UNIVERSITY
OF OSLO

IN4310 Introduction to Neural Networks

Ali Ramezani-Kebrya

ali@uio.no

Key Learning Goals [1, 2]

Understand logistic regression

Learn structure of dense feedforward neural networks

Understand forward pass for dense feedforward neural networks

Understand the cross-entropy loss

Outline

Recap: What Is Machine/Deep Learning?

Classification by Logistic Regression

Artificial Neurons

Neural Networks

Machine Learning (ML) and Its Components

Learning: converting experience into expertise or knowledge

ML: automated learning of meaningful patterns in data

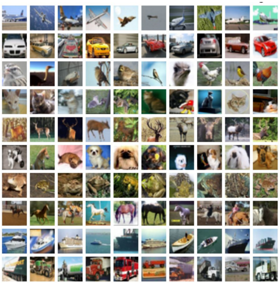
- In ML, our goal is to label unseen data (generalization matters)

Main components of ML systems:

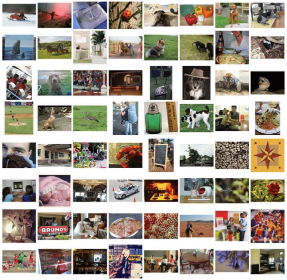
- Dataset
- Model (hypothesis class)
- Learning algorithm

[Shalev-Shwartz & Ben-David, 2014]

Supervised Learning: Multi-class Image Classification



CIFAR-10: 60000 32x32 colored images (3 channels) from 10 classes [Krizhevsky, 2009]



ImageNet: 14 million colored images (varying resolution) from 21K classes [Deng et al., 201]

Goal: For an image-label pair $(\mathbf{x}, y)$ drawn from an unknown distribution, find a classifier $h \in \mathcal{H}$ with minimum misclassification probability

$$\min_{h \in \mathcal{H}} \Pr(h(\mathbf{x}) \neq y)$$

Empirical Risk Minimization (ERM)

Let $h_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$ denote a predictor parameterized by $\mathbf{w} \in \mathbb{R}^d$. ERM:

$$\min_{\mathbf{w}} \left\{ R_n(\mathbf{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i) \right\}$$

where $\ell(h_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$ is the loss on sample i, i.e., $(\mathbf{x}_i, \mathbf{y}_i)$.

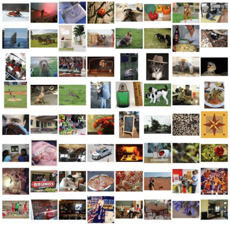
Some frequently used loss functions ℓ:

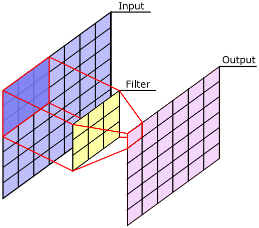
Logistic loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \log(1 + \exp(-\mathbf{y} \cdot h_{\mathbf{w}}(\mathbf{x})))$

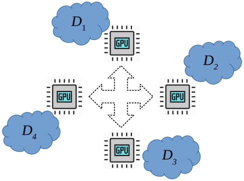
Quadratic loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \|\mathbf{y} - h_{\mathbf{w}}(\mathbf{x})\|_2^2$

Hinge loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \max(0, 1 - \mathbf{y} \cdot h_{\mathbf{w}}(\mathbf{x}))$

Main Components in Deep Learning Pipeline







Massive datasets

Inductive bias from large-scale and complex architectures

Nonconvex ERM via SGD with multi-GPU systems

Outline

Recap: What Is Machine/Deep Learning?

Classification by Logistic Regression

Artificial Neurons

Neural Networks

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w},\ f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \ f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w},\ \ f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \;\; f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \; f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \;\; f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

For every input sample $\mathbf{x} \in \mathcal{X}$, correctly predict the class y

Binary classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

Initial idea (from last lecture): Apply a linear mapping and classify according to the sign of the output:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w},\ f_{\mathbf{w}}(\mathbf{x}) = \mathrm{sgn}(h_{\mathbf{w}}(\mathbf{x}))$$

While $h_{\mathbf{w}}(\mathbf{x})$ has unbounded values, $f_{\mathbf{w}}(\mathbf{x})$ is either -1 or 1, but:

if $h_{\mathbf{w}}(\mathbf{x}) \approx 0$, we should be uncertain about the prediction

if $h_{\mathbf{w}}(\mathbf{x}) \gg 0$, we should be confident about the prediction 1

if $h_{\mathbf{w}}(\mathbf{x}) \ll 0$, we should be confident about the prediction -1

Classification

Encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$

$h_{\mathbf{w}}(\mathbf{x}) \approx 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0.5$
$h_{\mathbf{w}}(\mathbf{x}) \gg 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 1$, in particular let $\lim_{u \to +\infty} s(u) = 1$
$h_{\mathbf{w}}(\mathbf{x}) \ll 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0$, in particular let $\lim_{u \to -\infty} s(u) = 0$

