DATA MINING 2
(Deep) Neural Networks

Riccardo Guidotti

a.a. 2023/2024

Slides edited from a set of slides titled “Introduction to Machine Learning and Neural Networks” by Davide Bacciu
Nonlinearly Separable Data

• Since \( f(w,x) \) is a linear combination of input variables, decision boundary is linear.

• For nonlinearly separable problems, the perceptron fails because no linear hyperplane can separate the data perfectly.

• An example of nonlinearly separable data is the XOR function.

\[
y = x_1 \oplus x_2
\]

XOR Data

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Why Now?

(Big) Data

GPU

Theory
A quick look on Deep Learning
Deep learning

**Representation learning** methods that

- allow a machine to be fed with raw data and
- to automatically discover the representations needed for detection or classification.

**Raw representation**

- Age: 35
- Weight: 65
- Income: 23 k€
- Children: 2
- Likes sport: 0.3
- Likes reading: 0.6
- Education: high
- ...

**Higher-level representation**

- Young parent: 0.9
- Fit sportsman: 0.1
- High-educated reader: 0.8
- Rich obese: 0.0
- ...
- ...

---

Note: The diagram illustrates the process of how raw data is transformed into higher-level representations through deep learning methods.
Multiple Levels Of Abstraction
Multilayer Neural Network

- **Hidden Layers**: intermediary layers between input and output layers.
- More general **activation functions** (sigmoid, linear, hyperbolic tangent, etc.).
- Multi-layer neural network can solve any type of classification task involving nonlinear decision surfaces.
- Perceptron is single layer.
- We can think to each hidden node as a perceptron that tries to construct one hyperplane, while the output node combines the results to return the decision boundary.

XOR Data
General Structure of ANN

Training ANN means learning the weights of the neurons.

Activation function $g(S_i)$

\[ S_i = \sum w_i x_i + t \]

\[ O_i = g(S_i) \]

Input Layer

Hidden Layer

Output Layer

$x_1, x_2, x_3, x_4, x_5$

$I_1, I_2, I_3$

$w_{i1}, w_{i2}, w_{i3}$

$y$

$t$
Artificial Neural Networks (ANN)

• Various types of neural network topology
  • single-layered network (perceptron) versus multi-layered network
  • Feed-forward versus recurrent network

• Various types of activation functions ($f$)

\[ Y = f\left(\sum_i w_i X_i \right) \]
Deep Neural Networks
Deep Neural Networks

Input $x_1 \ldots x_M$

Hidden Layer 1 $a_1 \ldots a_D$

Hidden Layer 2 $b_1 \ldots b_M$

Output $y$

$W \times x_1 + b_1 \rightarrow a_1 \rightarrow b_1 \rightarrow y$
Deep Neural Networks

Backpropagation through many layers has numerical problems that makes learning not-straightforward (Gradient Vanish/Explosion)

Actually deep learning is way more than having neural networks with a lot of layers
Representation Learning

• We don’t know the “right” levels of abstraction of information that is good for the machine
• So let the model figure it out!

Example from Honglak Lee (NIPS 2010)
Representation Learning

**Face Recognition:**
- Deep Network can build up increasingly higher levels of abstraction
- Lines, parts, regions

Example from Honglak Lee (NIPS 2010)
Representation Learning

Example from Honglak Lee (NIPS 2010)
Activation Functions

• A new change: modifying the nonlinearity
  • The logistic is not widely used in modern ANNs

Alternative 1:
tanh
Like logistic function but shifted to range [-1, +1]
Activation Functions

Alternative 2: rectified linear unit

Linear with a cutoff at zero

(Implementation: clip the gradient when you pass zero)

\[ \text{max}(0, w \cdot x + b). \]
Activation Functions

Alternative 3: soft exponential linear unit

Soft version: $\log(e^x + 1)$

Doesn’t saturate (at one end)
Sparsifies outputs
Helps with vanishing gradient
Activation Functions Summary

Hyperbolic Tangent:
\[
f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
1 & \text{for } x \geq 0 
\end{cases}
\]

Softmax Function:
\[
f(x_j) = \frac{e^{x_j}}{\sum_k e^{x_k}}
\]

Sigmoid Function:
\[
f(x) = \frac{1}{1 + e^{-x}}
\]
Learning Multi-layer Neural Network

