What you will see today

more theory:

- recap: the inner product and its properties
- logistic sigmoid to map unbounded outputs onto $[0, 1]$ with an interpretation as probability
- cross-entropy loss
- connection between the cross-entropy loss and the principle of maximum likelihood

Outline

1. Important Intermezzo: the inner product
2. Classification by logistic regression
3. Binary cross-entropy and Maximum Likelihood
4. Softmax and Cross-entropy loss for multiple classes

Inner product properties I

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

- maps two real vectors u, v onto a real number $u \cdot v$
- linear in the first argument (u)

$$a \in \mathbb{R},\ (au) \cdot v = a(u \cdot v)$$
$$(u^{\{1\}} + u^{\{2\}}) \cdot v = u^{\{1\}} \cdot v + u^{\{2\}} \cdot v$$

  or in short:

$$(a_1 u^{\{1\}} + a_2 u^{\{2\}}) \cdot v = a_1(u^{\{1\}} \cdot v) + a_2(u^{\{2\}} \cdot v)$$

- linear in the second argument (v)

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

$$\text{Symmetry: } u \cdot v = v \cdot u$$
$$v \neq \mathbf{0} \; \Rightarrow v \cdot v > 0$$
$$\mathbf{0} \cdot u = 0$$

where $\mathbf{0}$ is the Null vector. For example in $\mathbb{R}^3$ this is the element $(0, 0, 0)$

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

has the following properties:

- it defines a norm $\|v\|$ is a norm, that is a notion of the length of a vector v:

$$v \cdot v = \|v\|^2$$

Inner product properties IV

$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

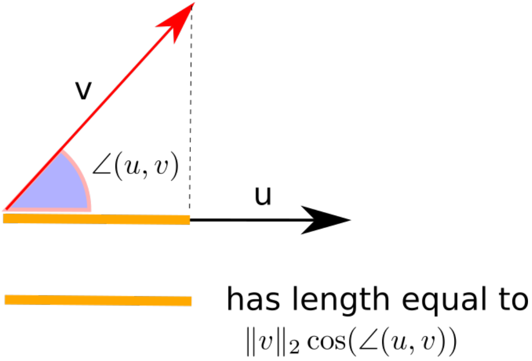
has the following properties:

- it defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2}(v \cdot v)^{1/2}} = \cos(\angle(u, v))$$

  - the angle can be measured in any dimensions
  - in higher dimensions, the angle is measured in the 2-dim plane spanned by u, v:

$$L(u, v) = \{a_0 u + a_1 v, a_0 \in \mathbb{R}, a_1 \in \mathbb{R}\}$$

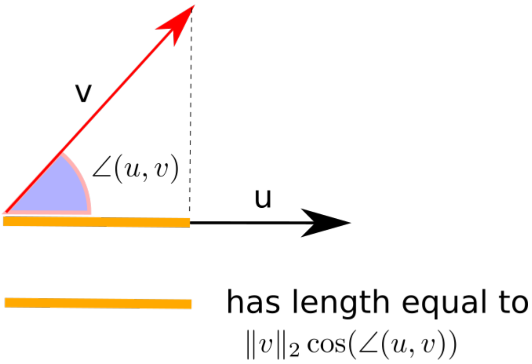


$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

- $u \cdot v$ defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2} (v \cdot v)^{1/2}} = \cos(\angle(u, v))$$

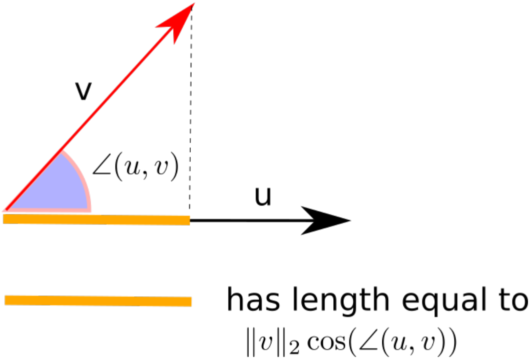
- for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product
  - lies in $[-1, +1]$
  - $u \cdot v = 1$ if $u = v$
  - gets close to 1 if their angle is close to zero,
  - gets close to 0 if their angle is close to $\pi/2 \sim 90$ deg,
  - $u \cdot v = -1$ if $u = -v$



$$u \cdot v = \sum_{k=0}^{d-1} u_d v_d \in \mathbb{R}$$

- $u \cdot v$ defines an angle between two vectors:

$$\frac{u \cdot v}{(u \cdot u)^{1/2}(v \cdot v)^{1/2}} = \cos(\angle(u, v))$$



Interpretation of the inner product

for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product computes a similarity measure between u and v based on their angle!

