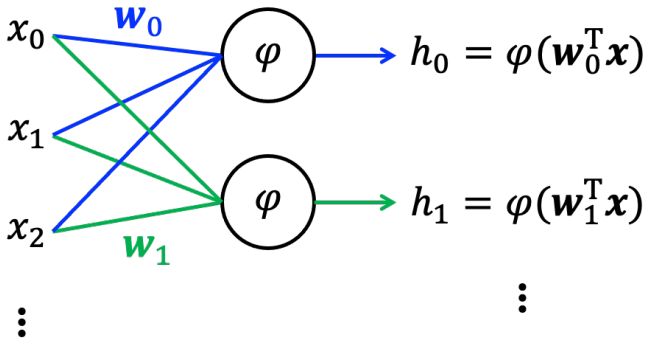
Fully Connected Layer

Definition

A layer where D input features and H output units are connected "with" weight parameters $\mathbf{W}$, bias parameters $\mathbf{b}$, and activation functions *i.e.*,

$$\varphi(\boldsymbol{W}\mathbf{x} + \mathbf{b})$$

is a *fully connected layer*, where $\mathbf{x} \in \mathbb{R}^D$, $\boldsymbol{W} \in \mathbb{R}^{H \times D}$, $\mathbf{b} \in \mathbb{R}^H$.



Multilayer Perceptron

Definition

A multilayer perceptron (MLP) is a stack of fully-connected layers, *i.e.*,

$$\begin{aligned} \boldsymbol{h}_1 &= f_1(\mathbf{x}) = \varphi(\boldsymbol{W}_1\mathbf{x} + \mathbf{b}_1) \\ \boldsymbol{h}_2 &= f_2(\boldsymbol{h}_1) = \varphi(\boldsymbol{W}_2\boldsymbol{h}_1 + \mathbf{b}_2) \\ &\vdots \\ \hat{\mathbf{f}} = \boldsymbol{h}_L &= f_L(\boldsymbol{h}_{L-1}) \end{aligned}$$

or simply

$$\hat{\mathbf{f}} = f_L \circ \cdots \circ f_1(\mathbf{x}).$$

- Note that MLPs provide modularity – we can (almost) freely stack MLPs on top of MLPs.

Learning Objective

Objective

$$\mathcal{L} = \sum_{n=1}^{N} \ell(\mathbf{y}_n, \hat{\mathbf{f}}_n)$$

- We need to compute the partial derivative w.r.t. the model parameter $\boldsymbol{W}_l$ and $\mathbf{b}_l$ for $1 \leq l \leq L$ to train the model.
- ℓ is a loss function, which will be defined later.

Key Question in Learning

Assume that we model single-hidden layer neural network (without bias terms) for regression with squared loss and sigmoid activation by minimizing the following objective:

$$\mathcal{L} = \sum_{n=1}^{N} (y_n - w_2 \sigma(\boldsymbol{w}_1^\top \mathbf{x}_n))^2,$$

where $w_2 \in \mathbb{R}$, $\boldsymbol{w}_1 \in \mathbb{R}^D$, and $\mathbf{x}_n \in \mathbb{R}^D$.
To use gradient descent, we need to compute partial derivative w.r.t. parameters. What would be

- $\partial \mathcal{L} / \partial w_2 =$?
- $\partial \mathcal{L} / \partial \boldsymbol{w}_1 =$?

Compute Partial Derivatives

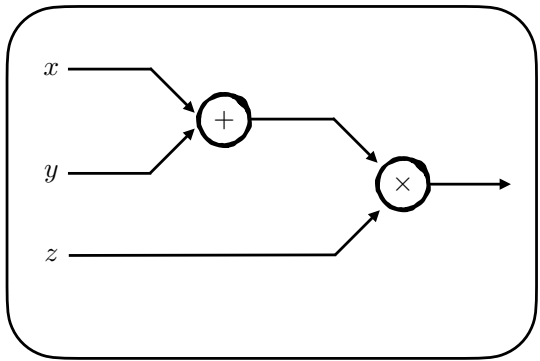
Remark

All we need in learning is the partial derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_l}$ for all $l = 1, ..., L$ (followed by the chain rule).

