Faculty of Sciences of Tetouan

Introduction to Deep Learning

3- Training (Part 1) : Loss functions

Prof. Monir EL ANNAS

Loss function

- Training dataset of I pairs of input/output examples:

$$\{X_i, Y_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

$$\mathcal{L}\left[\phi, f\{X, \phi\}, \{X_i, Y_i\}_{i=1}^{I}\right]$$

- or for short:

$$\mathcal{L}[\phi]$$

  - Returns a scalar that is smaller when model maps inputs to outputs better

- During training, we seek parameter values ϕ that minimize the loss:

$$\hat{\phi} = \underset{\phi}{\arg\min}\, \mathcal{L}[\phi]$$

Loss function

Maximum likelihood:

- Model predicts a conditional probability distribution $\Pr(y|x)$ over outputs y given inputs x.
- Loss function aims to make the outputs have high probability

Loss function

Maximum likelihood criterion:

$$\hat{\phi} = \arg\max_{\phi} \left[\prod_{i=1}^{I} Pr(y_i|x_i) \right]$$

$$= \arg\max_{\phi} \left[\prod_{i=1}^{I} Pr(y_i|\theta_i) \right]$$

$$= \arg\max_{\phi} \left[\prod_{i=1}^{I} Pr(y_i|f[x_i, \phi]) \right]$$

- When we consider this probability as a function of the parameters, we call it a likelihood.

Loss function

Maximum likelihood criterion:

- Problem :
  - The estimator is defined as:

$$\hat{\phi} = \underset{\phi}{\operatorname{arg\,max}} \left[\prod_{i=1}^{I} Pr(y_i | f[x_i, \phi]) \right]$$

  - The terms in this product might all be small
  - The product might get so small that we can't easily represent it

Loss function

Maximum log likelihood:

$$\hat{\phi} = \underset{\phi}{\arg\max}\left[\prod_{i=1}^{I} Pr(y_i|f[x_i,\phi])\right]$$

$$= \underset{\phi}{\arg\max}\left[\log\prod_{i=1}^{I} Pr(y_i|f[x_i,\phi])\right]$$

$$= \underset{\phi}{\arg\max}\left[\sum_{i=1}^{I} \log Pr(y_i|f[x_i,\phi])\right]$$

Now it's a sum of terms, so doesn't matter so much if the terms are small

Loss function

Minimizing negative log likelihood:

By convention, we minimize things (i.e., a loss)

$$\begin{aligned}\hat{\phi} &= \underset{\phi}{\operatorname{arg\,max}}\left[\sum_{i=1}^{I} \log Pr(y_i|f[x_i,\phi])\right] \\ &= \underset{\phi}{\operatorname{arg\,min}}\left[-\sum_{i=1}^{I} \log Pr(y_i|f[x_i,\phi])\right] \\ &= \underset{\phi}{\operatorname{arg\,min}}\,\mathcal{L}[\phi]\end{aligned}$$

Loss function

Inference:

- But now we predict a probability distribution
- We need an actual prediction (point estimate)
- Find the peak of the probability distribution (i.e., mean for normal)

$$\hat{y} = \arg\max_{y} \left[Pr(y|f[x, \phi]) \right]$$

Loss function

Recipe for loss functions:

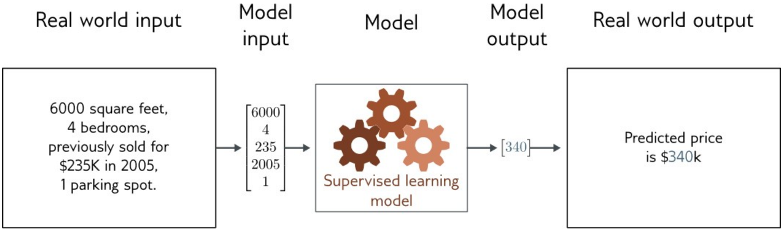
1. Choose a suitable probability distribution $Pr(y|\theta)$ that is defined over the domain of the predictions y and has distribution parameters θ.
2. Set the machine learning model $f(x, \phi)$ to predict one or more of these parameters so $\theta = f(x, \phi)$ and $Pr(y|\theta) = Pr(y|f(x, \phi))$.
3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{x_i, y_i\}$:

$$\hat{\phi} = \underset{\phi}{\arg\min}\,[L(\phi)] = \underset{\phi}{\arg\min}\left[-\sum_{i=1}^{I} \log Pr(y_i|f(x_i, \phi))\right].$$

4. To perform inference for a new test example x, return either the full distribution $Pr(y|f(x, \hat{\phi}))$ or the maximum of this distribution.

