Introduction to Machine Learning

CS/SE 4X03

Ned Nedialkov

McMaster University

November 3, 2021

Outline

This is a summary of Sections 1-4 from

C. F. Higham, D. J. Higham, Deep Learning: An Introduction for Applied Mathematicians

Figures are cropped from this article

Example

- Points in $\mathbb{R}^2$ classified in two categories A and B
- This is labeled data

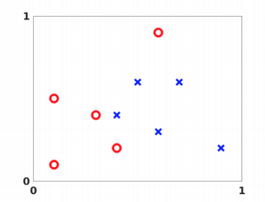
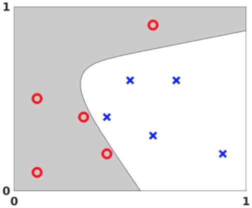


Figure 1: Labeled data points in $\mathbb{R}^2$. Circles denote points in category A. Crosses denote points in category B.

- Given a new point, how to use the labeled data to classify this point?
  Possible classification

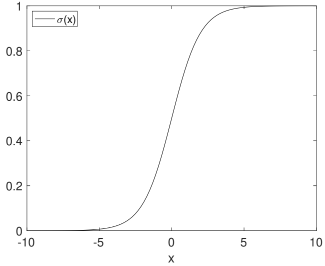


Activation function

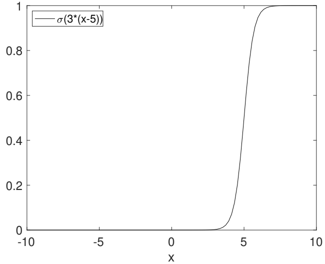
- A neuron fires or is inactive
- Activation can be modeled by the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- $\sigma(0) = 0.5$, $\sigma(x) \approx 1$ when x large, $\sigma(x) \approx 0$ when x small



- Steepness can be changed by scaling
- Location can be changed by shifting
- Useful property $\sigma'(x) = \sigma(x)(1 - \sigma(x))$



A simple network

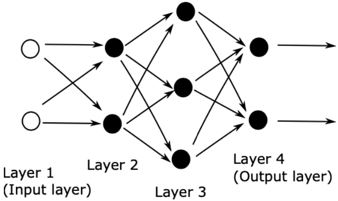


Figure 3: A network with four layers.

- Each neuron
  - outputs a real number
  - sends to every neuron in next layer
- Neuron in next layer
  - forms a linear combination of inputs + bias
  - applies activation function

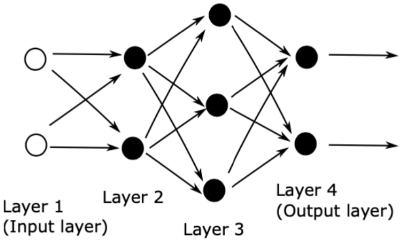
Consider layers 2 and 3

Layer 2: neurons 1 and 2 output real a_1 and a_2, respectively, and send to neurons $1, 2, 3$ in layer 3
Layer 3:

- neuron
  1 combines a_1 and a_2 and ads bias b_1:

  $$w_{11}a_1 + w_{12}a_2 + b_1$$

  outputs
  $\sigma\left(w_{11}a_1 + w_{12}a_2 + b_1\right)$
- neuron 2 outputs
  $\sigma\left(w_{21}a_1 + w_{22}a_2 + b_2\right)$
- neuron 3 outputs
  $\sigma\left(w_{31}a_1 + w_{32}a_2 + b_3\right)$



Denote

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}, \qquad a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$z = Wa + b$$

W is a matrix with weights, b is a bias vector

For a vector z, apply σ component wise

$$\left(\sigma(z)\right)_i = \sigma(z_i)$$

The output of layer 3 is

$$\sigma(z) = \sigma(Wa + b)$$

- Denote the input by x, the W and b at layer i by $W^{[i]}$ and $b^{[i]}$, and the output of layer i by $a^{[i]}$
- Output of layer 2 is

$$a^{[2]} = \sigma\left(W^{[2]}x + b^{[2]}\right) \in \mathbb{R}^2, \qquad W^{[2]} \in \mathbb{R}^{2\times 2},\ b^{[2]} \in \mathbb{R}^2$$

- Output of layer 3 is

$$a^{[3]} = \sigma\left(W^{[3]}a^{[2]} + b^{[3]}\right) \in \mathbb{R}^3, \qquad W^{[3]} \in \mathbb{R}^{3\times 2},\ b^{[3]} \in \mathbb{R}^3$$

- Output of layer 4 is

$$a^{[4]} = \sigma\left(W^{[4]}a^{[3]} + b^{[4]}\right) \in \mathbb{R}^2, \qquad W^{[4]} \in \mathbb{R}^{2\times 3},\ b^{[4]} \in \mathbb{R}^2$$

- Write the above as

$$F(x) = \sigma\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) + b^{[4]}\right)$$

- Layer i:
  - $W^{[i]}$ is of size (# outputs)$\times$(# inputs)
  - $b^{[i]}$ is of size (# outputs)