Interpretation of the inner product

for two vectors u, v of unit length ($\|u\|_2 = 1$) the inner product computes a similarity measure between u and v based on their angle!

next step:

- now we can use $u \cdot v$ to define a simple classifier:

Classification

$$f(x) = w \cdot x + b$$
$$s(x) = sign(f(x)) \in \{-1, +1\}$$

Mechanism:

- $f(x)$ is large if the angle $\angle(w, x)$ is close to zero,
- it assigns large values to x with $\angle(w, x)$ close to zero,

Simplest neural network:

- $x = (x_0, x_1, \ldots, x_{d-1})^\top \in \mathbb{R}^d$ - input vector.
- $f(x)$ output of the only weight layer
- with weight vector $w = (w_0, w_1, \ldots, w_{d-1}, b) \in \mathbb{R}^d$
- $y = sign(z)$ - activation function on top of $f(x)$

Outline

1. Important Intermezzo: the inner product
2. Classification by logistic regression
3. Binary cross-entropy and Maximum Likelihood
4. Softmax and Cross-entropy loss for multiple classes

Towards logistic regression

Next steps: There is lots to derive from this simple case $f(x) = w \cdot x + b$

- derive an activation function with nice properties ($\rightarrow$ logistic sigmoid)
- derive a loss for this case ($\rightarrow$ Binary cross entropy loss)
- extend this to more than 2 classes as outputs ($\rightarrow$ softmax)
- derive a loss function for more than 2 classes as outputs ($\rightarrow$ Cross entropy loss)

Derivation of logistic regression

Goal of classification: For every input sample $x \in \mathcal{X}$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

First attempt: Apply a linear mapping and classify according to the sign of the output:

$$f(x) = w \cdot x + b, \; s(x) = sign(f(x)) \in \{-1, +1\}$$

While $f(x)$ has unbounded values, $s(x)$ is either -1 or 1, but:

- if $f(x) \approx 0$, we should be uncertain about the prediction.
- if $f(x) \gg 0$, we should be confident about the prediction 1.
- if $f(x) \ll 0$, we should be confident about the prediction -1.

Goal: encode this uncertainty

Derivation of logistic regression

Goal of classification: For every input sample $x \in \mathcal{X}$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

First attempt: Apply a linear mapping and classify according to the sign of the output:

$$f(x) = w \cdot x + b, \ s(x) = sign(f(x)) \in \{-1, +1\}$$

While $f(x)$ has unbounded values, $s(x)$ is either -1 or 1, but:

- if $f(x) \approx 0$, we should be uncertain about the prediction.
- if $f(x) \gg 0$, we should be confident about the prediction 1.
- if $f(x) \ll 0$, we should be confident about the prediction -1.

Goal: encode this uncertainty

Derivation of logistic regression

Goal of classification: For every input sample $x \in \mathcal{X}$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

First attempt: Apply a linear mapping and classify according to the sign of the output:

$$f(x) = w \cdot x + b, \;\; s(x) = sign(f(x)) \in \{-1, +1\}$$

While $f(x)$ has unbounded values, $s(x)$ is either -1 or 1, but:

- if $f(x) \approx 0$, we should be uncertain about the prediction.
- if $f(x) \gg 0$, we should be confident about the prediction 1.
- if $f(x) \ll 0$, we should be confident about the prediction -1.

Goal: encode this uncertainty

Derivation of logistic regression

Goal of classification: For every input sample $x \in \mathcal{X}$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

First attempt: Apply a linear mapping and classify according to the sign of the output:

$$f(x) = w \cdot x + b, \;\; s(x) = sign(f(x)) \in \{-1, +1\}$$

While $f(x)$ has unbounded values, $s(x)$ is either -1 or 1, but:

- if $f(x) \approx 0$, we should be uncertain about the prediction.
- if $f(x) \gg 0$, we should be confident about the prediction 1.
- if $f(x) \ll 0$, we should be confident about the prediction -1.

Goal: encode this uncertainty

Derivation of logistic regression

Goal of classification: For every input sample $x \in \mathcal{X}$, correctly predict which class y it belongs.

- For 2-class classification, $y \in \{-1, +1\}$ or $y \in \{0, 1\}$.

First attempt: Apply a linear mapping and classify according to the sign of the output:

$$f(x) = w \cdot x + b, \; s(x) = sign(f(x)) \in \{-1, +1\}$$

While $f(x)$ has unbounded values, $s(x)$ is either -1 or 1, but:

- if $f(x) \approx 0$, we should be uncertain about the prediction.
- if $f(x) \gg 0$, we should be confident about the prediction 1.
- if $f(x) \ll 0$, we should be confident about the prediction -1.