Least squares loss function

Univariate regression:



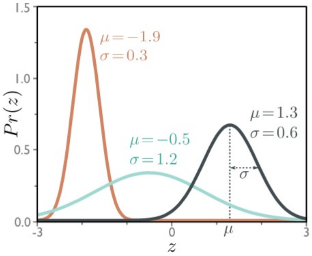
- Goal: predict a single scalar output $y \in \mathbb{R}$ from input x

Least squares loss function

Univariate regression:

1. Choose a suitable probability distribution $Pr(y|\theta)$ that is defined over the domain of the predictions y and has distribution parameters θ.
    - Predict scalar output: $y \in \mathbb{R}$
    - Sensible probability distribution:
        - Normal distribution

$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$



Least squares loss function

Univariate regression:

2. Set the machine learning model $f(x, \phi)$ to predict one or more of these parameters so $\theta = f(x, \phi)$ and $Pr(y|\theta) = Pr(y|f(x, \phi))$.

$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

$$Pr(y|f(x, \phi), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-f(x, \phi))^2}{2\sigma^2}\right)$$

Least squares loss function

Univariate regression:

3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{x_i, y_i\}$:

$$
\begin{aligned}
L(\phi) &= -\sum_{i=1}^{I} \log\left[Pr(y_i | f(x_i, \phi), \sigma^2)\right] \\
&= -\sum_{i=1}^{I} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(x_i, \phi))^2}{2\sigma^2}\right)\right]
\end{aligned}
$$

Least squares loss function

Univariate regression:

$$
\begin{aligned}
\hat{\phi} &= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(x_i, \phi))^2}{2\sigma^2} \right) \right) \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \log\left(\exp\left(-\frac{(y_i - f(x_i, \phi))^2}{2\sigma^2} \right) \right) \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \left(\frac{(y_i - f(x_i, \phi))^2}{2\sigma^2} \right) \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left(\frac{(y_i - f(x_i, \phi))^2}{2\sigma^2} \right) \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (y_i - f(x_i, \phi))^2 \right],
\end{aligned}
$$

Least squares Loss (L2 Loss)

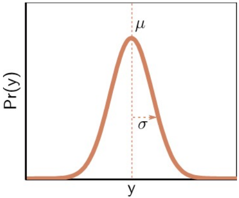
Least squares loss function

Univariate regression:

4. To perform inference for a new test example x, return either the full distribution $Pr(y|f(x,\hat{\phi}))$ or the maximum of this distribution.

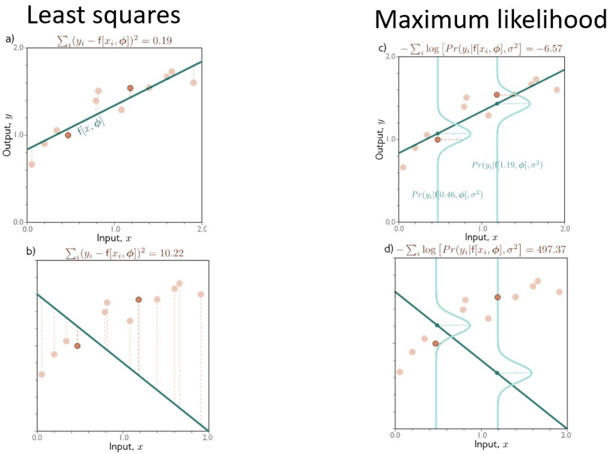
$$Pr(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

$$Pr(y|f(x,\hat{\phi}),\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-f(x,\hat{\phi}))^2}{2\sigma^2}\right)$$



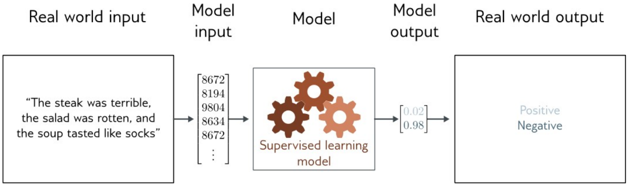
Least squares loss function

Univariate regression:



Binary cross-entropy loss function

Binary classification:



- Goal: predict which of two classes $y \in \{0, 1\}$ the input x belongs to

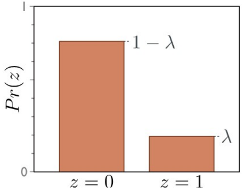
Binary cross-entropy loss function

Binary classification:

