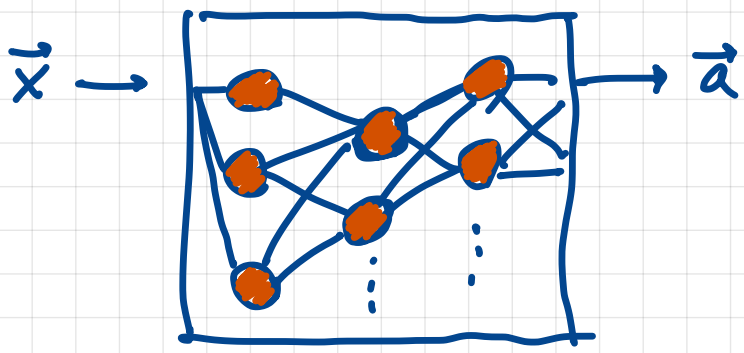
Short Takes 331

Machine learning: Stochastic gradient descent

More ML: Stochastic gradient descent

- A neural network (last video!) can be regarded as a complicated mathematical expression connecting the input vector $\vec{x}$ with the output layer activation $\vec{a}^{(L)}$.



w_{1i} w_{2i} ...

b_i b_i ...

- Our job is to optimize the weights and biases such that the output matches a desired result for a given input.
  $\vec{a}$ $\vec{y}$ $\vec{x}$

- Suppose we are given data samples

$$\left\{ (\vec{x},\vec{y})_1 , (\vec{x},\vec{y})_2 , \dots , (\vec{x},\vec{y})_{N_{samples}} \right\}$$

You'll want to divide these Nsamples into a training set and a test set

N_T N_V

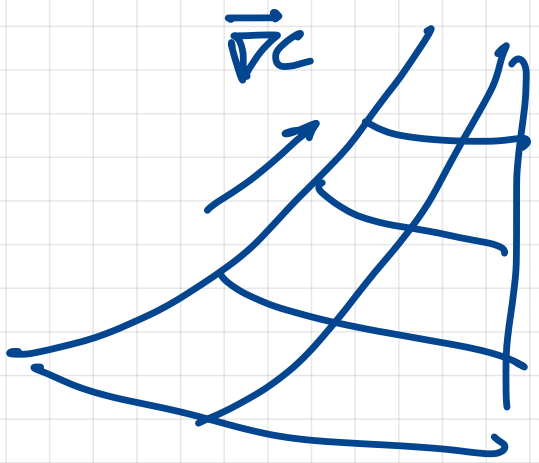
- To optimize the network one chooses a cost function $C(w,b)$.
  (all weights, all biases)

  Normally, to optimize in this multivariable space, one uses methods like gradient descent (GD)...

  Idea: To minimize $C(w,b)$, calculate the gradient vector $\vec{\nabla} C = \begin{pmatrix} \frac{\partial C}{\partial w} \\ \frac{\partial C}{\partial b} \end{pmatrix}$ and move in the opposite direction...

- start search at $\begin{pmatrix} w_{(0)} \\ b_{(0)} \end{pmatrix}$

- update to $\begin{pmatrix} w_{(1)} \\ b_{(1)} \end{pmatrix} = \begin{pmatrix} w_{(0)} \\ b_{(0)} \end{pmatrix} - \eta \cdot \vec{\nabla} C$



free parameter (learning rate)

- Assuming $C(w,b)$ does have a minimum, this process will (at least intuitively) take us there.

- In practice,

- $$C(w,b) = \frac{1}{N_T} \sum_{i=1}^{N_T} C_i(w,b) \quad ,$$

where $C_i(w,b)$ is calculated based on sample $(\vec{x}, \vec{y})_i$ only.

"quadratic cost"

Eg. $$C_i(w,b) = \left(\vec{y}_i - \vec{a}(\vec{x}_i, w, b) \right)^2$$

$\vec{y}_i$: desired output; $\vec{a}(\vec{x}_i, w, b)$: network output

Another choice: "cross entropy"

$$C_i(w,b) = -\left[\vec{y}_i \cdot \ln \vec{a}(\vec{x}_i, w, b) + (1 - \vec{y}_i) \cdot \ln \left(1 - \vec{a}(\vec{x}_i, w, b) \right) \right]$$

note dot prod.

(we assume here $0 \leq y \leq 1$)

- Calculating $\vec{\nabla} C$ for all the samples is prohibitively costly.

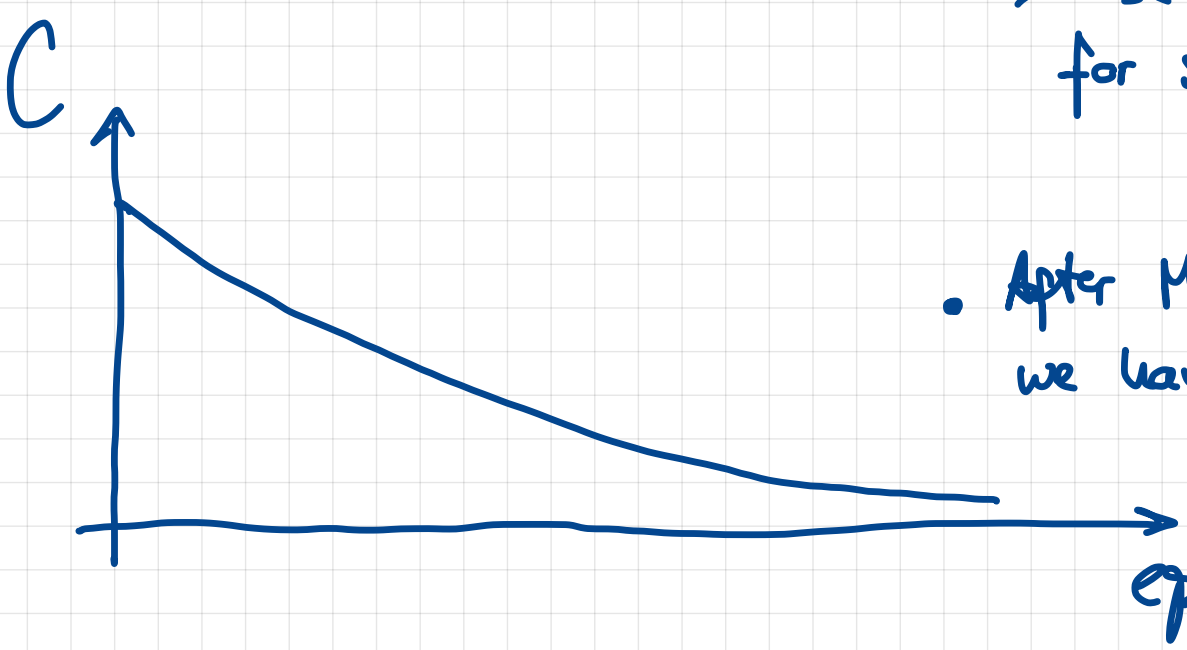
We take mini-batches to approximate ...

$N_T = m \cdot M$

- m: Size of mini-batch
- M: mini-batches of randomly chosen samples ← The "stochastic" part.

$$\vec{\nabla} C = \frac{1}{N_T} \cdot \sum_{i=1}^{N_T} \vec{\nabla} C_i \simeq \frac{1}{m} \sum_{i=1}^{m} \vec{\nabla} C_i$$

↳ use in gradient descent for some number of GD iterations.



- After M mini-batch rounds, we have completed an "epoch".

For NN's, $\vec{\nabla} C_i$ can be calculated quickly via "back propagation"... next time!