Interpret $s(u)$ as a probability over inputs u

There are many possible choices for the function $s(u)$

Classification

Encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$

$h_{\mathbf{w}}(\mathbf{x}) \approx 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0.5$
$h_{\mathbf{w}}(\mathbf{x}) \gg 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 1$, in particular let $\lim_{u \to +\infty} s(u) = 1$
$h_{\mathbf{w}}(\mathbf{x}) \ll 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0$, in particular let $\lim_{u \to -\infty} s(u) = 0$

Interpret $s(u)$ as a probability over inputs u

There are many possible choices for the function $s(u)$

Classification

Encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$

$h_{\mathbf{w}}(\mathbf{x}) \approx 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0.5$
$h_{\mathbf{w}}(\mathbf{x}) \gg 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 1$, in particular let $\lim_{u \to +\infty} s(u) = 1$
$h_{\mathbf{w}}(\mathbf{x}) \ll 0$ to $s(h_{\mathbf{w}}(\mathbf{x})) \approx 0$, in particular let $\lim_{u \to -\infty} s(u) = 0$

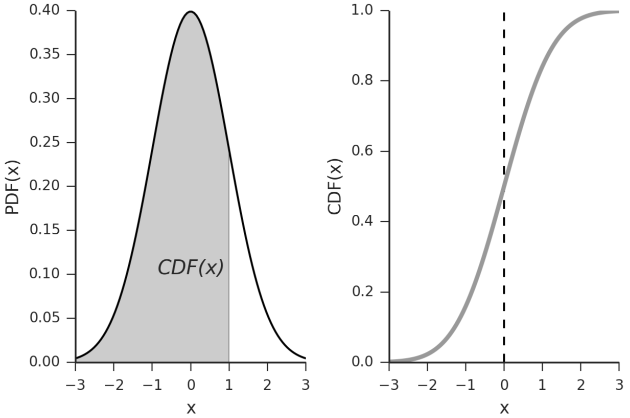
Interpret $s(u)$ as a probability over inputs u

There are many possible choices for the function $s(u)$

Probit (Probability and Unit) Regression

Use the CDF of any distribution with a median of 0

Standard normal distribution is used in probit regression



Logistic Regression

Use a simpler function called the logistic sigmoid function:

Definition: Logistic sigmoid function

$$s(u) = \frac{\exp(u)}{1+\exp(u)} = \frac{\exp(u)}{1+\exp(u)}\frac{\exp(-u)}{\exp(-u)} = \frac{1}{\exp(-u)+1}$$

Note that:

$$s(0) = \frac{1}{\exp(-0)+1} = \frac{1}{1+1} = 0.5$$

$$\lim_{u\to\infty} s(u) = \lim_{u\to\infty} \frac{1}{\exp(-u)+1} = \frac{1}{0+1} = 1$$

$$\lim_{u\to-\infty} s(u) = \lim_{u\to-\infty} \frac{\exp(u)}{1+\exp(u)} = \frac{0}{1+0} = 0$$

Logistic Regression

Use a simpler function called the logistic sigmoid function:

Definition: Logistic sigmoid function

$$s(u) = \frac{\exp(u)}{1+\exp(u)} = \frac{\exp(u)}{1+\exp(u)}\frac{\exp(-u)}{\exp(-u)} = \frac{1}{\exp(-u)+1}$$

Note that:

$$s(0) = \frac{1}{\exp(-0)+1} = \frac{1}{1+1} = 0.5$$

$$\lim_{u\to\infty} s(u) = \lim_{u\to\infty} \frac{1}{\exp(-u)+1} = \frac{1}{0+1} = 1$$

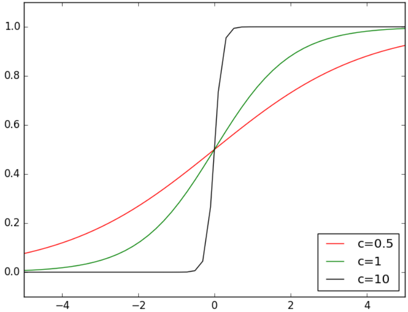
$$\lim_{u\to-\infty} s(u) = \lim_{u\to-\infty} \frac{\exp(u)}{1+\exp(u)} = \frac{0}{1+0} = 0$$

Logistic Regression

How does the logistic sigmoid function look?

Let's plot it for different scalings $c > 0$:

$$s(cu) = \frac{\exp(cu)}{1 + \exp(cu)} = \frac{1}{\exp(-cu) + 1}$$



Logistic regression

Definition: Logistic regression model

Plug a linear mapping $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ into sigmoid function $s : \mathbb{R} \to [0, 1]$

$$s(h_{\mathbf{w}}(\mathbf{x})) = \frac{\exp(\mathbf{x}^\top \mathbf{w})}{1 + \exp(\mathbf{x}^\top \mathbf{w})} = \frac{1}{\exp(-\mathbf{x}^\top \mathbf{w}) + 1}$$

For binary classification, the output $s(h_{\mathbf{w}}(\mathbf{x})) \in [0, 1]$

$s(h_{\mathbf{w}}(\mathbf{x}))$ is the predicted probability $\Pr(\{y = 1 | \mathbf{x}\})$.