- You may wish to compute the partial derivative analytically (*i.e.*, by writing down the equations on a paper).
- This doesn't seem good idea since
  - You need lots of matrix calculus (and paper)
  - What if you want to change the loss? you need to start from beginning
  - Not feasible for complex models.
- *Back-propagation* is the solution.
  - compute partial derivatives for each components
  - combine the partial derivaties via the chain rule, *e.g.*, for $\frac{\partial (f \circ g)(x)}{\partial x} = \frac{\partial f(z)}{\partial z} \frac{\partial g(x)}{\partial x}$, where $z = g(x)$

Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

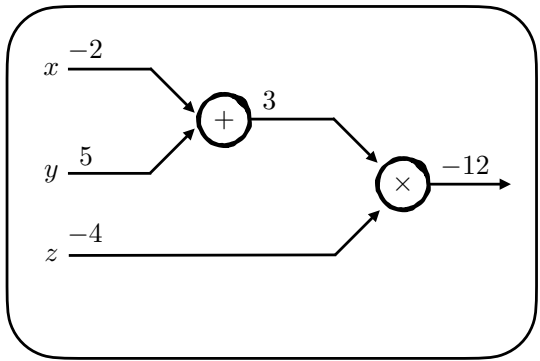


Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

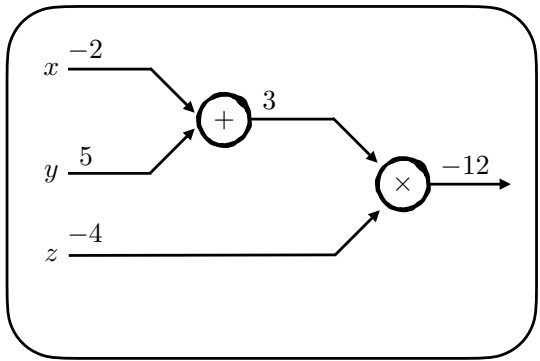
$$q = x + y$$

$$f = qz$$



Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$



$$q = x + y$$

$$f = qz$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$

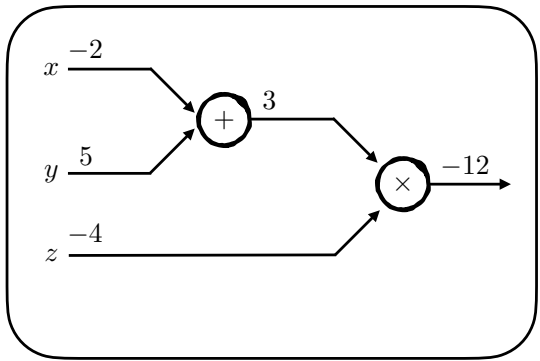
Backpropagation: Simple example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



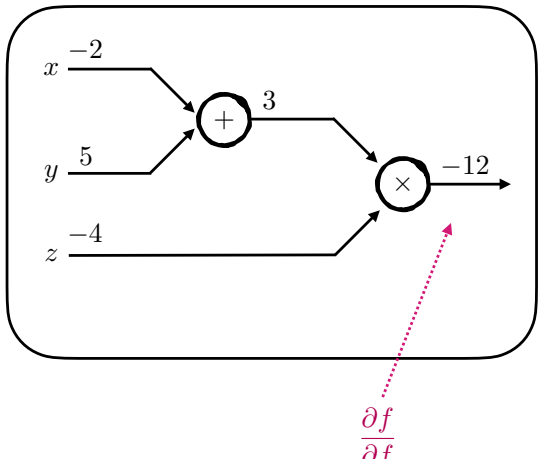
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



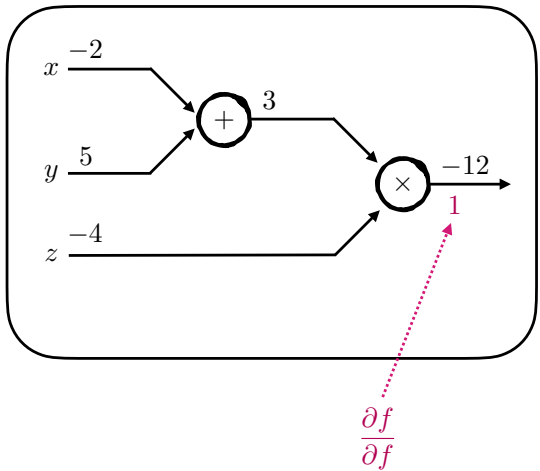
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



