CSE 5526: Introduction to Neural Networks

Linear Regression
Problem statement
Problem statement

(a) Unknown stochastic environment: \( \mathbf{w} \) generates responses \( d \) given input \( \mathbf{x} \).

(b) Detailed diagram showing regression \( x_i \rightarrow w_i \rightarrow \cdots \rightarrow d \) with error \( \varepsilon \).
Linear regression with one variable

• Given a set of $N$ pairs of data $<x_i, d_i>$, approximate $d$ by a linear function of $x$ (regressor)
  
i.e.

$$d \approx wx + b$$

or

$$d_i = y_i + \varepsilon_i = \varphi(wx_i + b) + \varepsilon_i$$

$$= wx_i + b + \varepsilon_i$$

where the activation function $\varphi(x) = x$ is a linear function, and it corresponds to a linear neuron. $y$ is the output of the neuron, and

$$\varepsilon_i = d_i - y_i$$

is called the regression (expectational) error
Linear regression (cont.)

• The problem of regression with one variable is how to choose $w$ and $b$ to minimize the regression error

• The least squares method aims to minimize the square error $E$:

$$E = \frac{1}{2} \sum_{i=1}^{N} \varepsilon_i^2 = \frac{1}{2} \sum_{i=1}^{N} (d_i - y_i)^2$$
Linear regression (cont.)

• To minimize the two-variable square function, set

$$\begin{align*}
\frac{\partial E}{\partial b} &= 0 \\
\frac{\partial E}{\partial w} &= 0
\end{align*}$$
Linear regression (cont.)

\[
\frac{\partial E}{\partial b} = \frac{1}{2} \sum_i \frac{\partial (d_i - wx_i - b)^2}{\partial b} = - \sum_i (d_i - wx_i - b) = 0
\]

\[
\frac{\partial E}{\partial w} = \frac{1}{2} \sum_i \frac{\partial (d_i - wx_i - b)^2}{\partial w} = - \sum_i (d_i - wx_i - b)x_i = 0
\]
Linear regression (cont.)

- Hence

\[
b = \frac{\sum_{i} x_i^2 \sum_{i} d_i - \sum_{i} x_i \sum_{i} x_i d_i}{N \left[ \sum_{i} (x_i - \bar{x})^2 \right]}
\]

\[
w = \frac{\sum_{i} (x_i - \bar{x})(d_i - \bar{d})}{\sum_{i} (x_i - \bar{x})^2}
\]

where an overbar (i.e. \(\bar{x}\)) indicates the mean

Derive yourself!
Linear regression (cont.)

- This method gives an optimal solution, but it can be time- and memory-consuming as a batch solution
Finding optimal parameters via search

• Without loss of generality, set $b = 0$

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} (d_i - wx_i)^2$$

$E(w)$ is called a cost function
Question: how can we update $w$ to minimize $E$?
Gradient and directional derivatives

Without loss of generality, consider a two-variable function \( f(x, y) \). The gradient of \( f(x, y) \) at a given point \((x_0, y_0)^T\) is

\[
\nabla f = \left( \frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y} \right)^T \bigg|_{x=x_0}^{y=y_0} \\
= f_x(x_0, y_0) \mathbf{u}_x + f_y(x_0, y_0) \mathbf{u}_y
\]

where \( \mathbf{u}_x \) and \( \mathbf{u}_y \) are unit vectors in the \( x \) and \( y \) directions, and

\( f_x = \frac{\partial f}{\partial x} \) and \( f_y = \frac{\partial f}{\partial y} \)
Gradient and directional derivatives (cont.)

• At any given direction, \( \mathbf{u} = a \mathbf{u}_x + b \mathbf{u}_y \), with \( \sqrt{a^2 + b^2} = 1 \), the directional derivative at \((x_0, y_0)^T\) along the unit vector \( \mathbf{u} \) is

\[
D_{\mathbf{u}} f(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0)}{h}
\]

\[
= \lim_{h \to 0} \frac{[f(x_0 + ha, y_0 + hb) - f(x_0, y_0 + hb)] + [f(x_0, y_0 + hb) - f(x_0, y_0)]}{h}
\]

\[
= af_x(x_0, y_0) + bf_y(x_0, y_0)
\]

\[
= \nabla f^T(x_0, y_0) \mathbf{u}
\]

• Which direction has the greatest slope?
  – The gradient because of the dot product!
Gradient and directional derivatives (cont.)

• Example: see blackboard
Gradient and directional derivatives (cont.)

- To find the gradient at a particular point \((x_0, y_0)^T\), first find the level curve or contour of \(f(x, y)\) at that point, \(C(x_0, y_0)\). A tangent vector \(u\) to \(C\) satisfies

\[
D_u = \nabla f^T (x_0, y_0) u = 0
\]

because \(f(x, y)\) is constant on a level curve. Hence the gradient vector is perpendicular to the tangent vector.
An illustration of level curves
Gradient and directional derivatives (cont.)