• Can we apply perceptron learning to each node, including hidden nodes?
• Perceptron computes error $e = y - f(w, x)$ and updates weights accordingly
• Problem: how to determine the true value of $y$ for hidden nodes?
• Approximate error in hidden nodes by error in the output nodes
• Problems:
  • Not clear how adjustment in the hidden nodes affect overall error
  • No guarantee of convergence to optimal solution
Gradient Descent for Multilayer NN

- Error function to minimize: \( E = \frac{1}{2} \sum_{i=1}^{N} (y_i - f(\sum_j w_j x_{ij}))^2 \)

- Weight update: \( w_j^{(k+1)} = w_j^{(k)} - \lambda \frac{\partial E}{\partial w_j} \)

- Activation function \( f \) must be differentiable

- For sigmoid function: \( w_j^{(k+1)} = w_j^{(k)} + \lambda \sum_i (y_i - o_i) o_i (1 - o_i) x_{ij} \)

- Stochastic Gradient Descent (update the weight immediately)
Gradient Descent for Multilayer NN

- Weights are updated in the opposite direction of the gradient of the loss function.
- Gradient direction is the direction of uphill of the error function.
- By taking the negative we are going downhill.
- Hopefully to a minimum of the error.

\[ w^{(k+1)}_j = w^{(k)}_j - \lambda \frac{\partial E}{\partial w_j} \]
Gradient Descent for Multilayer NN

• For output neurons, weight update formula is the same as before (gradient descent for perceptron)

• For hidden neurons:

\[ w^{(k+1)}_{pi} = w^{(k)}_{pi} + \lambda o_i (1 - o_i) \sum_{j \in \Phi_i} \delta_j w_{ij} x_{pi} \]

Output neurons: \( \delta_j = o_j (1 - o_j) (t_j - o_j) \)

Hidden neurons: \( \delta_j = o_j (1 - o_j) \sum_{k \in \Phi_j} \delta_k w_{jk} \)

\( o \): output of the network  
\( t \): target value (ground truth)
Training Multilayer NN

(A) Input
Given $x_i$, $\forall i$

(B) Hidden (linear)
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \forall j$$

(C) Hidden (sigmoid)
$$z_j = \frac{1}{1+\exp(-a_j)}, \forall j$$

(D) Output (linear)
$$b = \sum_{j=0}^{D} \beta_j z_j$$

(E) Output (sigmoid)
$$y = \frac{1}{1+\exp(-b)}$$
Training Multilayer NN

How do we update these weights given the loss is available only at the output unit?

\[ E(y, y^*) \]

Output

Hidden Layer

Input

(A) Input

Given \( x_i, \forall i \)

(B) Hidden (linear)

\[ a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \forall j \]

(C) Hidden (sigmoid)

\[ z_j = \frac{1}{1+\exp(-a_j)}, \forall j \]

(D) Output (linear)

\[ b = \sum_{j=0}^{D} \beta_j z_j \]

(E) Output (sigmoid)

\[ y = \frac{1}{1+\exp(-b)} \]

\( (\in) \) Loss

\[ E = \frac{1}{2} (y - y^*)^2 \]
Error Backpropagation

Error is computed at the output and propagated back to the input by chain rule to compute the contribution of each weight (a.k.a. derivative) to the loss.

A 2-step process
1. Forward pass - Compute the network output
2. Backward pass – Compute the loss function gradients and update

Error Backpropagation - Example
The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.
Error Backpropagation - Example

- initial weights
- biases
- training inputs/outputs
- activation: logistic
Example - The Forward Pass

\[ net_{h1} = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1 \]

\[ net_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 \times 1 = 0.3775 \]
Example - The Forward Pass

\[ \text{net}_{h1} = w_1 \cdot i_1 + w_2 \cdot i_2 + b_1 \cdot 1 \]

\[ \text{net}_{h1} = 0.15 \cdot 0.05 + 0.2 \cdot 0.1 + 0.35 \cdot 1 = 0.3775 \]

\[ \text{out}_{h1} = \frac{1}{1 + e^{-\text{net}_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992 \]
Example - The Forward Pass

\[
net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1 \\
net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775 \\
out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992 \\
out_{h2} = 0.596884378
\]
Example - The Forward Pass

\[ net_{o1} = w_5 \times out_{h1} + w_6 \times out_{h2} + b_2 \times 1 \]

\[ net_{o1} = 0.4 \times 0.593269992 + 0.45 \times 0.596884378 + 0.6 \times 1 = 1.10 \]

\[ out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.055903587}} = 0.75136507 \]

\[ out_{o2} = 0.772928465 \]
Example – Calculating the Total Error

\[ E_{total} = \sum \frac{1}{2} (target - output)^2 \]

\[ E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083 \]

\[ E_{o2} = 0.023560026 \]

\[ E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109 \]
Example - The Backward Pass

