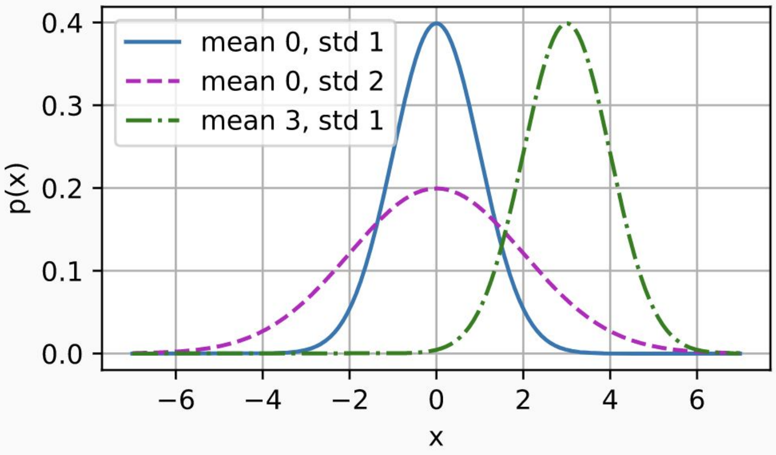
STATS / DATA SCI 315
Lecture 04

Regression wrap-up

The Normal Distribution and Squared Loss

Gaussian distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$



Linear regression and Gaussian distribution

- Linear model with Gaussian errors

$$y = \mathbf{w}^\top \mathbf{x} + b + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2).$$

- This gives us a **likelihood** of observing a particular *y* for a given **x**

$$P(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top \mathbf{x} - b)^2\right).$$

- Note that this likelihood is a function of **w** and *b* for fixed **x**, *y*

Maximum likelihood

- Likelihood of the entire dataset (assuming independence among samples)

$$P(\mathbf{y} \mid \mathbf{X}) = \prod_{i=1}^{n} p(y^{(i)} | \mathbf{x}^{(i)}).$$

- Maximum likelihood principle: *maximize* this (over parameters)
- We can take log (monotonic transformation) and a minus sign
- Then, equivalently, *minimize the negative log likelihood*
- The equivalence is for the minimizers, not for the value of the objective function!

Expression for the negative log likelihood

- Is it clear to everyone how to derive this?
- Since variance is a positive constant, this shows that MLE is equivalent to minimizing squared error (sum or average)

$$-\log P(\mathbf{y} \mid \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} - b\right)^2.$$

From Linear Regression to Deep Networks

Linear model as a beginning for deep learning

- Understand the complex/unfamiliar by relating it to sth simple/familiar
- We have only talked about **linear models**
- Deep learning builds *much more complex models*
- However, we can think about linear model in the language of NNs
- To begin, let us rewrite a linear model in “layer” notation

Number of output nodes = 1

Fully connected
Or **dense** layer

All inputs connect to all outputs

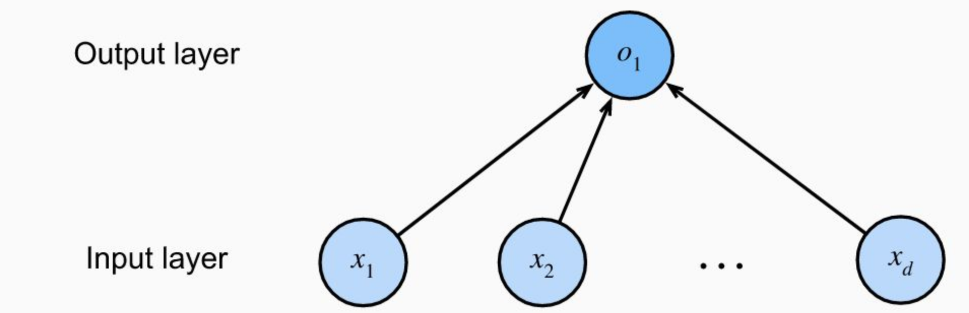


Fig. 3.1.2 Linear regression is a single-layer neural network.

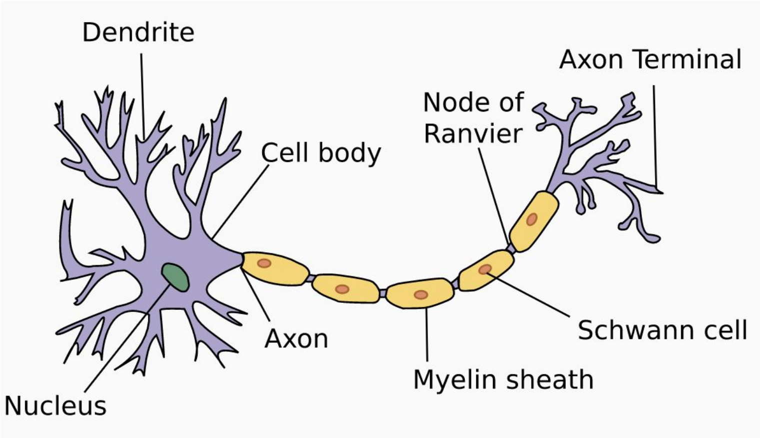
Number of input nodes = *d*

Number of layers is 1 (we don't count input layer)

Cartoonish picture of a biological neuron

axon = output wire
Axon terminal = output terminal

Dendrites = input terminals



Axons connect to other neurons via connections called *synapses* (not shown)

Nucleus = computational unit

Neuronal computation

- Info x_i arriving from other neurons (or sensors, e.g., retina) is received in the dendrites
- Info is weighted by synaptic weights w_i determining the effect of the inputs
  - Positive weights – activation
  - Negative weights – inhibition
- Simple linear weighting gives the result

$$y = \sum_i x_i w_i + b$$

- After applying a **nonlinear** σ, the result $\sigma(y)$ is sent to other neurons via the axon for further processing

Neuroscience provides high level inspiration

- Simple units can be cobbled together to produce far more interesting and complex behavior than a single neuron
- Need the right **connectivity** (DL engineers provide this)
- Need the right **learning algorithm** (DL uses backprop which is unlikely to be used by the brain)
- As far as these high level ideas are concerned, DL does derive its inspiration from neuroscience

On biological vs artificial neurons

- The cartoonish picture is imprecise
- There is evidence (see paper if interested) that a single biological neurons actually needs an artificial NN with several layers to model its complexity
- Deep learning today draws little direct inspiration in neuroscience
- Airplanes might have been inspired by birds
- But ornithology has not been the primary driver of aeronautics innovation
- Deep learning inspiration comes from math, stats, and CS