- The gradient of a cost function is a vector with the dimension of $\mathbf{w}$ that points to the direction of maximum $E$ increase and with a magnitude equal to the slope of the tangent of the cost function along that direction
  - Can the slope be negative?
Gradient illustration

\[ \nabla E(w_0) = \lim_{\Delta w \to 0} \frac{E(w_0 + \Delta w) - E(w_0 - \Delta w)}{2\Delta w} \]

Diagram showing the gradient of the energy function \( E(w) \) with a point \( w^* \) where \( E_{\text{min}} \) is attained, and the gradient vector \( \nabla E(w_0) \) at \( w_0 \).
Gradient descent

• Minimize the cost function via gradient (steepest) descent – a case of hill-climbing

\[ w(n + 1) = w(n) - \eta \nabla E(n) \]

\( n \): iteration number
\( \eta \): learning rate

• See previous figure
Gradient descent (cont.)

• For the mean-square-error cost function:

\[ E(n) = \frac{1}{2} e^2(n) = \frac{1}{2} [d(n) - y(n)]^2 \]

\[ = \frac{1}{2} [d(n) - w(n)x(n)]^2 \quad \text{linear neurons} \]

\[ \nabla E(n) = \frac{\partial E}{\partial w(n)} = \frac{1}{2} \frac{\partial e^2(n)}{\partial w(n)} \]

\[ = -e(n)x(n) \]
Gradient descent (cont.)

• Hence

\[ w(n + 1) = w(n) + \eta e(n)x(n) \]

\[ = w(n) + \eta[d(n) - y(n)]x(n) \]

• This is the least-mean-square (LMS) algorithm, or the Widrow-Hoff rule
Multi-variable case

- The analysis for the one-variable case extends to the multi-variable case

\[
E(n) = \frac{1}{2} [d(n) - w^T(n)x(n)]^2
\]

\[
\nabla E(w) = \left( \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_m} \right)^T
\]

where \( w_0 = b \) (bias) and \( x_0 = 1 \), as done for perceptron learning
Multi-variable case (cont.)

• The LMS algorithm

\[ w(n + 1) = w(n) - \eta \nabla E(n) \]

\[ = w(n) + \eta e(n)x(n) \]

\[ = w(n) + \eta [d(n) - y(n)]x(n) \]
LMS algorithm

- **Remarks**
  - The LMS rule is exactly the same in math form as the perceptron learning rule.
  - Perceptron learning is for McCulloch-Pitts neurons, which are nonlinear, whereas LMS learning is for linear neurons. In other words, perceptron learning is for classification and LMS is for function approximation.
  - LMS should be less sensitive to noise in the input data than perceptrons. On the other hand, LMS learning converges slowly.
  - Newton’s method changes weights in the direction of the minimum $E(w)$ and leads to fast convergence. But it is not an online version and computationally extensive.
Stability of adaptation

- When $\eta$ is too small, learning converges slowly
Stability of adaptation (cont.)

- When $\eta$ is too large, learning doesn’t converge
Learning rate annealing

• Basic idea: start with a large rate but gradually decrease it
• Stochastic approximation

\[ \eta(n) = \frac{c}{n} \]

\( c \) is a positive parameter
Learning rate annealing (cont.)

• Search-then-converge

$$\eta(n) = \frac{\eta_0}{1 + (n/\tau)}$$

$\eta_0$ and $\tau$ are positive parameters

• When $n$ is small compared to $\tau$, learning rate is approximately constant
• When $n$ is large compared to $\tau$, learning rate schedule roughly follows stochastic approximation
Rate annealing illustration

\[ \eta(n) \]

\[ \eta_0 \]

\[ \tau \]

Slope = \(-c\)

Search-then-converge schedule

Stochastic approximation schedule

\( n \)

(log scale)
Nonlinear neurons

• To extend the LMS algorithm to nonlinear neurons, consider differentiable activation function $\phi$ at iteration $n$

$$E(n) = \frac{1}{2} [d(n) - y(n)]^2$$

$$= \frac{1}{2} [d(n) - \phi(\sum_j w_j x_j(n))]^2$$
By chain rule of differentiation

\[
\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial v} \frac{\partial v}{\partial w_j}
\]

\[
= -[d(n) - y(n)]\varphi'(\nu(n))x_j(n)
\]

\[
= -e(n)\varphi'('nu(n))x_j(n)
\]
Nonlinear neurons (cont.)

• The gradient descent gives

\[ w_j(n + 1) = w_j(n) + \eta e(n) \phi'(v(n)) x_j(n) \]

\[ = w_j(n) + \eta \delta(n) x_j(n) \]

• The above is called the delta (\( \delta \)) rule

• If we choose a logistic sigmoid for \( \phi \)

\[ \phi(v) = \frac{1}{1 + \exp(-av)} \]

then

\[ \phi'(v) = a \phi(v)[1 - \phi(v)] \] (see textbook)
Role of activation function

- The role of $\varphi'$: weight update is most sensitive when $\nu$ is near zero