Goal: encode this uncertainty

Derivation of logistic regression

How can we encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$? We would like to map using a function $s(\cdot)$:

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

This would allow us to interpret $s(u)$ as a probability over inputs u.

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

How can we encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$? We would like to map using a function $s(\cdot)$:

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

This would allow us to interpret $s(u)$ as a probability over inputs u.

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

How can we encode the uncertainty into a mapping from $(-\infty, +\infty)$ onto $[0, 1]$? We would like to map using a function $s(\cdot)$:

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

This would allow us to interpret $s(u)$ as a probability over inputs u.

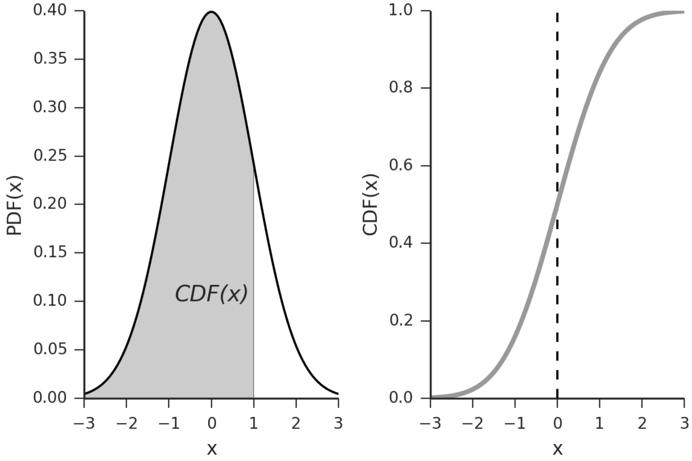
There are many possible choices for the function $s(u)$.

Derivation of logistic regression

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

We could e.g. use the cumulative distribution function (CDF) of any probability distribution function (PDF) with a median of 0.

- The normal (Gaussian) distribution is used in probit (from **prob**ability and un**it**) regression.



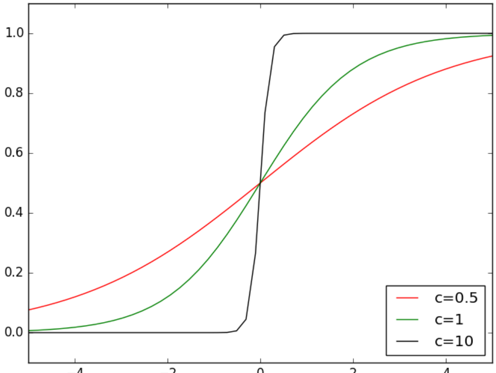
We will use a simpler function called the logistic sigmoid function:

Definition: Logistic sigmoid function

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

How does the logistic sigmoid function look like?
Let's plot it for different scaling factors $c > 0$:

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



Definition: Logistic sigmoid function

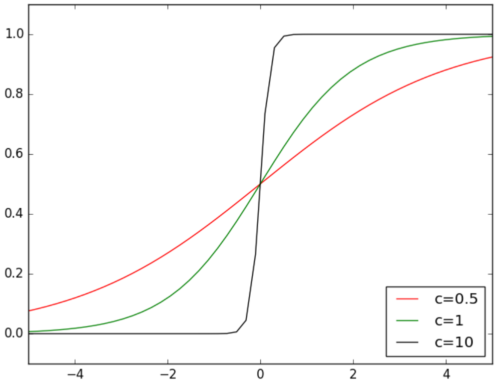
$$s(u) = \frac{\exp(u)}{1+\exp(u)} = \frac{1}{\exp(-u)+1}\frac{\exp(u)}{\exp(u)} = \frac{1}{\exp(-u)+1}$$

Note that:

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

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

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



Definition: Logistic regression model

Assume we have an affine mapping $f_{w,b}(x) = w \cdot x + b$. Plugging it into the logistic sigmoid function $s(u)$ provides a logistic regression model:

$$s(f_{w,b}(x)) = \frac{\exp(w \cdot x + b)}{1 + \exp(w \cdot x + b)} = \frac{1}{\exp(-w \cdot x - b) + 1}$$

For 2-class classification problems, the output $s(f_{w,b}(x)) \in [0, 1]$ is the predicted probability that sample x has class label $y = 1$, that is, $P(Y = 1 | X = x)$.

Interpretation of Logistic regression model outputs

$$s(f_{w,b}(x)) = \frac{\exp(w \cdot x + b)}{1 + \exp(w \cdot x + b)} = \frac{1}{\exp(-w \cdot x - b) + 1}$$

is a model for $P(Y = 1|X = x)$ in the classification prediction.

