Neural Networks and Deep Learning

Learning I

Nicholas Dronen

Department of Computer Science
dronen@colorado.edu

January 23, 2019

Questions?

Linear Associators

Hebbian Learning

Least Mean Squares (LMS)

Notation

Vector	$\mathbf{x}$
Element i of $\mathbf{x}$	x_i
Matrix	$\mathbf{X}$
Row i of $\mathbf{X}$	$\mathbf{X}_i$
Element i, j of $\mathbf{X}$	x_{ij}

Network input	$\mathbf{x}$
Network output	$\hat{\mathbf{y}}$, $\hat{y}$
Network target	$\mathbf{y}$, y
Weight matrix	$\mathbf{W}$
Bias vector	$\mathbf{b}$
Set of network parameters	θ

Auto-Associative Memory

Given an incomplete representation of data, recall the data itself.

$\Longrightarrow$

$\Longrightarrow$

Hetero-Associative Memory

Given a complete representation of data, recall related data.

$\Longrightarrow$

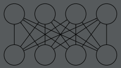
Ford GP
World War II
1941
Fun!

$\Longrightarrow$

Name
Hobbies
Home town

Linear Associators (Anderson, Kohonen)

- Two sets of units – input and output
- Fully-connected – in the neural networks literature, a "fully-connected" network is a sequence of complete, bipartite graphs.



- Linear activation function

$$\widehat{\mathbf{y}} = \mathbf{W}\mathbf{x}$$

$$\begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_B \end{bmatrix} = \begin{bmatrix} w_{11} \ldots w_{1A} \\ \vdots \\ w_{B1} \ldots w_{BA} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_A \end{bmatrix}$$

Linear Associators (Anderson, Kohonen)

Supervised task - learn map from input $\mathbf{x}$ to output $\mathbf{y}$, e.g. for example α

$$\begin{bmatrix} x_1^\alpha \\ \vdots \\ x_A^\alpha \end{bmatrix} \to \begin{bmatrix} y_1^\alpha \\ \vdots \\ y_B^\alpha \end{bmatrix}$$

How do we set weights $\mathbf{W}$ so $\widehat{\mathbf{y}}^\alpha = \mathbf{W}\mathbf{x}^\alpha \sim \mathbf{y}^\alpha$?

Hebbian Learning

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." (Donald O. Hebb, 1949)

Hebbian Weight Update Rule

Formalization of Hebb rule:

1. $\mathbf{W} \sim \mathcal{N}(\mu, \sigma^2)$, e.g.
2. For $t = 1 \ldots p$

$$\begin{aligned} \alpha &= t \\ \widehat{\mathbf{y}}^{\alpha} &= \mathbf{W}\mathbf{x}^{\alpha} \\ \Delta\mathbf{W}_t &= \widehat{\mathbf{y}}^{\alpha}\mathbf{x}^{\alpha\,T} \\ \mathbf{W}_{t+1} &= \mathbf{W}_t + \Delta\mathbf{W}_t \end{aligned}$$

$$\begin{bmatrix} \Delta w_{11} \ldots \Delta w_{1A} \\ \vdots \\ \Delta w_{B1} \ldots \Delta w_{BA} \end{bmatrix} = \begin{bmatrix} \hat{y}_1^{\alpha} \\ \vdots \\ \hat{y}_B^{\alpha} \end{bmatrix} \begin{bmatrix} x_1^{\alpha} & \ldots & x_A^{\alpha} \end{bmatrix}$$

Where p is the number of training examples.

After presentation of p patterns, $\mathbf{W} = \sum_{\alpha=1}^{p} \mathbf{y}^{\alpha}\mathbf{x}^{\alpha\,T}$ or $w_{ji} = \sum y_j^{\alpha} x_i^{\alpha}$.