How much a change in w5 affects the total error?
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_1} \ast \frac{\partial \text{out}_1}{\partial \text{net}_1} \ast \frac{\partial \text{net}_1}{\partial w_5}
\]

\[
\frac{\partial \text{net}_1}{\partial w_5} \ast \frac{\partial \text{out}_1}{\partial \text{net}_1} \ast \frac{\partial E_{\text{total}}}{\partial \text{out}_1} = \frac{\partial E_{\text{total}}}{\partial w_5}
\]

\[
E_{o_1} = \frac{1}{2}(\text{target}_{o_1} - \text{out}_{o_1})^2
\]

\[
E_{\text{total}} = E_{o_1} + E_{o_2}
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

how much does the total error change with respect to the output?
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}
\]

\[
E_{total} = \frac{1}{2} (target_{o1} - out_{o1})^2 + \frac{1}{2} (target_{o2} - out_{o2})^2
\]

how much does the total error change with respect to the output?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \cdot \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
E_{\text{total}} = \frac{1}{2}(\text{target}_{o1} - \text{out}_{o1})^2 + \frac{1}{2}(\text{target}_{o2} - \text{out}_{o2})^2
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} = 2 \cdot \frac{1}{2}(\text{target}_{o1} - \text{out}_{o1})^{2-1} \cdot -1 + 0
\]

how much does the total error change with respect to the output?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \cdot \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
E_{\text{total}} = \frac{1}{2}(\text{target}_{o1} - \text{out}_{o1})^2 + \frac{1}{2}(\text{target}_{o2} - \text{out}_{o2})^2
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} = 2 \cdot \frac{1}{2}(\text{target}_{o1} - \text{out}_{o1})^{2-1} \cdot -1 + 0
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} = -(\text{target}_{o1} - \text{out}_{o1}) = -(0.01 - 0.75136507) = 0.74136507
\]

how much does the total error change with respect to the output?
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}
\]

how much does the output $o_1$ change with respect to its total net input?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \cdot \frac{\partial \text{out}_o}{\partial \text{net}_o} \cdot \frac{\partial \text{net}_o}{\partial w_5}
\]

\[
\text{out}_o = \frac{1}{1 + e^{-\text{net}_o}}
\]

how much does the output o1 change with respect to its total net input?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \cdot \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\text{out}_{o1} = \frac{1}{1 + e^{-\text{net}_{o1}}}
\]

\[
\frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = \text{out}_{o1}(1 - \text{out}_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602
\]

How much does the output o1 change with respect to its total net input?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \times \frac{\partial \text{out}_o}{\partial \text{net}_o} \times \frac{\partial \text{net}_o}{\partial w_5}
\]

how much does the total net input of o1 change with respect to w5?
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial o_{\text{out}1}} \times \frac{\partial o_{\text{out}1}}{\partial n_{\text{et}1}} \times \frac{\partial n_{\text{et}1}}{\partial w_5}
\]

\[
n_{\text{et}1} = w_5 \times o_{\text{out}h1} + w_6 \times o_{\text{out}h2} + b_2 \times 1
\]

how much does the total net input of o1 change with respect to w5?
**Example - The Backward Pass**

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1
\]

\[
\frac{\partial \text{net}_{o1}}{\partial w_5} = 1 \times \text{out}_{h1} \times w_5^{(1-1)} + 0 + 0 = \text{out}_{h1} = 0.593269992
\]

How much does the total net input of o1 change with respect to w5?
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}
\]

\[
\frac{\partial E_{total}}{\partial w_5} = 0.74136507 \times 0.186815602 \times 0.593269992 = 0.082167041
\]
Example - The Backward Pass

Slope of the Activation Function obtained as partial derivative by the Gradient Descent