Indeed, for every input x we have that $s(f_{w,b}(x)) \in (0, 1)$. So it can be considered as a probability value.

Since we have only two classes, it must hold:

$$P(Y = 1|X = x) + P(Y = 0|X = x) = 1$$
$$P(Y = 0|X = x) = 1 - P(Y = 1|X = x) = 1 - s(f_{w,b}(x))$$

and we can use $s(f_{w,b}(x))$ to express the probability for the other class $P(Y = 0|X = x)$.

next step:

- derive a loss function which can be used for the logistic regression prediction model!

Binary cross entropy loss for a single output

Let us consider classification with 2 classes with labels $\{0, 1\}$, and let $s(x)$ be a model for $P(Y = 1|X = x)$. Then the binary cross entropy loss for a pair of prediction and label $(s(x), y)$ is given as

$$e(x, y) = -y \ln(s(x)) - (1 - y) \ln(1 - s(x))$$

This looks complicated but it has a simple interpretation:

- $y = 1 \Rightarrow e = -\ln(s(x))$ ...! the neg-log-probability $P(Y = 1|X = x)$ of the ground truth class $y = 1$
- $y = 0 \Rightarrow e = -\ln(1 - s(x))$ ...! the neg-log-probability $P(Y = 0|X = x)$ of the ground truth class $y = 0$

Binary cross-entropy loss function

Binary cross entropy loss for a single output

Let us consider classification with 2 classes with labels $\{0, 1\}$, and let $s(x)$ be a model for $P(Y = 1|X = x)$. Then the binary cross entropy loss for a pair of prediction and label $(s(x), y)$ is given as

$$e(x, y) = -\ln(P(Y = 1|X = x))1[y == 1] - \ln(P(Y = 0|X = x))1[y == 0]$$

Derivation of the cross-entropy loss function

Binary cross entropy loss for a single output

Let us consider classification with 2 classes with labels $\{0, 1\}$, and let $s(x)$ be a model for $P(Y = 1|X = x)$. Then the binary cross entropy loss for a pair of prediction and label $(s(x), y)$ is given as

$$e = -y \ln(s(x)) - (1 - y) \ln(1 - s(x))$$

This is the neg-logarithm of the probability $P(Y = y|X = x)$ for the ground truth label class y where $s(x) = P(Y = 1|X = x)$ – if one uses 0-1-labels.
(for a given pair (x, y) of feature x and its label y)

Next: Why is the neg-logarithm of the probability of the ground-truth class a good loss function?

Why is the neg-logarithm of the probability of the ground-truth class a good loss function?

Next: Why is the neg-logarithm of the probability of the ground-truth class a good loss function?

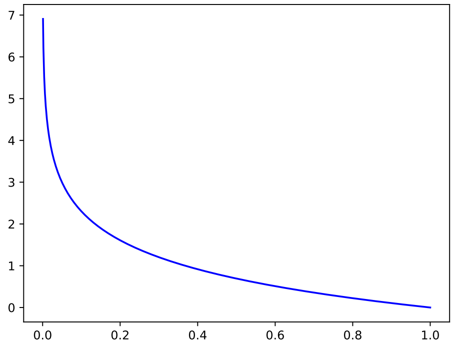
Desirable properties for a loss function:

- it should have low loss if the prediction is close to the ground truth
- it should have high loss if the prediction is far from the ground truth
- if one wants to use gradient-based optimization, it should be almost everywhere differentiable

Why is the neg-logarithm of the probability of the ground-truth class a good loss function?

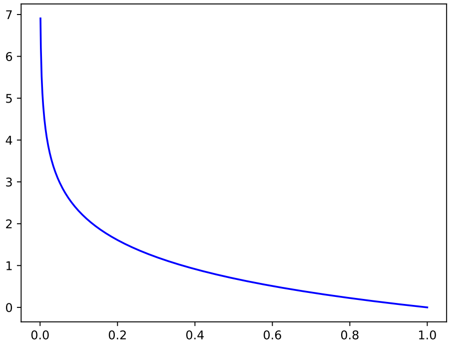
Suppose $y = 1$. Then the optimal prediction is $P(Y = 1|X = x) = 1$.

- If we had he optimal prediction is $P(Y = 1|X = x) = 1$, then the loss is $-\ln(1) = 0$. Zero loss
- If we had the least desirable prediction $P(Y = 1|X = x) = 0$, then the loss is $\lim_{x \to 0, x > 0} -\ln(x) = \infty$.
  The neg logarithm looks like this:



Why is the neg-logarithm of the probability of the ground-truth class a good loss function?