Analyzing Retrieval

Suppose the input patterns $\mathbf{X}$ are orthonormal, i.e., normalized such that

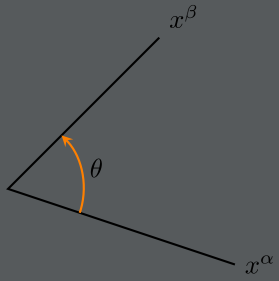
$$\begin{aligned} \|\mathbf{x}^\alpha\| &= \sqrt{\mathbf{x}^{\alpha\,T} \cdot \mathbf{x}^\alpha} \\ &= \sqrt{\sum x_i^{\alpha\,2}} \\ &= 1 \end{aligned}$$

and $\mathbf{x}^\alpha$ and $\mathbf{x}^\beta$ are orthogonal, $\alpha \neq \beta$, $\sum x_i^\alpha x_i^\beta = 0$.

Analyzing Retrieval

For normalized vectors, dot product measures similarity:

$$cos(\theta) = \mathbf{x}^{\alpha\,T} \cdot \mathbf{x}^{\beta}$$
$$= \frac{\mathbf{x}^{\alpha\,T} \cdot \mathbf{x}^{\beta}}{\|\mathbf{x}^{\alpha}\| \, \|\mathbf{x}^{\beta}\|}$$



$$\cos(0) = 1$$
$$\cos(90) = 0$$

Analyzing Retrieval

Given the input to a stored example, $\mathbf{x}^\alpha$, what will the model retrieve? (Note that $\mathbf{x}^{\beta^T}\mathbf{x}^\alpha$ is 0 when $\beta \neq \alpha$.)

$$
\begin{aligned}
\mathbf{y} &= \mathbf{W}\mathbf{x}^\alpha \\
&= (\sum_{\beta=1}^{p} \mathbf{y}^\beta \mathbf{x}^{\beta^T})\mathbf{x}^\alpha \\
&= \mathbf{y}^1 \mathbf{x}^{1^T} \mathbf{x}^\alpha + \ldots + \mathbf{y}^p \mathbf{x}^{p^T} \mathbf{x}^\alpha \\
&= \mathbf{y}^\alpha \mathbf{x}^{\alpha^T} \mathbf{x}^\alpha \\
&= \mathbf{y}^\alpha
\end{aligned}
$$

Inference with Non-Orthogonal Inputs

Suppose two examples are stored – $(\mathbf{x}^1, \mathbf{y}^1)$ and $(\mathbf{x}^2, \mathbf{y}^2)$ and $\mathbf{x}^1$ is not orthogonal to $\mathbf{x}^2$ (e.g. ${\mathbf{x}^1}^T\mathbf{x}^2 = .2$, angle is $cos^{-1}(.2) = 78.5°$.)

What will the model retrieve given input $\mathbf{x}^1$?

$$\begin{aligned}
\mathbf{y} &= \mathbf{W}\mathbf{x}^1 \\
&= (\mathbf{y}^1{\mathbf{x}^1}^T + \mathbf{y}^2{\mathbf{x}^2}^T)\mathbf{x}^1 \\
&= \mathbf{y}^1{\mathbf{x}^1}^T\mathbf{x}^1 + \mathbf{y}^2{\mathbf{x}^2}^T\mathbf{x}^1 \\
&= \mathbf{y}^1 + 0.2\mathbf{y}^2
\end{aligned}$$

Interference of example 2 on example 1 is related to similarity of $\mathbf{x}^1$ and $\mathbf{x}^2$ (i.e. the angle between them).

Inference with Non-Orthogonal Inputs - Generalization

Suppose two examples are stored – $(\mathbf{x}^1, \mathbf{y}^1)$ and $(\mathbf{x}^2, \mathbf{y}^2)$ – and the model is probed with another input $\mathbf{x}^\alpha$.

$$\begin{aligned}
\mathbf{y} &= \mathbf{W}\mathbf{x}^\alpha \\
&= (\mathbf{y}^1{\mathbf{x}^1}^T + \mathbf{y}^2{\mathbf{x}^2}^T)\mathbf{x}^\alpha \\
&= \mathbf{y}^1[{\mathbf{x}^1}^T\mathbf{x}^\alpha] + \mathbf{y}^2[{\mathbf{x}^2}^T\mathbf{x}^\alpha]
\end{aligned}$$

The model produces $\mathbf{y}^\alpha$ with magnitude (strength) proportional to the similarity of $\mathbf{x}^\alpha$ and $\mathbf{x}^\beta$.