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial o_{o1}} \times \frac{\partial o_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = -(target_{o1} - o_{o1}) \times o_{o1} (1 - o_{o1}) \times o_{h1}
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = - (\text{target}_{o1} - \text{out}_{o1}) \times \text{out}_{o1} (1 - \text{out}_{o1}) \times \text{out}_{h1}
\]

\[
\delta_{o1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{\text{total}}}{\partial \text{net}_{o1}}
\]

Rewriting as delta rule
Example - The Backward Pass

\[ \frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \cdot \frac{\partial \text{net}_{o1}}{\partial w_5} \]

\[ \frac{\partial E_{\text{total}}}{\partial w_5} = -(target_{o1} - \text{out}_{o1}) \cdot \text{out}_{o1} (1 - \text{out}_{o1}) \cdot \text{out}_{h1} \]

\[ \delta_{o1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{\text{total}}}{\partial \text{net}_{o1}} \]

\[ \delta_{o1} = -(target_{o1} - \text{out}_{o1}) \cdot \text{out}_{o1} (1 - \text{out}_{o1}) \]

Rewriting as delta rule
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \cdot \frac{\partial \text{out}_o}{\partial \text{net}_o} \cdot \frac{\partial \text{net}_o}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = -(\text{target}_o - \text{out}_o) \cdot \text{out}_o(1 - \text{out}_o) \cdot \text{out}_{h1}
\]

\[
\delta_{o1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \cdot \frac{\partial \text{out}_o}{\partial \text{net}_o} = \frac{\partial E_{\text{total}}}{\partial \text{net}_o}
\]

\[
\delta_{o1} = -(\text{target}_o - \text{out}_o) \cdot \text{out}_o(1 - \text{out}_o)
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \delta_{o1} \cdot \text{out}_{h1}
\]

Rewriting as delta rule
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \cdot \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 0.74136507 \cdot 0.186815602 \cdot 0.593269992 = 0.082167041
\]

\[
w_5^+ = w_5 - \eta \cdot \frac{\partial E_{\text{total}}}{\partial w_5} = 0.4 - 0.5 \cdot 0.082167041 = 0.35891648
\]

Apply the step size to update \(w_5\).
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \times \frac{\partial \text{out}_o}{\partial \text{net}_o} \times \frac{\partial \text{net}_o}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 0.74136507 \times 0.186815602 \times 0.593269992 = 0.082167041
\]

\[
w_5^+ = w_5 - \eta \times \frac{\partial E_{\text{total}}}{\partial w_5} = 0.4 - 0.5 \times 0.082167041 = 0.35891648
\]

Apply the step size to update w5.
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial w_5}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 0.74136507 \times 0.186815602 \times 0.593269992 = 0.082167041
\]

\[
w_5^+ = w_5 - \eta \times \frac{\partial E_{\text{total}}}{\partial w_5} = 0.4 - 0.5 \times 0.082167041 = 0.35891648
\]

The same calculus is applied to update w6, w7 and w8.
Example - The Backward Pass

After that $w_5$, $w_6$, $w_7$ and $w_8$ have been updated we continue backwards to update $w_1$, $w_2$, $w_3$ and $w_4$. 
After that $w_5$, $w_6$, $w_7$ and $w_8$ have been updated we continue backwards to update $w_1$, $w_2$, $w_3$ and $w_4$.
Example - The Backward Pass

\[ \frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \ast \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \ast \frac{\partial \text{net}_{h1}}{\partial w_1} \]

\[ \frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \ast \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \ast \frac{\partial \text{net}_{h1}}{\partial w_1} \]

\[ \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{\text{out1}}}{\partial \text{out}_{h1}} + \frac{\partial E_{\text{out2}}}{\partial \text{out}_{h1}} \]

\[ E_{\text{total}} = E_{\text{out1}} + E_{\text{out2}} \]
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial \text{out}_h} \star \frac{\partial \text{out}_h}{\partial \text{net}_h} \star \frac{\partial \text{net}_h}{\partial w_1}
\]
Example - The Backward Pass

\[ \frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1} \]

\[ \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}} \]
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \ast \frac{\partial out_{h1}}{\partial net_{h1}} \ast \frac{\partial net_{h1}}{\partial w_1}
\]

\[
\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}
\]

\[
\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} \ast \frac{\partial net_{o1}}{\partial out_{h1}}
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1}
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}}
\]