The neg logarithm looks like this:



- So if $y = 1$, then too low predicted probabilities $s(x)$ close to zero get a large loss.
- Similarly if $y = 0$, then too high probabilities $s(x) \approx 1$ close to one get a large loss (because then $1 - s(x) \approx 0$ is close to zero again, thus its neg log will be high)

Takeaway:

- the neg log probability prediction for the ground truth class is a reasonable loss function for classification problems.
- we have a classification model, which returns probabilities
- we have a loss to train it (not said how to use it)

Next:

- show that this loss has a derivation from a principle.

Outline

1 Important Intermezzo: the inner product

2 Classification by logistic regression

3 Binary cross-entropy and Maximum Likelihood

4 Softmax and Cross-entropy loss for multiple classes

Derivation of cross-entropy loss

Next:

- show that this loss has a derivation from a principle.

Steps:

(1) define a probability model $P(Y|X=x)$ for the prediction on a single input sample x

(2) define a probability model $P(Y_0, Y_1, Y_2, \ldots | X_0 = x_0, X_1 = x_1, X_2 = x_2, \ldots)$ for predictions on a set of samples $x_0, x_1, x_2, \ldots$

(3) employ the principle of maximum likelihood to select parameters for the probability model

(4) use a log-space transformation

Derivation of cross-entropy loss, step (1)

(a probability model $P(Y|X=x)$)

- The logistic output $s(f_{w,b}(x)) \in [0,1]$ can be interpreted as the probability $P(\{Y=1|X=x\})$ of observing label $Y=1$, predicted by the model.
- then:

$$
\begin{aligned}
P(Y=0|X=x) = 1 - P(Y=1|X=x) = 1 - s(f_{w,b}(x)) \\
y \in \{0,1\}: \; P(Y=y|X=x) = sy + (1-s)(1-y) \\
y \in \{0,1\}: \; P(Y=y|X=x) = s^y(1-s)^{1-y}
\end{aligned}
$$

  both are valid expressions for $P(Y=y|X=x)$
- we will use $y \in \{0,1\}: \; P(Y=y|X=x) = s^y(1-s)^{1-y}$ (compat. w. log-transform)

Derivation of cross-entropy loss, step (2)

- define a model for $P(Y_0, Y_1, Y_2, \ldots | X_0 = x_0, X_1 = x_1, X_2 = x_2, \ldots)$ by multiplying all probabilities $P(\{Y_k | X_k = x_k\})$ together:

$$\begin{aligned} P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \\ P(\{Y_0 | X_0 = x_0\}) P(\{Y_1 | X_1 = x_1\}) \cdot \ldots \cdot P(\{Y_{n-1} | X_{n-1} = x_{n-1}\}) \\ = \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\}) \end{aligned}$$

- why does this make sense?
  - note: The set of all values for $(Y_0, Y_1, \ldots, Y_{n-1})$ are all sequences $(\underbrace{11010\ldots1}_{len=n})$ of length n.
  - example values for $n = 3$ are $(0,0,0), (1,0,1), (0,1,1)$
  - Expressed by math: $\{0,1\}^n$

Derivation of cross-entropy loss, step (2)

$$P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = $$
$$= \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\})$$

- this is a probability over all possible values of $Y_0, Y_1, \ldots, Y_{n-1}$:
  - $P(\{Y_k | X_k = x_k\}) \in [0,1] \Rightarrow 0 \leq \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\}) \leq 1$
  - it sums up to $= 1$ over all possible values:

$$\sum_{Y_0 \in \{0,1\}, Y_1 \in \{0,1\}, \ldots, Y_{n-1} \in \{0,1\}} \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\})$$
$$= \sum_{Y_0 \in \{0,1\}} \sum_{Y_1 \in \{0,1\}} \cdots \sum_{Y_{n-1} \in \{0,1\}} \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\}) = 1$$

Derivation of cross-entropy loss, step (2)

$$\sum_{Y_0\in\{0,1\},Y_1\in\{0,1\},\dots,Y_{n-1}\in\{0,1\}} \prod_{k=0}^{n-1} P(\{Y_k|X_k = x_k\})$$

$$= \sum_{Y_0\in\{0,1\}} \sum_{Y_1\in\{0,1\}} \cdots \sum_{Y_{n-1}\in\{0,1\}} \prod_{k=0}^{n-1} P(\{Y_k|X_k = x_k\})$$

$$= \sum_{Y_0\in\{0,1\}} \sum_{Y_1\in\{0,1\}} \cdots \sum_{Y_{n-1}\in\{0,1\}} P(\{Y_0|X_0 = x_0\}) \prod_{k=1}^{n-1} P(\{Y_k|X_k = x_k\})$$