Inference with Non-Orthogonal Inputs - Generalization

Suppose two examples are stored – $(\mathbf{x}^1, \mathbf{y}^1)$ and $(\mathbf{x}^2, \mathbf{y}^2)$ – and the model is probed with input $\mathbf{x}^\alpha = .5\mathbf{x}^1 + .5\mathbf{x}^2$.

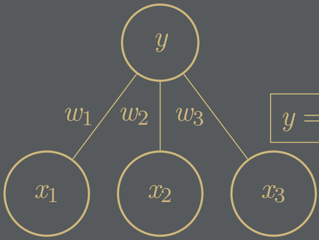
$$
\begin{aligned}
\mathbf{y} &= \mathbf{W}\mathbf{x}^\alpha \\
&= (\mathbf{y}^1{\mathbf{x}^1}^T + \mathbf{y}^2{\mathbf{x}^2}^T)\mathbf{x}^\alpha = .5\mathbf{x}^1 + .5\mathbf{x}^2 \\
&= .5(\mathbf{y}^1{\mathbf{x}^1}^T\mathbf{x}^1 + \cancel{\mathbf{y}^1{\mathbf{x}^1}^T\mathbf{x}^2} + \cancel{\mathbf{y}^2{\mathbf{x}^2}^T\mathbf{x}^1} + \mathbf{y}^2{\mathbf{x}^2}^T\mathbf{x}^2) \\
&= .5\mathbf{y}^1 + .5\mathbf{y}^2
\end{aligned}
$$

An input that is the interpolation of two stored inputs results in the retrieval of the interpolation of the corresponding outputs.

LMS Weight Update Rule

Can we do better than Hebb rule?

What would the optimal set of weights be?



$$\boxed{y = \sum w_i x_i}$$

$$
\begin{aligned}
{x_1}^1 w_1 + {x_2}^1 w_2 \quad &+ {x_3}^1 w_3 = y^1 \\
{x_1}^2 w_1 + {x_2}^2 w_2 \quad &+ {x_3}^2 w_3 = y^2 \\
&\vdots \\
{x_1}^k w_1 + {x_2}^k w_2 \quad &+ {x_3}^k w_3 = y^k
\end{aligned}
$$

Find weights that satisfy a system of linear equations.

General case is many output units – we'll focus on a single output unit.

Normal Equation

If input vectors span the input space (i.e. every input vector can be expressed as a weighted combination of the $\mathbf{x}^{\alpha}$), then the LMS solution can be obtained via the normal equation

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$$

Limitations of using the normal equation for solving for $\mathbf{W}$:

- Space complexity: all training examples must be in memory simultaneously
- Time complexity: matrix inversion is $O(n^3)$

LMS Weight Update Rule

What if there is no set of weights that makes all equations true (e.g. when there are more examples to be learned (equations) than weights)?

Answer: Find least mean squares (LMS) solution – the set of weights that minimizes the error E

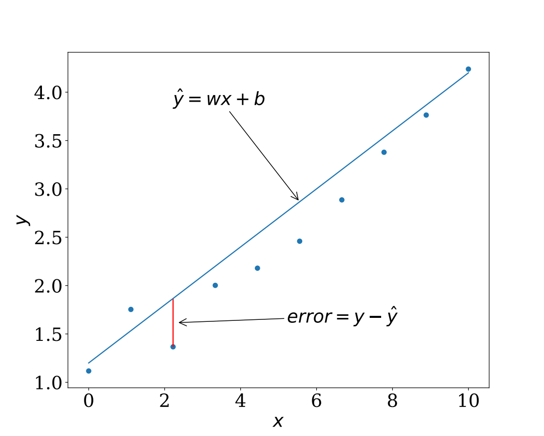
$$E = \frac{1}{p} \sum_{\alpha=1}^{p} \frac{1}{2} (\hat{y}^{\alpha} - y^{\alpha})^2$$

where $\hat{y}$ is the output of the network and y is the ground truth, and p is the number of examples.