\[
\frac{\partial E_{o1}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}}
\]

\[
\frac{\partial E_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = 0.74136507 \times 0.186815602 = 0.138498562
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h_1}} \ast \frac{\partial \text{out}_{h_1}}{\partial \text{net}_{h_1}} \ast \frac{\partial \text{net}_{h_1}}{\partial w_1}
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{h_1}} = \frac{\partial E_{o_1}}{\partial \text{out}_{h_1}} + \frac{\partial E_{o_2}}{\partial \text{out}_{h_1}}
\]

\[
\frac{\partial E_{o_1}}{\partial \text{out}_{h_1}} = \frac{\partial E_{o_1}}{\partial \text{net}_{o_1}} \ast \frac{\partial \text{net}_{o_1}}{\partial \text{out}_{h_1}}
\]

\[
\frac{\partial E_{o_1}}{\partial \text{net}_{o_1}} = \frac{\partial E_{o_1}}{\partial \text{out}_{o_1}} \ast \frac{\partial \text{out}_{o_1}}{\partial \text{net}_{o_1}} = 0.74136507 \ast 0.186815602 = 0.138498562
\]

\[
\text{net}_{o_1} = w_5 \ast \text{out}_{h_1} + w_6 \ast \text{out}_{h_2} + b_2 \ast 1
\]
Example - The Backward Pass

\[ \frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1} \]

\[ \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}} \]

\[ \frac{\partial E_{o1}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} \]

\[ \frac{\partial E_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = 0.74136507 \times 0.186815602 = 0.138498562 \]

\[ \text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1 \]

\[ \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} = w_5 = 0.40 \]
Example - The Backward Pass

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}}$$

$$\frac{\partial E_{o1}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} = 0.138498562 \times 0.40 = 0.055399425$$

$$\frac{\partial E_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = 0.74136507 \times 0.186815602 = 0.138498562$$

$$\text{net}_{o1} = \text{w}_5 \times \text{out}_{h1} + \text{w}_6 \times \text{out}_{h2} + b_2 \times 1$$

$$\frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} = \text{w}_5 = 0.40$$
Example - The Backward Pass

\[ \frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1} \]

\[ \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}} \]

\[ \frac{\partial E_{o1}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{net}_{o1}} \times \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} = 0.138498562 \times 0.40 = 0.055399425 \]

\[ \frac{\partial E_{o1}}{\partial \text{net}_{o1}} = \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \times \frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = 0.74136507 \times 0.186815602 = 0.138498562 \]

\[ \text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1 \]

\[ \frac{\partial \text{net}_{o1}}{\partial \text{out}_{h1}} = w_5 = 0.40 \]

Same process is followed for
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}
\]

\[
\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + (-0.019049119) = 0.036350306
\]
Example - The Backward Pass

\[ \frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1} \]

\[ out_{h1} = \frac{1}{1 + e^{-net_{h1}}} \]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \cdot \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \cdot \frac{\partial \text{net}_{h1}}{\partial w_1}
\]

\[
\text{out}_{h1} = \frac{1}{1 + e^{-\text{net}_{h1}}}
\]

\[
\frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} = \text{out}_{h1}(1 - \text{out}_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709
\]
Example - The Backward Pass

\[ \frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \ast \frac{\partial out_{h1}}{\partial net_{h1}} \ast \frac{\partial net_{h1}}{\partial w_1} \]

net_{h1} = w_1 \ast i_1 + w_3 \ast i_2 + b_1 \ast 1
Example - The Backward Pass

\[
\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1}
\]

\[
\text{net}_{h1} = w_1 \times i_1 + w_3 \times i_2 + b_1 \times 1
\]

\[
\frac{\partial \text{net}_{h1}}{\partial w_1} = i_1 = 0.05
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = 0.036350306 \times 0.241300709 \times 0.05 = 0.000438568
\]
Example - The Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_h1} \times \frac{\partial \text{out}_h1}{\partial \text{net}_h1} \times \frac{\partial \text{net}_h1}{\partial w_1}
\]

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = 0.036350306 \times 0.241300709 \times 0.05 = 0.000438568
\]

\[
w_1^+ = w_1 - \eta \times \frac{\partial E_{\text{total}}}{\partial w_1} = 0.15 - 0.5 \times 0.000438568 = 0.149780716
\]
Backpropagation in other words

• In order to get the loss of a node (e.g. Z0), we multiply the value of its corresponding $f'(z)$ by the loss of the node it is connected to in the next layer (delta_1), by the weight of the link connecting both nodes.