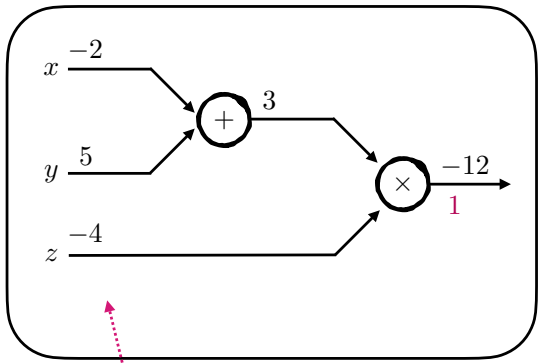
Backpropagation: Simple Example

$f(x, y, z) = (x + y)z$ Let $x = -2$, $y = 5$, $z = -4$

$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$

$f = qz \quad \frac{\partial f}{\partial q} = z \quad \boxed{\frac{\partial f}{\partial z} = q}$

$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$



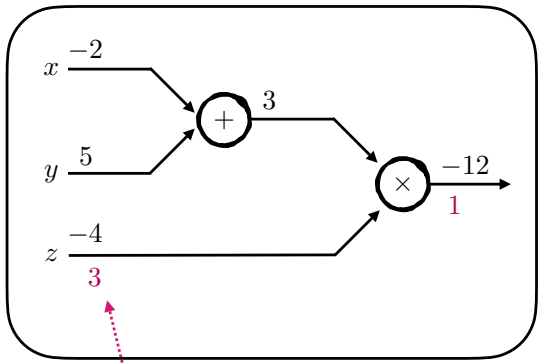
Backpropagation: Simple Example

$$f(x,y,z) = (x+y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \boxed{\frac{\partial f}{\partial z} = q}$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



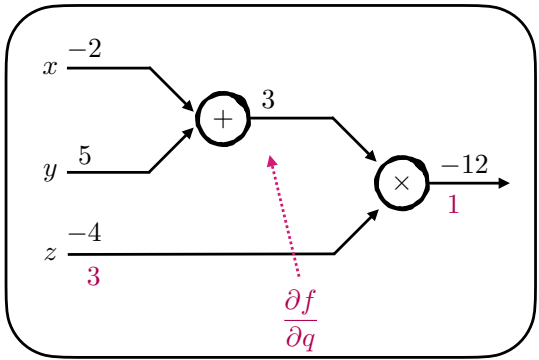
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \boxed{\frac{\partial f}{\partial q} = z} \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



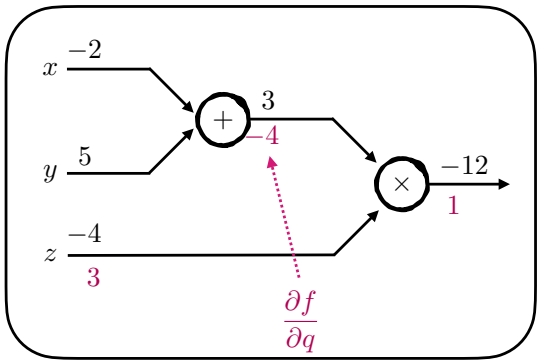
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{matrix} x = -2 \\ y = 5 \\ z = -4 \end{matrix}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \boxed{\frac{\partial f}{\partial q} = z} \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



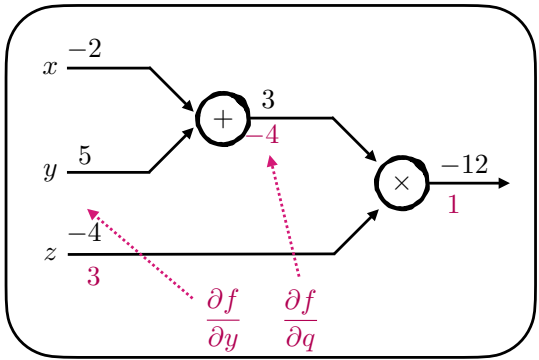
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



Chain rule: $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$

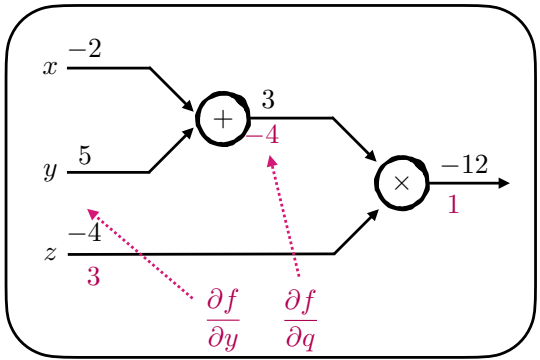
Backpropagation: Simple Example

$f(x, y, z) = (x + y)z$ Let $x = -2$, $y = 5$, $z = -4$

$q = x + y$ $\quad \frac{\partial q}{\partial x} = 1$ $\quad \boxed{\frac{\partial q}{\partial y} = 1}$