This is linear regression – finding the set of coefficients that best predicts one variable ($y \in \mathbb{R}$) from some other variables ($\mathbf{x} \in \mathbb{R}^k$, where k is the number of features).

LMS Weight Update Rule

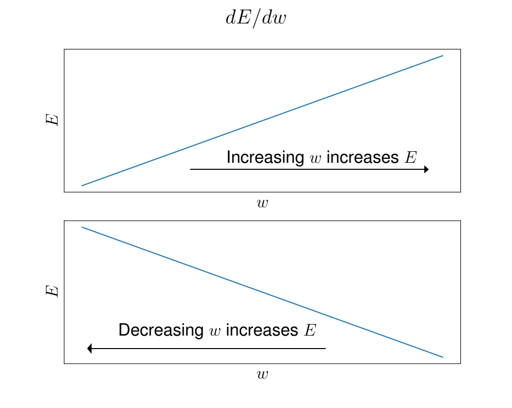
Consider a network with one input and output, $y = wx + b$.



Each point corresponds to a training example.

Gradient Descent (with Scalar w)

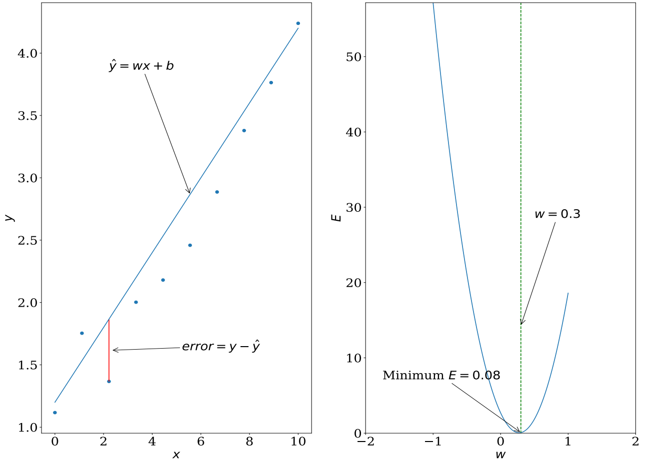
Strategy: iteratively adjust the weights to decrease error.



1. Use the slope of the error surface, dE/dw_t, to determine direction to change w_t, where t is the timestep.
2. $w_{t+1} = w_t + \Delta w_t = w_t - \epsilon \frac{dE}{dw_t}$, where ϵ is the *learning rate* or *step size*.

With a scalar weight, we obtain the *derivative* $\frac{dE}{dw}$. We obtain the *gradient* $\frac{\partial E}{\partial w_{ij}}$ with a matrix weight $\mathbf{W}$.

Gradient Descent (with Scalar w)



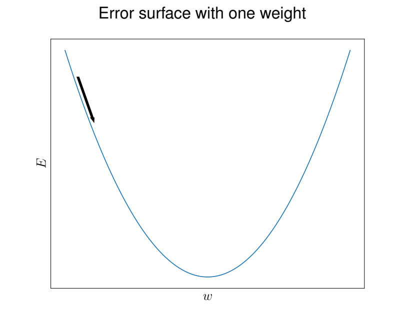
Gradient Descent (with Matrix $\mathbf{W}$)