1. Choose a suitable probability distribution $Pr(y|\theta)$ that is defined over the domain of the predictions y and has distribution parameters θ.
   - Domain: $y \in \{0, 1\}$
   - Bernoulli distribution
   - One parameter $\lambda \in [0, 1]$

$$Pr(y|\lambda) = \begin{cases} 1 - \lambda & \text{if } y = 0 \\ \lambda & \text{if } y = 1 \end{cases}$$

$$Pr(y|\lambda) = (1 - \lambda)^{1-y} \cdot \lambda^{y}$$



Binary cross-entropy loss function

Binary classification:

2. Set the machine learning model $f(x, \phi)$ to predict one or more of these parameters so $\theta = f(x, \phi)$ and $Pr(y|\theta) = Pr(y|f(x, \phi))$.

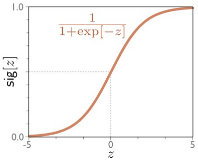
Problem:

- Output of neural network can be anything
- Parameter $\lambda \in [0, 1]$

Solution:

- Pass through logistic sigmoid function that maps "anything to [0,1]":

$$\text{sig}[z] = \frac{1}{1 + \exp[-z]}$$



- The likelihood:

$$Pr(y|\lambda) = (1 - \lambda)^{1-y} \cdot \lambda^{y}$$

$$Pr(y|x) = (1 - \text{sig}[f(x|\phi)])^{1-y} \cdot \text{sig}[f(x|\phi)]^{y}$$

Binary cross-entropy loss function

Binary classification:

3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{x_i, y_i\}$:

$$\hat{\phi} = \underset{\phi}{\mathrm{argmin}}\left[L(\phi)\right] = \underset{\phi}{\mathrm{argmin}}\left[-\sum_{i=1}^{I} \log\left(Pr(y_i|f(x_i,\phi))\right)\right]$$

$$Pr(y|x) = (1 - \mathrm{sig}[f(x|\phi)])^{1-y} \cdot \mathrm{sig}[f(x|\phi)]^{y}$$

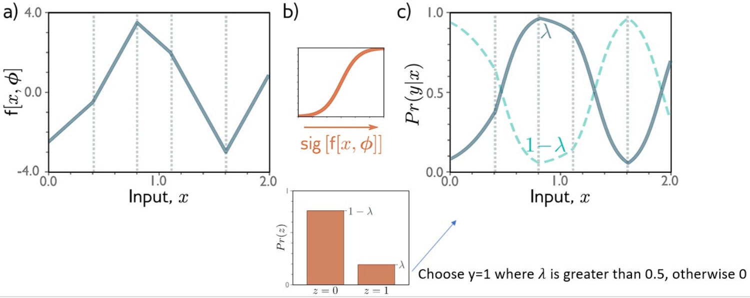
$$L(\phi) = \sum_{i=1}^{I} -\left((1-y_i)\log\left[1 - \mathrm{sig}[f(x_i|\phi)]\right] + y_i \log\left[\mathrm{sig}[f(x_i|\phi)]\right]\right)$$

Binary cross-entropy loss

Binary cross-entropy loss function

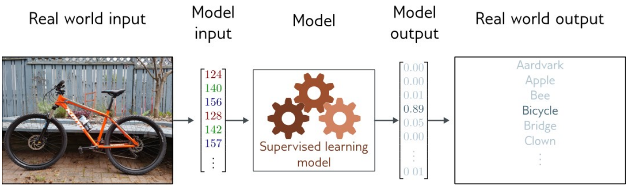
Binary classification:

4. To perform inference for a new test example x, return either the full distribution $Pr(y|f(x, \hat{\phi}))$ or the maximum of this distribution.



Multiclass cross-entropy loss function

Multiclass classification:



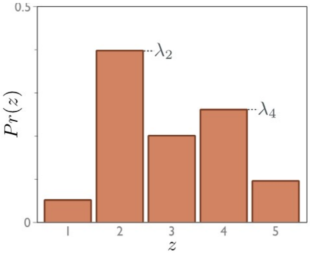
- Goal: predict which of K classes $y \in \{1, 2, \ldots, K\}$ the input x belongs to

Multiclass cross-entropy loss function

Multiclass classification:

1. Choose a suitable probability distribution $Pr(y|\theta)$ that is defined over the domain of the predictions y and has distribution parameters θ.
    - Domain: $y \in \{1, 2, \ldots, K\}$
    - Categorical distribution
    - K parameters $\lambda_k \in [0, 1]$
    - Sum of all parameters $= 1$

$$Pr(y = k) = \lambda_k$$



Multiclass cross-entropy loss function

Multiclass classification:

2. Set the machine learning model $f(x, \phi)$ to predict one or more of these parameters so $\theta = f(x, \phi)$ and $Pr(y|\theta) = Pr(y|f(x, \phi))$.