$f = qz$ $\quad \boxed{\frac{\partial f}{\partial q} = z}$ $\quad \frac{\partial f}{\partial z} = q$

$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$



Chain rule: $\frac{\partial f}{\partial y} = \boxed{\frac{\partial f}{\partial q}}\boxed{\frac{\partial q}{\partial y}}$

Upstream Gradient — Downstream Gradient

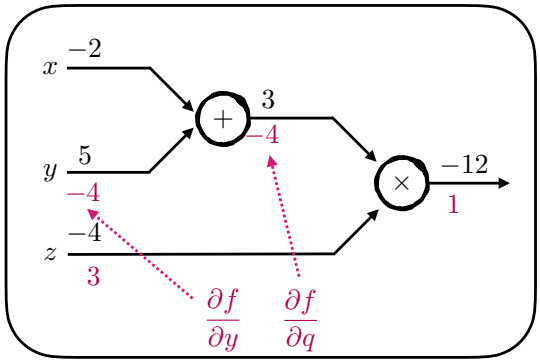
Backpropagation: Simple Example

$f(x, y, z) = (x + y)z$ Let $x = -2$, $y = 5$, $z = -4$

$q = x + y$ $\quad \frac{\partial q}{\partial x} = 1$ $\quad \boxed{\frac{\partial q}{\partial y} = 1}$

$f = qz$ $\quad \boxed{\frac{\partial f}{\partial q} = z}$ $\quad \frac{\partial f}{\partial z} = q$

$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$



Chain rule: $\frac{\partial f}{\partial y} = \boxed{\frac{\partial f}{\partial q}}\boxed{\frac{\partial q}{\partial y}}$

Upstream Gradient — Downstream Gradient

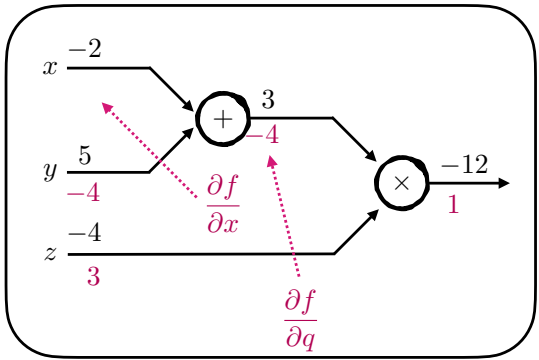
Backpropagation: Simple Example

$$f(x, y, z) = (x + y)z \quad \text{Let} \quad \begin{aligned} x &= -2 \\ y &= 5 \\ z &= -4 \end{aligned}$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$$



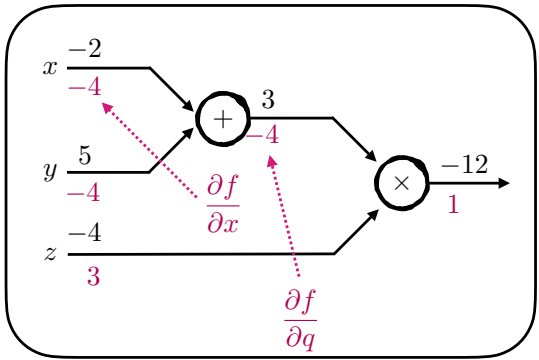
Backpropagation: Simple Example

$f(x, y, z) = (x + y)z$ Let $x = -2$, $y = 5$, $z = -4$

$q = x + y$ $\quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$

$f = qz$ $\quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$

$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} = ?$



Chain rule: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$

Upstream Gradient | Downstream Gradient

Key Messages So Far?

- We need one forward pass for backpropagation.
- We can compute the derivatie of a complex function if
  - each “computational component” has a closed form derivative and
  - the complex function consists of a directed acyclic graph of computational components.

Coming Back to NN

Remark

All we need in learning is the partial derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_l}$ for all $l = 1, ..., L$.