Logit: Logarithm of Odds

Logistic model may also be called a logit (**log**istic and un**it**) model

Logit function (maps $(0,1) \to (-\infty, +\infty)$) is the logarithm of the odds

$$\operatorname{logit}(p) = \log \frac{p}{1-p}$$

Logit function is the inverse of logistic sigmoid function

$$s(\operatorname{logit}(p)) = \frac{\exp(\operatorname{logit}(p))}{1+\exp(\operatorname{logit}(p))} = \frac{\frac{p}{1-p}}{1+\frac{p}{1-p}} = \frac{p}{1-p+p} = \frac{p}{1} = p$$

Logistic regression estimates the odds ratio between two events

Logit: Logarithm of Odds

Logistic model may also be called a logit (**log**istic and un**it**) model

Logit function (maps $(0,1) \rightarrow (-\infty, +\infty)$) is the logarithm of the odds

$$\operatorname{logit}(p) = \log \frac{p}{1-p}$$

Logit function is the inverse of logistic sigmoid function

$$s(\operatorname{logit}(p)) = \frac{\exp(\operatorname{logit}(p))}{1 + \exp(\operatorname{logit}(p))} = \frac{\frac{p}{1-p}}{1 + \frac{p}{1-p}} = \frac{p}{1-p+p} = \frac{p}{1} = p$$

Logistic regression estimates the odds ratio between two events

Logit: Logarithm of Odds

Logistic model may also be called a logit (**log**istic and un**it**) model

Logit function (maps $(0, 1) \to (-\infty, +\infty)$) is the logarithm of the odds

$$\operatorname{logit}(p) = \log \frac{p}{1-p}$$

Logit function is the inverse of logistic sigmoid function

$$s(\operatorname{logit}(p)) = \frac{\exp(\operatorname{logit}(p))}{1+\exp(\operatorname{logit}(p))} = \frac{\frac{p}{1-p}}{1+\frac{p}{1-p}} = \frac{p}{1-p+p} = \frac{p}{1} = p$$

Logistic regression estimates the odds ratio between two events

Logit: Logarithm of Odds

Logistic model may also be called a logit (**log**istic and un**it**) model

Logit function (maps $(0, 1) \rightarrow (-\infty, +\infty)$) is the logarithm of the odds

$$\text{logit}(p) = \log \frac{p}{1-p}$$

Logit function is the inverse of logistic sigmoid function

$$s(\text{logit}(p)) = \frac{\exp(\text{logit}(p))}{1 + \exp(\text{logit}(p))} = \frac{\frac{p}{1-p}}{1 + \frac{p}{1-p}} = \frac{p}{1 - p + p} = \frac{p}{1} = p$$

Logistic regression estimates the odds ratio between two events

Log Loss Function for Binary Classification

$\hat{p}_{\mathbf{w}}(1|\mathbf{x}) = s(h_{\mathbf{w}}(\mathbf{x})) = \frac{\exp(\mathbf{x}^\top \mathbf{w})}{1+\exp(\mathbf{x}^\top \mathbf{w})}$ what we estimate not the exact prob.

$\hat{p}_{\mathbf{w}}(-1|\mathbf{x}) = 1 - s(h_{\mathbf{w}}(\mathbf{x})) = \frac{1}{1+\exp(\mathbf{x}^\top \mathbf{w})}$

Output: $[\hat{p}_{\mathbf{w}}(-1|\mathbf{x}) \quad \hat{p}_{\mathbf{w}}(1|\mathbf{x})]^\top$

If $y = 1$, then $\ell(\mathbf{w}) = -\log\left(\hat{p}_{\mathbf{w}}(1|\mathbf{x})\right)$

If $y = -1$, then $\ell(\mathbf{w}) = -\log\left(\hat{p}_{\mathbf{w}}(-1|\mathbf{x})\right)$

If $(\mathbf{x}_n, y_n)$ is a training sample, then $\ell_n(\mathbf{w}) = -\log\left(\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n)\right)$

Log Loss Function for Binary Classification

If $(\mathbf{x}_n, y_n)$ is a training sample, then $\ell_n(\mathbf{w}) = -\log\big(\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n)\big)$

$$[\hat{p}_{\mathbf{w}}(-1|\mathbf{x}_n) \quad \hat{p}_{\mathbf{w}}(1|\mathbf{x}_n)] = [0.8 \quad 0.2]$$

If true label $y_n = 1$, then $\ell_n(\mathbf{w}) = -\log(0.2) \approx 1.61$

If true label $y_n = -1$, then $\ell_n(\mathbf{w}) = -\log(0.8) \approx 0.22$

To have smaller loss, give higher probability to true label

$$[\hat{p}_{\mathbf{w}}(-1|\mathbf{x}_n) \quad \hat{p}_{\mathbf{w}}(1|\mathbf{x}_n)] = [0.9999 \quad 0.0001]$$

$$\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n) = s(y_n h_{\mathbf{w}}(\mathbf{x})) = \frac{\exp(y_n \mathbf{x}^\top \mathbf{w})}{1 + \exp(y_n \mathbf{x}^\top \mathbf{w})}$$

$$\ell_n(\mathbf{w}) = -\log\big(\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n)\big) = \log\big(1 + \exp(-y_n \mathbf{x}_n^\top \mathbf{w})\big)$$