  Number of parameters is 23:

layer i	inputs	outputs	$W^{[i]}$	$b^{[i]}$
2	2	2	2×2	2
3	2	3	3×2	3
4	3	2	2×3	2
			16	7

- $F(x)$ is a function from $\mathbb{R}^2 \to \mathbb{R}^2$ with 23 parameters
- Training is about finding parameters

Training

Residual

- Denote the input points by $x^{\{i\}}$
- Let

$$y\left(x^{\{i\}}\right) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{if } x^{\{i\}} \in A \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } x^{\{i\}} \in B \end{cases}$$

- Suppose we have computed $W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}$ and evaluate $F\left(x^{\{i\}}\right)$
- Residual

$$\left\| y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right) \right\|_2$$

Training

Cost function

- Cost function

$$\text{Cost}\left(W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}\right) \\ = \frac{1}{10} \sum_{i=1}^{10} \frac{1}{2} \left\| y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right) \right\|_2^2$$

- Training: find the parameters that minimize the cost function
- Nonlinear least squares problem

Classifying

- Suppose we have computed values for the parameters
- Given $x \in \mathbb{R}^2$, compute $y = F(x)$
- If $y_1 > y_2$ classify x as A, y closer to $[1, 0]^T$
- If $y_1 < y_2$ classify x as B, y closer to $[0, 1]^T$
- Tie breaking when $=$

Steepest descent

- Consider the parameters in a vector $p \in \mathbb{R}^s$. Here $s = 23$
- Cost function is $\mathsf{Cost}(p)$
- Find Δp such that

$$\mathsf{Cost}(p + \Delta p) < \mathsf{Cost}(p)$$

- For small Δp,

$$\begin{aligned}
\mathsf{Cost}(p + \Delta p) &\approx \mathsf{Cost}(p) + \sum_{r=1}^{s} \frac{\partial \mathsf{Cost}(p)}{\partial p_r} \Delta p_r \\
&= \mathsf{Cost}(p) + \nabla \mathsf{Cost}(p)^T \Delta p \\
\nabla \mathsf{Cost}(p) &= \left[\frac{\partial \mathsf{Cost}(p)}{\partial p_1}, \frac{\partial \mathsf{Cost}(p)}{\partial p_2}, \cdots, \frac{\partial \mathsf{Cost}(p)}{\partial p_s} \right]^T
\end{aligned}$$

Example

- To illustrate the above, suppose

$$\mathsf{Cost}(p) = p_1^2 + p_2^2 + 2p_1 + 3$$

- Gradient is

$$\nabla \mathsf{Cost}(p) = [2p_1 + 2, 2p_2]^T$$

$$\begin{aligned}\mathsf{Cost}(p + \Delta p) &\approx \mathsf{Cost}(p) + \nabla \mathsf{Cost}(p)^T \Delta p \\ &= \mathsf{Cost}(p) + (2p_1 + 2)\Delta p_1 + 2p_2 \Delta p_2\end{aligned}$$

Steepest descent cont

- $\text{Cost}(p) \geq 0$
- From
$$\text{Cost}(p + \Delta p) \approx \text{Cost}(p) + \nabla \text{Cost}(p)^T \Delta p,$$
we want to make $\nabla \text{Cost}(p)^T \Delta p$ as negative as possible
- Given $\nabla \text{Cost}(p)$ how to choose Δp?
- For $u, v \in \mathbb{R}^s$,
$$u^T v = \|u\| \cdot \|v\| \cos \theta$$
is most negative when $v = -u$
- Chose Δp in the direction of $-\nabla \text{Cost}(p)$
That is move along the direction of steepest descent

$$\Delta p = p_{\mathsf{new}} - p = -\eta \nabla \mathsf{Cost}(p)$$
$$p_{\mathsf{new}} = p - \eta \nabla \mathsf{Cost}(p)$$

η is learning rate

Steepest descent:

chose initial p
repeat
$\quad p \leftarrow p - \eta \nabla \mathsf{Cost}(p)$
until stopping criterion is met or max # of iterations is reached

- In general N input points

$$\text{Cost}(p) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{1}{2} \left\| y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right) \right\|_2^2}_{C_i(p)}$$

$$= \frac{1}{N} \sum_{i=1}^{N} C_i(p)$$

$$\nabla \text{Cost}(p) = \frac{1}{N} \sum_{i=1}^{N} \nabla C_i(p)$$

- N can be large
- Number of parameters can be very large
- Evaluating $\nabla \text{Cost}(p)$ can be very expensive

Stochastic gradient descent

- Idea: replace $\frac{1}{N}\sum_{i=1}^{N}\nabla C_i(p)$ by random $\nabla C_i(p)$
- Iterate until a stopping criterion is met or max # of iterations is reached:
  - pick a random integer i from $\{1, 2, \ldots, N\}$
  - $p \leftarrow p - \eta \nabla C_i(p)$