- Let's compute $\partial \mathcal{L} / \partial \boldsymbol{W}_L$ first via chain-rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_L} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \frac{\partial \mathbf{h}_L}{\partial \boldsymbol{W}_L}$$

Coming Back to NN

Remark

All we need in learning is the partial derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_l}$ for all $l = 1, ..., L$.

- Let's compute $\partial \mathcal{L} / \partial \boldsymbol{W}_L$ first via chain-rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_L} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \frac{\partial \mathbf{h}_L}{\partial \boldsymbol{W}_L}$$

- For $\partial \mathcal{L} / \partial \boldsymbol{W}_{L-1}$, we need to compute:

We can reuse this computed above.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{L-1}} &= \overbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{h}_L}} \frac{\partial \mathbf{h}_L}{\partial \boldsymbol{W}_{L-1}} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \underbrace{\frac{\partial \mathbf{h}_L}{\partial \mathbf{h}_{L-1}}} \frac{\partial \mathbf{h}_{L-1}}{\partial \boldsymbol{W}_{L-1}} \end{aligned}$$

We need to compute partial derivative w.r.t input of layer L.

Coming Back to NN

Remark

All we need in learning is the partial derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_l}$ for all $l = 1, ..., L$.

- Let's compute $\partial \mathcal{L} / \partial \boldsymbol{W}_L$ first via chain-rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_L} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \frac{\partial \mathbf{h}_L}{\partial \boldsymbol{W}_L}$$

- For $\partial \mathcal{L} / \partial \boldsymbol{W}_{L-1}$, we need to compute:

We can reuse this computed above.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{L-1}} &= \overbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{h}_L}} \frac{\partial \mathbf{h}_L}{\partial \boldsymbol{W}_{L-1}} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \underbrace{\frac{\partial \mathbf{h}_L}{\partial \mathbf{h}_{L-1}}} \frac{\partial \mathbf{h}_{L-1}}{\partial \boldsymbol{W}_{L-1}} \end{aligned}$$

We need to compute partial derivative w.r.t input of layer L.

- We can do the (stochastic) gradient descent with the gradient values obtained.

Loss Functions: MSE

Regression

The mean squared error (MSE):

$$\frac{1}{2N}\sum_{n=1}^{N}\left\|\mathbf{y}_n-\widehat{\mathbf{f}}_n\right\|^2$$

- $\mathbf{f}(\mathbf{x};\theta)$: a neural network parameterized by $\boldsymbol{\theta}$, which takes $\mathbf{x}$ as input.
- $\hat{\mathbf{f}}_n \coloneqq \mathbf{f}(\mathbf{x}_n;\theta)$
- How about classification? Cross entropy loss!

Loss Functions: Cross Entropy Loss

Softmax Layer for Multi-Class Classification

$$\hat{p}_k = \frac{\exp(\hat{f}_k)}{\sum_{j=1}^{K} \exp(\hat{f}_j)},$$

where $\hat{\mathbf{f}} = [\hat{f}_1, \ldots, \hat{f}_K]^T$

Cross Entropy

Definition

Cross entropy between two probability distributions p and q (over the same underlying set of events) measures a distance between two distributions as follows:

$$H(p, q) = \mathbb{E}_p[-\log q] = -\sum_x p(x) \log q(x).$$

Cross Entropy for Binary Classification

Recall that, in the case of logistic regression, we have

$$p(k) \in \{y, 1-y\}, \quad q(k) \in \{\hat{p}, 1-\hat{p}\},$$

where k is the label. Then, the cross entropy error is calculated as

$$\mathcal{L} = \sum_{n=1}^{N} \left[-y_n \log \hat{p}_n - (1-y_n) \log(1-\hat{p}_n) \right].$$

Cross Entropy for Multiclass Classification

In the case of softmax regression, we have

$$p(k) \in \{y_1, y_2, \ldots, y_K\}, \quad q(k) \in \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_K\}.$$

Then, the cross entropy error is calculated as

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} [-y_{k,n} \log \hat{p}_{k,n}].$$

Outline

Conclusion

- Now we know the definition of neural nets, learning objectives, and their learning algorithm.

$$\min_{\theta} \sum_{i=1}^{N} \ell(f_\theta, x_i, y_i)$$

- What is an attack method for a neural net?

$$\max_{\delta} \ell(f_\theta, x + \delta, y)$$

- Learning and attacks have opposite objectives.
- How to solve the maximization?