Derivation of cross-entropy loss function

Interpret $s(h_{\mathbf{w}}(\mathbf{x}))$ as $\hat{p}_{\mathbf{w}}(y=1|\mathbf{x})$ predicted by our model

Given samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, our goal is to find parameters $\mathbf{w}$

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right)$$

Derivation of cross-entropy loss function

Interpret $s(h_{\mathbf{w}}(\mathbf{x}))$ as $\hat{p}_{\mathbf{w}}(y=1|\mathbf{x})$ predicted by our model

Given samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, our goal is to find parameters $\mathbf{w}$

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right)$$

Derivation of cross-entropy loss function

Interpret $s(h_{\mathbf{w}}(\mathbf{x}))$ as $\hat{p}_{\mathbf{w}}(y = 1|\mathbf{x})$ predicted by our model

Given samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, our goal is to find parameters $\mathbf{w}$

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right)$$

Log Loss Function

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right)$$

Assuming $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are independent and identically distributed

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right) = \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_n | \mathbf{x}_n)$$

Maximize likelihood as a function of parameters $\mathbf{w}$

Log Loss Function

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right)$$

Assuming $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are independent and identically distributed

$$\Pr\left(\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\}\right) = \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_n | \mathbf{x}_n)$$

Maximize likelihood as a function of parameters $\mathbf{w}$

Log Loss Function

If our logistic regression model was true, then the joint probability of observing the class labels would be:

$$\Pr\left(\left\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\right\}\right)$$

Assuming $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are independent and identically distributed

$$\Pr\left(\left\{y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}\right\}\right) = \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_n | \mathbf{x}_n)$$

Maximize likelihood as a function of parameters $\mathbf{w}$

Cross-entropy Loss Function

Maximum likelihood estimates of $\mathbf{w}$ given data

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$$

If $y_i = 1$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is our model output $s(h_{\mathbf{w}}(\mathbf{x}_i))$

If $y_i = 0$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is $1 - s(h_{\mathbf{w}}(\mathbf{x}_i))$

For $y_i \in \{0, 1\}$

$$\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n) = s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i}\big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}$$

Cross-entropy Loss Function

Maximum likelihood estimates of $\mathbf{w}$ given data

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$$

If $y_i = 1$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is our model output $s(h_{\mathbf{w}}(\mathbf{x}_i))$

If $y_i = 0$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is $1 - s(h_{\mathbf{w}}(\mathbf{x}_i))$

For $y_i \in \{0, 1\}$

$$\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n) = s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i}\big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}$$

Cross-entropy Loss Function

Maximum likelihood estimates of $\mathbf{w}$ given data

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$$

If $y_i = 1$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is our model output $s(h_{\mathbf{w}}(\mathbf{x}_i))$

If $y_i = 0$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is $1 - s(h_{\mathbf{w}}(\mathbf{x}_i))$

For $y_i \in \{0, 1\}$

$$\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n) = s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i}\big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}$$

Cross-entropy Loss Function

Maximum likelihood estimates of $\mathbf{w}$ given data

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} \hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$$

If $y_i = 1$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is our model output $s(h_{\mathbf{w}}(\mathbf{x}_i))$

If $y_i = 0$, then $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$ is $1 - s(h_{\mathbf{w}}(\mathbf{x}_i))$

For $y_i \in \{0, 1\}$

$$\hat{p}_{\mathbf{w}}(y_n|\mathbf{x}_n) = s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \Big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\Big)^{1-y_i}$$

Cross-entropy Loss Function

Maximum likelihood estimates

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}$$

Take the logarithm does not change the location of the maximum

$$\begin{aligned}\mathbf{w}^\star &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \log\Big(s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}\Big) \\ &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \log\big((s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i}\big) + \log\Big(\big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}\Big) \\ &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) + (1 - y_i)\log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i)))\end{aligned}$$

Cross-entropy Loss Function

Maximum likelihood estimates

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i}$$

Take the logarithm does not change the location of the maximum

$$\begin{aligned} \mathbf{w}^\star &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \log \left(s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i} \right) \\ &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \log \big((s(h_{\mathbf{w}}(\mathbf{x}_i))^{y_i} \big) + \log \left(\big(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))\big)^{1-y_i} \right) \\ &= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) + (1 - y_i) \log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))) \end{aligned}$$

Cross-entropy Loss Function

Equivalent problem

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} -y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) - (1 - y_i) \log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i)))$$

This expression is called the cross-entropy loss of our data points

Cross-entropy Loss Function

Equivalent problem

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} -y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) - (1-y_i)\log(1-s(h_{\mathbf{w}}(\mathbf{x}_i)))$$

This expression is called the cross-entropy loss of our data points

Cross-entropy loss

Cross-entropy loss for binary classification

The cross-entropy loss of a data point (x_i, y_i) is defined as:

$$\ell_i(\mathbf{w}) = -y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) - (1 - y_i)\log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i)))$$

For one-hot labels $y_1, \dots, y_n \in \{0, 1\}$, the cross-entropy loss of a data point (x_i, y_i) is the negative logarithm of the predicted probability of the ground-truth class

$$\ell_i(\mathbf{w}) = \begin{cases} -\log(s(h_{\mathbf{w}}(\mathbf{x}_i)) & \text{if } y_i = 1 \\ -\log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))) & \text{if } y_i = 0 \end{cases}$$

Does such a log-probability make sense as a loss?