$P(Y_0|\dots)$ is a constant for summations over $Y_1, Y_2, \dots, Y_{n-1}$

DDerivation of cross-entropy loss, step (2)

The trick of pulling out a constant:

$$\sum_{Y_1}\sum_{Y_2} c(Y_0)t(Y_1, Y_2) = c(Y_0)\sum_{Y_1}\sum_{Y_2} t(Y_1, Y_2)$$

$$\Rightarrow \sum_{Y_0}\sum_{Y_1}\sum_{Y_2} c(Y_0)t(Y_1, Y_2) = \sum_{Y_0} c(Y_0)\sum_{Y_1}\sum_{Y_2} t(Y_1, Y_2)$$

Derivation of cross-entropy loss, step (2)

$$\sum_{Y_0\in\{0,1\},Y_1\in\{0,1\},\ldots,Y_{n-1}\in\{0,1\}} \prod_{k=0}^{n-1} P(\{Y_k|X_k = x_k\})$$

$$= \sum_{Y_0\in\{0,1\}} \sum_{Y_1\in\{0,1\}} \cdots \sum_{Y_{n-1}\in\{0,1\}} P(\{Y_0|X_0 = x_0\}) \prod_{k=1}^{n-1} P(\{Y_k|X_k = x_k\})$$

$P(Y_0|\ldots)$ is a constant for summations over $Y_1, Y_2, \ldots, Y_{n-1}$

$$= \sum_{Y_0\in\{0,1\}} P(\{Y_0|X_0 = x_0\}) \sum_{Y_1\in\{0,1\}} \cdots \sum_{Y_{n-1}\in\{0,1\}} \prod_{k=1}^{n-1} P(\{Y_k|X_k = x_k\})$$

$$=1 \sum_{Y_1\in\{0,1\}} \cdots \sum_{Y_{n-1}\in\{0,1\}} \prod_{k=1}^{n-1} P(\{Y_k|X_k = x_k\})$$

n-1 times iterate: $=1 \cdot \ldots \cdot 1 = 1$

- Result: We have shown that

$$P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \\ = \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\})$$

defines a probability distribution over the set $\{0, 1\}^n$ of all values for the sequence $(Y_0, Y_1, \ldots, Y_{n-1})$

Derivation of cross-entropy loss, step (3)

- using our logistic regression output $s(f_{w,b}(x))$, we have defined a probability model for observing values $y_0, y_1, \ldots, y_{n-1}$

$$P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}, w, b)$$

- we would like to find suitable values for the model parameters w and b.

- the simplest thought is to choose (w, b) is such that the probability to observe the data which we have $y_0, \ldots, y_{n-1} \in \{0, 1\}$ is maximized.
- This is called the principle of maximal likelihood
(wähle die Parameter derart, dass die Plausibilität der Beobachtung der gegebenen Daten maximiert wird)

$$(w^*, b^*) = \mathrm{argmax}_{\{(w,b)\}} P(Y_0 = y_0, Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1} | w, b)$$

Derivation of cross-entropy loss, step (3)

- using our logistic regression output $s(f_{w,b}(x))$, we have defined a probability model for observing values $y_0, y_1, \ldots, y_{n-1}$

$$P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}, w, b)$$

- we would like to find suitable values for the model parameters w and b.

- the simplest thought is to choose (w, b) is such that the probability to observe the data which we have $y_0, \ldots, y_{n-1} \in \{0, 1\}$ is maximized.
- This is called the principle of maximal likelihood
(wähle die Parameter derart, dass die Plausibilität der Beobachtung der gegebenen Daten maximiert wird)

$$(w^*, b^*) = \mathrm{argmax}_{\{(w,b)\}} P(Y_0 = y_0, Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1} | w, b)$$

Derivation of cross-entropy loss, step (3)

Maximum likelihood principle

Given some data observations $z_0, z_1, \ldots, z_{n-1}$, and a probability model $P(Z_0 = z_0, Z_1 = z_1, \ldots, Z_{n-1} = z_{n-1}|\theta)$ which depends on some parameter θ, the Maximum likelihood principle says, that one can choose θ such that it maximizes the probability of observing the given data:

$$\theta^* = \mathrm{argmax}_{\text{all possible } \theta} P(Z_0 = z_0, Z_1 = z_1, \ldots, Z_{n-1} = z_{n-1}|\theta)$$

Choose the model parameters such that observing the given data samples becomes most likely.