• We do the delta calculation step at every unit, back-propagating the loss into the neural net, and finding out what loss every node/unit is responsible for.

On the Key Importance of Error Functions

• The error/loss/cost function reduces all the various good and bad aspects of a possibly complex system down to a single number, a scalar value, which allows candidate solutions to be compared.

• It is important, therefore, that the function faithfully represent our design goals.

• If we choose a poor error function and obtain unsatisfactory results, the fault is ours for badly specifying the goal of the search.
# Objective Functions for NN

- **Regression**: A problem where you predict a real-value quantity.
  - Output Layer: One node with a linear activation unit.
  - Loss Function: Quadratic Loss (Mean Squared Error (MSE))

- **Classification**: Classify an example as belonging to one of K classes
  - Output Layer:
    - One node with a sigmoid activation unit (K=2, binary cross-entropy)
    - K output nodes in a softmax layer (K>2, categorical cross-entropy)*
  - Loss function: Cross-entropy (i.e. negative log likelihood)

*When K > 2 the target variable needs to be one-hot encoded

\[ J = \sum y^* \log(y) \]  

Cross Entropy (categorical)

<table>
<thead>
<tr>
<th>Function</th>
<th>Forward</th>
<th>Backward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic</td>
<td>[ J = \frac{1}{2} (y - y^*)^2 ]</td>
<td>[ \frac{dJ}{dy} = y - y^* ]</td>
</tr>
<tr>
<td>Cross Entropy (binary)</td>
<td>[ J = y^* \log(y) + (1 - y^*) \log(1 - y) ]</td>
<td>[ \frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{y - 1} ]</td>
</tr>
</tbody>
</table>
Design Issues in ANN

• Number of nodes in input layer
  • One input node per binary/continuous attribute
  • \( k \) or \( \log_2 k \) nodes for each categorical attribute with \( k \) values

• Number of nodes in output layer
  • One output for binary class problem
  • \( k \) or \( \log_2 k \) nodes for \( k \)-class problem

• Number of nodes in hidden layer

• Initial weights and biases
Characteristics of ANN

• Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large.
• Gradient descent may converge to local minimum.
• Model building can be very time consuming, but testing can be very fast.
• Can handle redundant attributes because weights are automatically learnt.
• Sensitive to noise in training data.
• Difficult to handle missing attributes.
Tips and Tricks of NN Training
Dataset Should Normally be Split Into

- **Training set**: use to update the weights. Records in this set are repeatedly in random order. The weight update equation are applied after a certain number of records.

- **Validation set**: use to decide when to stop training only by monitoring the error and to select the best model configuration

- **Test set**: use to test the performance of the neural network. It should not be used as part of the neural network development and model selection cycle
Before Starting: Weight Initialization

• Choice of *initial weight values is important as this decides starting position in weight space*. That is, how far away from global minimum
  • Aim is to select weight values which produce midrange function signals
  • Select weight values randomly from uniform probability distribution
  • Normalize weight values so number of weighted connections per unit produces midrange function signal

• Try different random initialization to
  • Assess robustness
  • Have more opportunities to find optimal results
Two learning fashion (plus one)

- **Sequential mode** (on-line, stochastic, or per-record)
  - Weights updated after each record is presented
  - Many weight updates, can quicker convergence but also make learning less stable

- **Batch mode** (off-line or per-epoch)
  - Weights updated after all records are presented
  - Can be very slow and lead to trapping in early local minima

- **Minibatch mode** (a blend of the two above)
  - Weights updated after a few records (from tens to thousands) are presented
  - Best of both (and good for GPU)
Convergence Criteria

• Learning is obtained by repeatedly supplying training data and adjusting by backpropagation
  • Typically 1 training set presentation = 1 epoch

• We need a stopping criteria to define convergence
  • Euclidean norm of the gradient vector reaches a sufficiently small value
  • Absolute rate of change in the average squared error per epoch is sufficiently small
  • Validation for generalization performance: stop when generalization performance reaches a peak
Early Stopping

• Running too many epochs may **overtrain** the network and result in **overfitting** and perform poorly in generalization

• Keep a hold-out validation set and test accuracy after every epoch. Maintain weights for best performing network on the validation set and stop training when error increases beyond this

• Always let the network run for some epochs before deciding to stop (**patience parameter**), then backtrack to best result
Model Selection

• **Too few hidden units** prevent the network from learning adequately fitting the data and learning the concept.