Cross-entropy loss

Cross-entropy loss for binary classification

The cross-entropy loss of a data point (x_i, y_i) is defined as:

$$\ell_i(\mathbf{w}) = -y_i \log(s(h_{\mathbf{w}}(\mathbf{x}_i)) - (1 - y_i) \log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i)))$$

For one-hot labels $y_1, \ldots, y_n \in \{0, 1\}$, the cross-entropy loss of a data point (x_i, y_i) is the negative logarithm of the predicted probability of the ground-truth class

$$\ell_i(\mathbf{w}) = \begin{cases} -\log(s(h_{\mathbf{w}}(\mathbf{x}_i)) & \text{if } y_i = 1 \\ -\log(1 - s(h_{\mathbf{w}}(\mathbf{x}_i))) & \text{if } y_i = 0 \end{cases}$$

Does such a log-probability make sense as a loss?

Classification via Logistic Regression

Key takeaways

Logistic regression is a method for classification problem

Apply sigmoid to a linear mapping

Output can be interpreted as the predicted probability $\hat{p}_{\mathbf{w}}(y_i|\mathbf{x}_i)$

Cross-entropy loss derived by maximum likelihood estimation of $\mathbf{w}$

Cross-entropy loss is the sum of the negative logarithms of the predicted probabilities weighted by the ground-truth distribution

For one-hot labels, the negative logarithm of the predicted probability of the ground-truth class

Outline

Recap: What Is Machine/Deep Learning?

Classification by Logistic Regression

Artificial Neurons

Neural Networks

What is an Artificial Neuron?

A mathematical function that can

- Take some inputs $\{x_i\}_{i=1}^{n}$
- Weighted sum of inputs with weights w_{ij} and bias b_j
- Apply some (non-linear) activation function g
- Output (neuron's activation) a_j

Forward pass:

$$z_j = \sum_{i=1}^{n} x_i w_{ij} + b_j \qquad \text{linear activation}$$

$$a_j = g(z_j) = g\Big(\sum_{i=1}^{n} x_i w_{ij} + b_j\Big) \qquad \text{non-linear activation}$$

Activation Functions

A function $g : \mathbb{R} \rightarrow \mathbb{R}$

Introduces non-linearity

Should be differentiable to use gradient-based optimisation

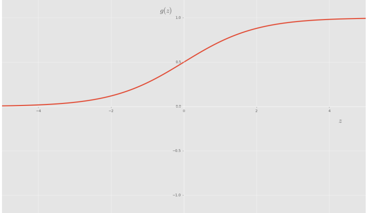
Typically prevents or reduces weak signals from passing through

Determines what fires the neuron for different activations

Sigmoid Activation (Smooth)

Logistic sigmoid function

$$g(x) = \frac{1}{\exp(-x) + 1}$$



Historically popular

Currently rarely used

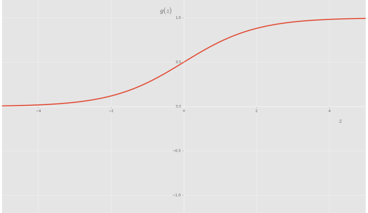
Suffer from gradient saturation: derivative gets exponentially small for large $|x|$:

$$\frac{\partial s(x)}{\partial x} = \frac{e^{-x}}{(e^{-x}+1)^2} \leq \begin{cases} e^{-x} & x \geq 0 \\ \frac{1}{e^{-x}+1} & x < 0 \end{cases}$$

Sigmoid Activation (Smooth)

Logistic sigmoid function

$$g(x) = \frac{1}{\exp(-x) + 1}$$



Historically popular

Currently rarely used

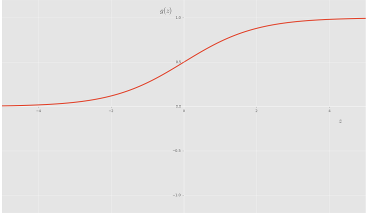
Suffer from gradient saturation: derivative gets exponentially small for large $|x|$:

$$\frac{\partial s(x)}{\partial x} = \frac{e^{-x}}{(e^{-x} + 1)^2} \leq \begin{cases} e^{-x} & x \geq 0 \\ \frac{1}{e^{-x}+1} & x < 0 \end{cases}$$

Sigmoid Activation (Smooth)

Logistic sigmoid function

$$g(x) = \frac{1}{\exp(-x) + 1}$$



Historically popular

Currently rarely used

Suffer from gradient saturation: derivative gets exponentially small for large $|x|$:

$$\frac{\partial s(x)}{\partial x} = \frac{e^{-x}}{(e^{-x}+1)^2} \leq \begin{cases} e^{-x} & x \geq 0 \\ \frac{1}{e^{-x}+1} & x < 0 \end{cases}$$

Rectified Linear Unit (ReLU) Activation (Non-smooth)