One missing point:

- Why does it make sense to define a probability as a product of observations for single samples ?

$$P(Y_0, Y_1, \ldots, Y_{n-1} | X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \\ = \prod_{k=0}^{n-1} P(\{Y_k | X_k = x_k\})$$

This makes sense if one can assume statistical independence (to be exact: conditional independence of Y given that one knows the X)

Give an example of data collection, where statistical independence is not fully satisfied?

Maximum likelihood principle with independence assumption

Given n vectors/data points $z_0, \dots, z_{n-1}$, a probability model $p(z_i;\theta)$ and an assumption of independence of z_i given θ, choose the parameters θ such that the probability of observing the data is maximized:

$$\theta^* = \mathrm{argmax}_\theta\, p(z_0;\theta)p(z_1;\theta)\dots p(z_{n-1};\theta)$$

$$\text{or } \theta^* = \mathrm{argmin}_\theta \sum_{i=0}^{n-1} -\ln(p(z_i;\theta))$$

Use as model for generating / simulating new data: $p(x;\theta^*)$

Derivation of cross-entropy loss, step (4)

Maximum likelihood with independence assumption:

$$\theta^* = \mathrm{argmax}_\theta \, p(z_0;\theta)p(z_1;\theta)\ldots p(z_{n-1};\theta)$$

for many distributions with exponentials it is easier to instead minimize the neg-log likelihood.

$$\begin{aligned}\theta^* &= \mathrm{argmin}_\theta - \ln\left[p(z_0;\theta)p(z_1;\theta)\ldots p(z_{n-1};\theta)\right] \\ &= \mathrm{argmin}_\theta\left[-\ln p(z_0;\theta) - \ln p(z_1;\theta) - \ldots - \ln p(z_{n-1};\theta)\right]\end{aligned}$$

This gives the same solution.

Derivation of cross-entropy loss, step (4)

Next step: apply this:

$$\operatorname{argmin}_\theta \Big[-\ln p(z_0;\theta) - \ln p(z_1;\theta) - \ldots - \ln p(z_{n-1};\theta) \Big]$$

- plug in our model: $z_i = (y_i, z_i)$, $P(z_i;\theta) = P(Y_i = y_i | X_i = x_i, w, b)$
- use $P(Y_i = y_i | X_i = x_i, w, b) = s(x_i)^{y_i}(1 - s(x_i))^{1-y_i}$ and $\ln(ab) = \ln(a) + \ln(b), \ln(a^c) = c\ln(a)$

$$\begin{aligned} -\ln P(Y_i = y_i | X_i = x_i, w, b) &= -\ln(s(x_i)^{y_i}(1 - s(x_i))^{1-y_i}) \\ &= y_i(-\ln(s(x_i))) + (1 - y_i)(-\ln(1 - s(x_i))) \end{aligned}$$

This is the binary cross entropy loss from above (compare!)

$$e = -y\ln(s(x)) - (1 - y)\ln(1 - s(x))$$

Derivation of the cross-entropy loss function

Take away:

Binary cross entropy loss for a single output

Let us consider classification with 2 classes with labels $\{0,1\}$, and let $s(x)$ be a model for $P(Y=1|X=x)$. Then the binary cross entropy loss for a pair of prediction and label $(s(x), y)$ is given as

$$e = -y\ln(s(x)) - (1-y)\ln(1-s(x)) = \sum_{i\in\{0,1\}} -\ln P(Y=i|X=x)\mathbb{1}[i=y]$$

- This is the neg-logarithm of the probability $P(Y=y|X=x)$ for the ground truth label class y
- It can be derived from applying the maximum likelihood principle aiming at choosing parameters to maximize the probability to observe the ground truth labels y_i

Outline

1. Important Intermezzo: the inner product
2. Classification by logistic regression
3. Binary cross-entropy and Maximum Likelihood
4. Softmax and Cross-entropy loss for multiple classes

- Suppose we have C classes: $y \in \{0, \ldots, C-1\}$ and
- we have a function $P(Y = k|X_i = x_i, \theta)$ which returns probabilities for each label k
- the problem is Multi-class classification: the C classes are mutually exclusive.