• **Too many hidden units** leads to overfitting, unless you regularize heavily (e.g. dropout, weight decay, weight penalties).

• Cross validation should be used to determine an appropriate number of hidden units by using the optimal validation error to select the model with optimal number of hidden layers and nodes.
Regularization

• Constrain the learning model to avoid overfitting and help improving generalization.

• Add **penalization terms** to the loss function that *punish* the model for excessive use of resources
  • Limit the **number of weights** that is used to learn a task
  • Limit the **total activation of neurons** in the network

\[
E' = E(y, y^*) + \lambda R(\cdot)
\]

Hyperparameter to be chosen in model selection  
\(R(W_\theta)\)  Penalty on parameters  
\(R(Z)\)  Penalty on activations
Common penalty terms (norms)

• 1-norm $||A||_1 = \sum_{ij} |a_{ij}|$
  • Parameters: $R(W_\theta) = ||W_\theta||_1^2$
  • Activations: $R(Z(X)) = ||Z(X)||_1^2$ (Z hidden unit activation)

• 2-norm $||A||_2 = \sqrt{\sum_{ij} a_{ij}^2}$
  • Parameters: $R(W_\theta) = ||W_\theta||_2^2$
  • Activations: $R(Z(X)) = ||Z(X)||_2^2$ (Z hidden unit activation)

• Any $p$-norm and more...
Dropout Regularization

Randomly disconnect units from the network during training
Dropout Regularization

Randomly disconnect units from the network during training
Dropout Regularization

Randomly disconnect units from the network during training
Dropout Regularization

Randomly disconnect units from the network during training

- Regulated by unit **dropping hyperparameter**
- Prevents unit **coadaptation**
- Committee machine effect
- Need to adapt **prediction phase**
- Used at prediction time gives predictions with confidence intervals

You can also **drop single connections** (dropconnect)
Momentum

• Adding a term to weight update equation to store an exponentially weight history of previous weights changes
• Reducing problems of instability while increasing the rate of convergence
  • If weight changes tend to have same signs, the momentum term increases, and gradient decrease speed up convergence on shallow gradient
  • If weight changes tend have opposing signs, the momentum term decreases, and gradient descent slows to reduce oscillations (stabilizes)
• Can help escape being trapped in local minima
Choosing the Optimization Algorithm

• Standard Stochastic Gradient Descent (SGD)
  • Easy and efficient
  • Difficult to pick up the best learning rate
  • Unstable convergence
  • Often used with momentum (exponentially weighted history of previous weights changes)

• RMSprop
  • Adaptive learning rate method (reduces it using a moving average of the squared gradient)
  • Fastens convergence by having quicker gradients when necessary

• Adagrad
  • Like RMSprop with element-wise scaling of the gradient

• ADAM
  • Like Adagrad but adds an exponentially decaying average of past gradients like momentum
Convolutional Neural Networks

- Are typically applied for the classification of images and time series
- Instead of having only “fully connected” layers adopt “convolutional layers”

![Diagram of Convolutional Neural Network]

- 32x32x3 image
- 5x5x3 filter $w$
- 1 number: the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. $5*5*3 = 75$-dimensional dot product + bias)

$$w^T x + b$$
Recurrent Neural Network

- Are typically applied in natural language processing (NLP).