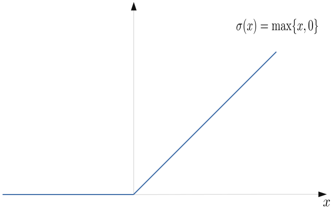
Set negative values to 0

$$g(x) = \max(0, x)$$

Gradient do not saturate

Not differentiable in 0

Define subgradient to be 1



The gradient is 0 for negative inputs

Might cause inactive nodes to remain inactive

Might still work fine in practice

Alternatives with non-zero gradient: leaky ReLU: $g(x) = \max(0.01x, x)$, Gaussian Error Linear Units (GELUs), ELU

Popular particularly for convolutional neural networks

Rectified Linear Unit (ReLU) Activation (Non-smooth)

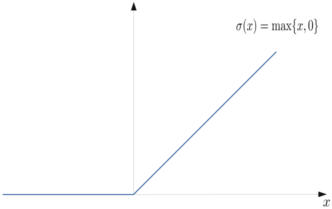
Set negative values to 0

$$g(x) = \max(0, x)$$

Gradient do not saturate

Not differentiable in 0

Define subgradient to be 1



The gradient is 0 for negative inputs

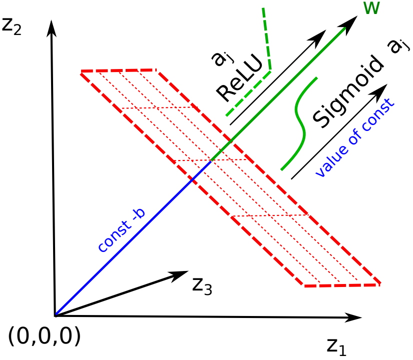
Might cause inactive nodes to remain inactive

Might still work fine in practice

Alternatives with non-zero gradient: leaky ReLU: $g(x) = \max(0.01x, x)$, Gaussian Error Linear Units (GELUs), ELU

Popular particularly for convolutional neural networks

What Does a Neuron Do?



Weights $\mathbf{w}$ define the orientation of the plane

Bias b defines the position of the plane

Output of activation function is constant in the red plane

Output of activation function varies along the direction of $\mathbf{w}$

Outline

Recap: What Is Machine/Deep Learning?

Classification by Logistic Regression

Artificial Neurons

Neural Networks

Neural Network

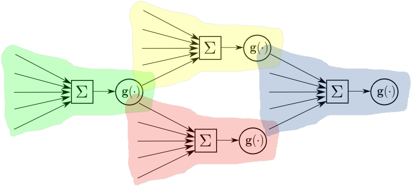
A directed graph structure made from connected neurons

A neuron can be input to many other neurons

A neuron can receive input to many other neurons

Each neuron has same structure $(g, \sum)$ with different parameters (w_{ij}, b_j)