Multi-class classification

- class labels are mutually exclusive in the ground truth
- if the predictions modeled as probability $P(Y = k|X_i = x_i, \theta)$, this requires that

$$\sum_{k=0}^{C-1} P(Y = k|X_i = x_i, \theta) = 1$$

- note: one can model the absence of anything by a background class label

Derivation of the cross-entropy loss function

- Suppose we have C classes: $y \in \{0, \dots, C-1\}$ and
- we have a function $P(Y = k | X_i = x_i, \theta)$ which returns probabilities for each label k
- Lets represent the label $y \in \{0, \dots, C-1\}$ via a one-hot vector $\widetilde{y} \in \mathbb{R}^c$:

$$\widetilde{y}[k] = \begin{cases} 1 & \text{if } y = k \\ 0 & \text{else} \end{cases}$$

$$\widetilde{y} = (0, \dots, 0, \underbrace{1}_{\text{at value of } y}, 0, \dots)^\top$$

Then $P(Y_i = y_i | X_i = x_i, \theta) = \prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]}$

Derivation of the cross-entropy loss function

- Suppose we have C classes: $y \in \{0, \dots, C-1\}$ and
- we have a function $P(Y = k | X_i = x_i, \theta)$ which returns probabilities for each label k
- Lets represent the label $y \in \{0, \dots, C-1\}$ via a one-hot vector $\widetilde{y} \in \mathbb{R}^c$

  Then $P(Y_i = y_i | X_i = x_i, \theta) = \prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]}$

  ... simply because for all k where $\widetilde{y_i}[k] = 0$ we have
  $P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} = P(Y = k | X_i = x_i, \theta)^0 = 1$

  ... for the sole one k where $\widetilde{y_i}[k] = 1$ this returns:

$$P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} = P(Y = k | X_i = x_i, \theta)^1 = P(Y_i = y_i | X_i = x_i, \theta)$$

$$\Rightarrow \prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} = P(Y_i = y_i | X_i = x_i, \theta)$$

We have have C classes: $y \in \{0, \dots, C-1\}$ and

$$\prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} = P(Y_i = y_i | X_i = x_i, \theta)$$

now apply the same principle: we try to find θ^* which is the minimizer of

$$\begin{aligned}
\theta^* &= \mathrm{argmin}_\theta \sum_i - \ln p(z_i; \theta) \\
&= \mathrm{argmin}_\theta \sum_i - \ln P(Y_i = y_i | X_i = x_i, \theta) \\
&= \mathrm{argmin}_\theta \sum_i - \ln \prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} \\
&= \mathrm{argmin}_\theta \sum_i \sum_{k=0}^{C-1} - \ln P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} \\
&= \mathrm{argmin}_\theta \sum_i \sum_{k=0}^{C-1} - \widetilde{y_i}[k] \ln P(Y = k | X_i = x_i, \theta)
\end{aligned}$$

Derivation of the cross-entropy loss function

We have have C classes: $y \in \{0, \ldots, C-1\}$ and

$$\prod_{k=0}^{C-1} P(Y = k | X_i = x_i, \theta)^{\widetilde{y_i}[k]} = P(Y_i = y_i | X_i = x_i, \theta)$$

we try to find θ^* which is the minimizer of

$$\theta^* = \operatorname{argmin}_\theta \sum_{\text{samples } i} \sum_{k=0}^{C-1} -\widetilde{y_i}[k] \ln P(Y = k | X_i = x_i, \theta)$$

Multi-class cross entropy loss

Let $\widetilde{y_i}$ be the C-dimensional one-hot vector encoding labels in $\{0, \ldots, C-1\}$, and let $P(Y = k | X_i = x_i, \theta)$ the probability for class k given input sample x_i, then the cross-entropy loss for one sample (x_i, y_i) is defined as:

$$\sum_{k=0}^{C-1} -\widetilde{y_i}[k] \ln P(Y = k | X_i = x_i, \theta)$$

Multi-class cross entropy loss

Let $\widetilde{y}_i$ be the C-dimensional one-hot vector encoding labels in $\{0, \ldots, C-1\}$, and let $P(Y = k|X_i = x_i, \theta)$ the probability for class k given input sample x_i, then the cross-entropy loss for one sample (x_i, y_i) is defined as:

$$\sum_{k=0}^{C-1} -\widetilde{y}_i[k] \ln P(Y = k|X_i = x_i, \theta)$$

- This is again the neg-log probability for the class of the ground-truth label
  - if the probability for the ground-truth class is $= 1$, we have $-\ln(1) = 0$ for the loss

Takeaway for today:

Takeaway for today:

- logistic regression: model which computes inner product $+b$ with one output, then combines it with the logistic sigmoid
- a possible loss: binary cross entropy - the neg-log of the output probability for the ground truth class y for a given sample (x, y)
- neg-log (close to 0) $=$ close to ∞, $-\ln(1.0) = 0$
- binary cross entropy - derived from maximum likelihood principle to observe the ground truth labels
- logistic regression: 1-layer NN with sigmoid activation function

The key you need to understand sigmoid activation and cross entropy loss, is how the exponential and the (neg) logarithm look like.