Problem:

- Output of neural network can be anything
- Parameters $\lambda_k \in [0, 1]$, sum to one

Solution:

- Pass through function that maps "anything" to [0,1], sum to one

$$Pr(y = k|x) = \text{softmax}_k[f(x, \phi)]$$

Where the softmax function is defined as:

$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K} \exp[z_{k'}]}$$

Multiclass cross-entropy loss function

Multiclass classification:

3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{x_i, y_i\}$:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\, L(\phi) = \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\left(Pr(y_i|f(x_i,\phi))\right)\right]$$

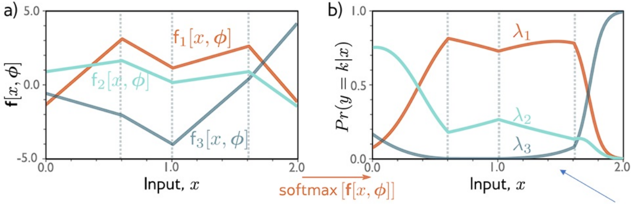
$$\begin{aligned} L(\phi) &= -\sum_{i=1}^{I} \log\left[\text{softmax}_{y_i}\left[f(x_i,\phi)\right]\right] \\ &= -\sum_{i=1}^{I} f_{y_i}[x_i,\phi] - \log\left[\sum_{k=1}^{K} \exp\left[f_k[x_i,\phi]\right]\right] \end{aligned}$$

Multiclass cross-entropy loss

Multiclass cross-entropy loss function

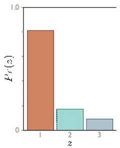
Multiclass classification:

4. To perform inference for a new test example x, return either the full distribution $Pr(y|f(x, \hat{\phi}))$ or the maximum of this distribution.



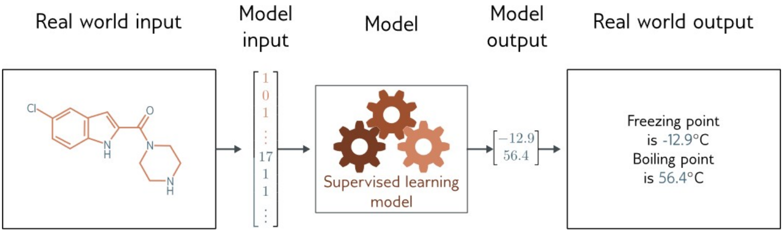
$$Pr(y = k|\mathbf{x}) = \text{softmax}_k[\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]]$$

Choose the class with the largest probability



Multivariate regression

Multivariate regression:



Multivariate regression

Multivariate regression:

- Treat each output y_d as independent:

$$Pr(y|f(x_i,\phi)) = \prod_d Pr(y_d|f_d(x_i,\phi))$$

- Negative log likelihood becomes sum of terms:

$$L(\phi) = -\sum_{i=1}^{I} \log\left[Pr(y|f(x_i,\phi))\right] = -\sum_{i=1}^{I}\sum_d \log\left[Pr(y_{id}|f_d(x_i,\phi))\right]$$

Multivariate regression

Multivariate regression:

- Goal: to predict a multivariate target $y \in \mathbb{R}^{D_o}$.
- Solution treat each dimension independently

$$Pr(y|\mu, \sigma^2) = \prod_{d=1}^{D_o} Pr(y_d|\mu_d, \sigma^2)$$

$$= \prod_{d=1}^{D_o} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_d - \mu_d)^2}{2\sigma^2}\right)$$

- Make network with D_o outputs to predict means

$$Pr(y|f(x, \phi), \sigma^2) = \prod_{d=1}^{D_o} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_d - f_d(x, \phi))^2}{2\sigma^2}\right)$$

Multivariate regression

Multivariate regression:

- What if the outputs vary in magnitude
  - E.g., predict weight in kilos and height in meters
  - One dimension has much bigger numbers than others
- Could learn a separate variance for each...
- ...or rescale before training, and then rescale output in opposite way

Next up

- We have models with parameters!
- We have loss functions!
- Now let's find the parameters that give the smallest loss
  - Training, learning, or fitting the model