Can stack neurons in layers

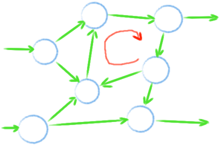


Neural Network

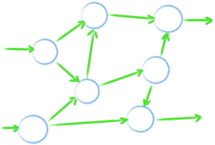
Definition: Neural Network

Any directed graph built from neurons

Two important types: recurrent and feedforward neural networks

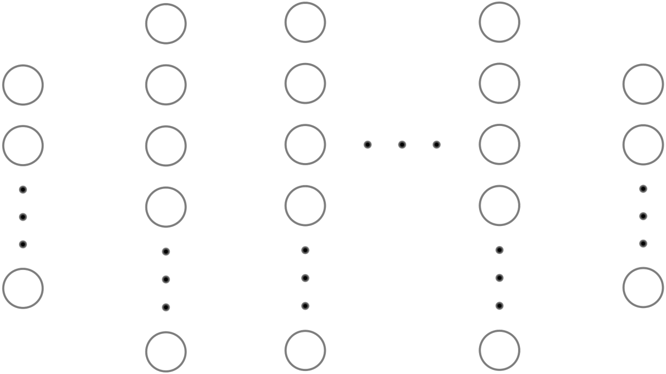


Recurrent (not covered in this lecture)
e.g. for speech processing

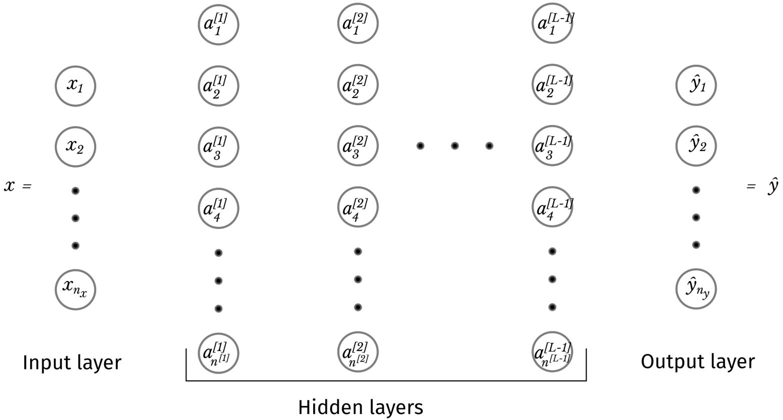


Feedforward
e.g. for image classification

Base Architecture

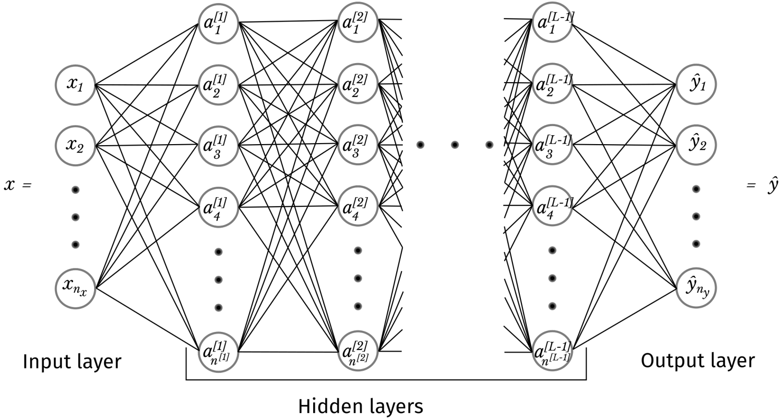


Nodes and Layers



Output layer: layer L (input layer typically does not count)

Dense (Fully-connected) Feedforward Neural Network



Notation — for Reference

Superscript w. brackets $[l]$: Layer l

L: Number of layers

$n^{[l]}$: Number of nodes in layer l

$n_x = n^{[0]}$: Input dimension

$n_y = n^{[L]}$: Number of classes

$\mathbf{x}$: Array of inputs

$\mathbf{y}$: Array of *true* outputs

$\hat{\mathbf{y}}$: Array of *predicted* output

$\mathbf{w}$: Edge weights

b: Node bias

$\mathbf{z}$: Linear combination of activations from previous layer

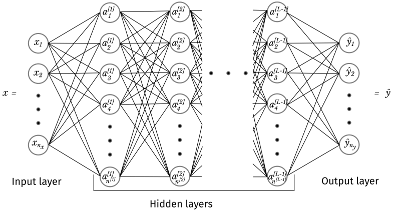
$a^{[l]}$: Node activation in layer l

$a^{[0]} = \mathbf{x}$: Input vector

$a^{[L]} = \hat{\mathbf{y}}$: Output vector

N: $\#$ samples in training dataset

General Formulation



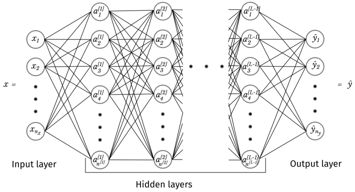
$$a_k^{[l]} = g\left(\mathbf{w}_k^{[l]} \cdot \mathbf{a}^{[l-1]} + b_k^{[l]}\right),$$

$$\text{output} = g(\text{inner product}(\text{input}, \mathbf{w}) + \text{bias})$$

$$k \in \{1, 2, \ldots, n^{[l]}\},\ l \in \{1, 2, \ldots, L\}$$

Input to layer l is output (activation) of layer $l - 1$

General formulation



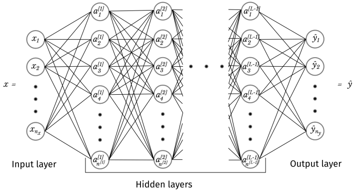
$$a_k^{[l]} = g\left(\mathbf{w}_k^{[l]} \cdot \mathbf{a}^{[l-1]} + b_k^{[l]}\right) = g(z_k^{[l]})$$

Output $a_k^{[l]}$ for neuron k in layer l is $g(z_k^{[l]})$ where $z_k^{[l]}$ is an inner product between

Activation vector $\mathbf{a}^{[l-1]}$ of layer $l-1$

k-th slice $\mathbf{w}_k^{[l]}$ of the weight matrix in layer l with an added bias term $b_k^{(l)}$

General Formulation



$$a_k^{[l]} = g\left(\mathbf{w}_k^{[l]} \cdot \mathbf{a}^{[l-1]} + b_k^{[l]}\right) = g(z_k^{[l]})$$

Concatenate all outputs $a_k^{[l]}$ of layer l and apply g element-wise

$$\mathbf{a}^{[l]} = g\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\right)$$

$\mathbf{W}^{[l]}\mathbf{a}^{[l-1]}$ is a matrix-vector multiplication

Hidden Layer Nodes

$$\mathbf{a}^{[l]} = g\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\right)$$

All $\mathbf{W}$'s and $\mathbf{b}$'s are "trainable" and will be adjusted

$\mathbf{a}^{[0]} = \mathbf{x}$ and $\mathbf{a}^{[L]} = \hat{\mathbf{y}}$

The network has L layers with $L-1$ hidden layers

Hidden Layer Nodes

$$\mathbf{a}^{[l]} = g\left(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\right)$$

All $\mathbf{W}$'s and $\mathbf{b}$'s are "trainable" and will be adjusted

$\mathbf{a}^{[0]} = \mathbf{x}$ and $\mathbf{a}^{[L]} = \hat{\mathbf{y}}$

The network has L layers with $L - 1$ hidden layers

Output Layer

An activation function at layer L will in general not provide outputs that can be interpreted as probabilities

For binary classification, apply sigmoid function as g

A generalisation of sigmoid called softmax function

$$s(\mathbf{x})_k = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}}$$

Converts linearly mapped values $z_1, \ldots, z_{n^{[L]}}$ to predicted probabilities