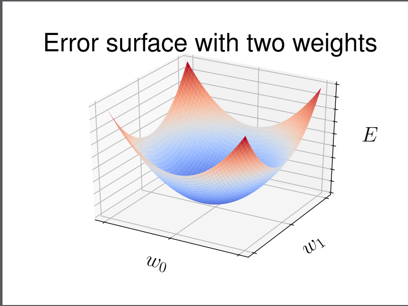
$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}} &= \frac{\partial}{\partial \mathbf{W}} \frac{1}{p} \sum_{\alpha=1}^{p} \frac{1}{2} (\mathbf{W}\mathbf{x}^{\alpha} - y^{\alpha})^2 \\
&= \frac{1}{p} \sum_{\alpha=1}^{p} \frac{\partial}{\partial \mathbf{W}} \frac{1}{2} (\mathbf{W}\mathbf{x}^{\alpha} - y^{\alpha})^2 \\
&= \frac{1}{p} \sum_{\alpha=1}^{p} (\mathbf{W}\mathbf{x}^{\alpha} - y^{\alpha}) \frac{\partial}{\partial \mathbf{W}} (\mathbf{W}\mathbf{x}^{\alpha} - y^{\alpha}) \\
&= \frac{1}{p} \sum_{\alpha=1}^{p} (\mathbf{W}\mathbf{x}^{\alpha} - y^{\alpha}) \mathbf{x}^{\alpha}
\end{aligned}
$$

Error Surface

Convex, quadratic in $\mathbf{W}$ (i.e. highest exponent of $\frac{\partial E}{\partial \mathbf{W}}$ is 2).
Training procedure:

1. Start at a random point in weight space.
2. Modify weights so as to move downhill in error.





$$\frac{dE}{dw}$$

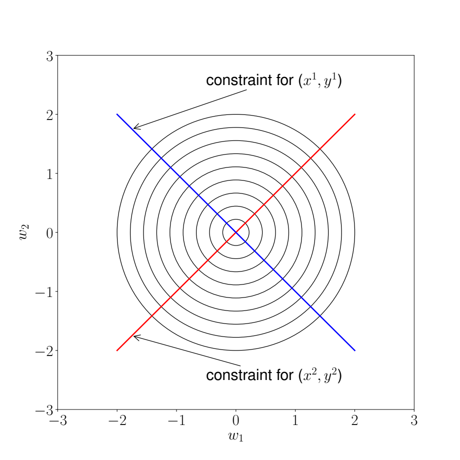
$$\left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}\right)$$

Online Versus Batch Learning

Suppose we have a network with two inputs, one output, and our dataset consists of two examples, $(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2)$.

Suppose the weights must satisfy the constraints

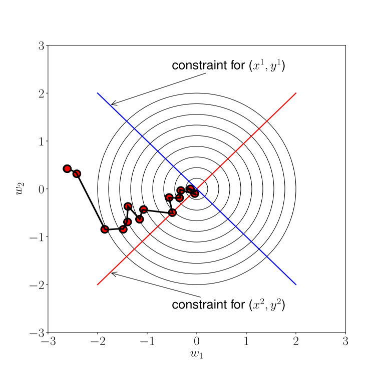
$$x_1{}^1 w_1 + x_2{}^1 w_2 = y^1$$
$$x_1{}^2 w_1 + x_2{}^2 w_2 = y^2$$



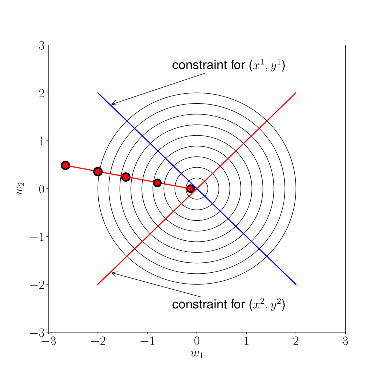
Online Versus Batch Learning

Two approaches to computing gradients:

1. Online: after each example, i.e. $\Delta\mathbf{W} = (\hat{y}^\alpha - y^\alpha)\mathbf{x}^\alpha$.
2. Batch: after all examples, i.e., $\Delta\mathbf{W} = \frac{1}{p}\sum_{\alpha=1}^{p}(\hat{y}^\alpha - y^\alpha)\mathbf{x}^\alpha$.



Stochastic Gradient Descent



Batch Gradient Descent