Output Layer

An activation function at layer L will in general not provide outputs that can be interpreted as probabilities

For binary classification, apply sigmoid function as g

A generalisation of sigmoid called softmax function

$$s(\mathbf{x})_k = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}}$$

Converts linearly mapped values $z_1, \ldots, z_{n^{[L]}}$ to predicted probabilities

Output Layer

An activation function at layer L will in general not provide outputs that can be interpreted as probabilities

For binary classification, apply sigmoid function as g

A generalisation of sigmoid called softmax function

$$s(\mathbf{x})_k = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}}$$

Converts linearly mapped values $z_1, \ldots, z_{n^{[L]}}$ to predicted probabilities

Output Layer

An activation function at layer L will in general not provide outputs that can be interpreted as probabilities

For binary classification, apply sigmoid function as g

A generalisation of sigmoid called softmax function

$$s(\mathbf{x})_k = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}}$$

Converts linearly mapped values $z_1, \ldots, z_{n^{[L]}}$ to predicted probabilities

Output Layer

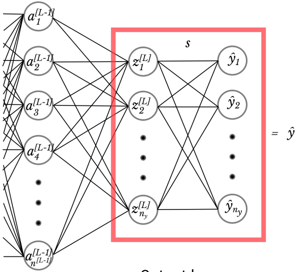
In output layer

$$a_k^{[L]} = s(\mathbf{z}^{[L]})_k$$
$$= \hat{y}_k$$

$k = 1, \ldots, n_y$ and $n_y = n^{[L]}$

$\mathbf{z}^{(L)}$ are called logits

$\hat{\mathbf{y}}$ interpreted as predicted probabilities



Output layer

Cross-entropy Loss

Model's predicted probabilities $\hat{\mathbf{y}}_i \in [0,1]^{n_y}$

Input sample $\mathbf{x}_i \in \mathbb{R}^{n_x}$ with true class label $\mathbf{y}_i \in [0,1]^{n_y}$

Definition: Cross-entropy loss

Cross-entropy loss between two probability distribution vectors $(y_1, \ldots, y_{n_y})$ and $(\hat{y}_1, \ldots, \hat{y}_{n_y})$:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\sum_{k=1}^{n_y} (\mathbf{y}_i)_k \log(\hat{\mathbf{y}}_i)_k$$

For one-hot labels, i.e. each $\mathbf{y}_i$ is 0 in all elements except one, negative logarithm of the predicted probability of the ground-truth class:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\log(\hat{\mathbf{y}}_i)_k \text{ where } \mathbf{y}_i = \mathbf{e}_k$$

Cross-entropy Loss

Model's predicted probabilities $\hat{\mathbf{y}}_i \in [0,1]^{n_y}$

Input sample $\mathbf{x}_i \in \mathbb{R}^{n_x}$ with true class label $\mathbf{y}_i \in [0,1]^{n_y}$

Definition: Cross-entropy loss

Cross-entropy loss between two probability distribution vectors $(y_1, \ldots, y_{n_y})$ and $(\hat{y}_1, \ldots, \hat{y}_{n_y})$:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\sum_{k=1}^{n_y} (\mathbf{y}_i)_k \log(\hat{\mathbf{y}}_i)_k$$

For one-hot labels, i.e. each $\mathbf{y}_i$ is 0 in all elements except one, negative logarithm of the predicted probability of the ground-truth class:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\log(\hat{\mathbf{y}}_i)_k \text{ where } \mathbf{y}_i = \mathbf{e}_k$$

ERM with Cross-entropy Loss

Use gradient descent to optimise the cross-entropy loss

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i | \boldsymbol{\theta})$$

We have a lot more parameters compared to logistic regression

$$\begin{aligned}\boldsymbol{\theta} &= \{w_{ij}^{[l]}, b_j^{[l]} : i \in \{1, \ldots, n^{[l-1]}\}, j \in \{1, \ldots, n^{[l]}\}, l \in \{1, \ldots, L\}\} \\ &= \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]} : l \in \{1, \ldots, L\}\}\end{aligned}$$

How to do this in practice will be the subject of a later lecture

ERM with Cross-entropy Loss

Use gradient descent to optimise the cross-entropy loss

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i | \boldsymbol{\theta})$$

We have a lot more parameters compared to logistic regression

$$\begin{aligned} \boldsymbol{\theta} &= \{w_{ij}^{[l]}, b_j^{[l]} : i \in \{1, \dots, n^{[l-1]}\}, j \in \{1, \dots, n^{[l]}\}, l \in \{1, \dots, L\}\} \\ &= \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]} : l \in \{1, \dots, L\}\} \end{aligned}$$

How to do this in practice will be the subject of a later lecture

Neural Network

Key takeaways

Logistic regression for binary classification is a neural net with one node in the output layer and 0 hidden layers

Stacking multiple layers gives a deep neural network

Each neuron (in hidden layers) in dense feedforward neural nets receives input from all neurons in the previous layer

Each neuron (in hidden layers) in dense feedforward neural nets sends its activation (output) to all neurons in the next layer

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data*. AMLBook New York, 2012.

[2] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.