Key idea: RNNs have an “internal state” that is updated as a sequence is processed.

\[ h_t = f_W[h_{t-1}, x_t] \]

- new state
- old state
- input vector at some time step

Some function with parameters W
Convolutional Neural Network

Slides edited from Stanford

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

input

1
3072

\[ Wx \]
10 x 3072 weights

activation

1
10
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

- **Input**: 3072
- **$Wx$**: 10 x 3072 weights
- **Activation**: 1 number: the result of taking a dot product between a row of $W$ and the input (a 3072-dimensional dot product)
Convolution Layer

32x32x3 image -> preserve spatial structure
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”
Convolution Layer

The result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. $5 \times 5 \times 3 = 75$-dimensional dot product + bias)

$$w^T x + b$$
Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolution Layer

Image

Convolved Feature

Convolution Kernel

1 0 1
0 1 0
1 0 1
Convolution Layer

Input Channel #1 (Red)  
0 0 0 0 0 0 
0 0 0 156 155 154 
0 153 154 157 159 159 
0 149 151 155 158 159 
0 146 146 149 151 159 
0 145 143 143 148 158 

Input Channel #2 (Green)  
0 0 0 0 0 0 
0 0 0 164 163 165 
0 160 162 166 165 166 
0 160 162 166 169 170 
0 0 0 0 0 0 

Input Channel #3 (Blue)  
0 0 0 0 0 0 
0 0 0 163 162 163 
0 160 162 164 166 166 
0 156 158 162 165 166 
0 155 155 158 162 167 

Kernel Channel #1  
-1 -1 1  
0 1 -1  
0 1 1  

Kernel Channel #2  
1 0 0  
1 -1 -1  
1 0 -1  

Kernel Channel #3  
0 1 1  
0 1 0  
1 -1 1  

Output  
-25 
...
...
...
...
...
...
...
...

Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps
Convolution Layer

For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolutional Neural Network
Convolutional Neural Network

• CNN is a sequence of Conv Layers, interspersed with activation functions.
• CNN shrinks volumes spatially.
• E.g. 32x32 input convolved repeatedly with 5x5 filters! (32 -> 28 -> 24 ...).
• Shrinking too fast is not good, doesn’t work well.
CNN for Image Classification

Low-level features → Mid-level features → High-level features → Linearly separable classifier

VGG-16 Conv1_1 → VGG-16 Conv3_2 → VGG-16 Conv5_3
Stride

7x7 input (spatially)
assume 3x3 filter

=> 5x5 output
Stride

7x7 input (spatially) assume 3x3 filter applied with stride 2 => 3x3 output!
Stride

7x7 input (spatially) assume 3x3 filter applied with stride 3?

doesn’t fit!
cannot apply 3x3 filter on 7x7 input with stride 3.
Stride

Output size:
\[(N - F) / \text{stride} + 1\]

e.g. \(N = 7, F = 3\):
- \(\text{stride 1} \Rightarrow (7 - 3)/1 + 1 = 5\)
- \(\text{stride 2} \Rightarrow (7 - 3)/2 + 1 = 3\)
- \(\text{stride 3} \Rightarrow (7 - 3)/3 + 1 = 2.33\)
Padding

In general, common to see CONV layers with stride 1, filters of size $F \times F$, and zero-padding with $(F-1)/2$.

- $F = 3 \Rightarrow$ zero pad with 1 pixel
- $F = 5 \Rightarrow$ zero pad with 2 pixels
- $F = 7 \Rightarrow$ zero pad with 3 pixels

Example: input 7x7

3x3 filter, applied with stride 1
pad with 1 pixel border => what is the output?

7x7 output!
Convolution Summary

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently
MaxPooling and AvgPooling
Pooling Summary

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent $F$,
  - the stride $S$,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers
Example of CNN

INPUT
(28 x 28 x 1)

Conv_1
Convolution
(5 x 5) kernel
valid padding

Max-Pooling
(2 x 2)

Conv_2
Convolution
(5 x 5) kernel
valid padding

Max-Pooling
(2 x 2)

fc_3
Fully-Connected
Neural Network
ReLU activation

fc_4
Fully-Connected
Neural Network
(with dropout)

OUTPUT
n3 units

28 x 28 x 1
24 x 24 x n1
12 x 12 x n1
8 x 8 x n2
4 x 4 x n2
Recurrent Neural Network

Slides edited from Stanford

Types of Recurrent Neural Networks

one to one
Vanilla NN
Image --> Sequence of Words
Image Captioning

one to many
Sequence of Words --> Sentiment

many to one
Sequence of Words --> TS Classification

many to many
Sequence of Words --> Sequence of Words
Machine Translation

Video Classification
Recurrent Neural Network - RNN

Key idea: RNNs have an “internal state” that is updated as a sequence is processed.
Recurrent Neural Network - RNN

- We can process a sequence of vectors $x$ by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

- The new state is a function of the old state and the input vector at some time step with parameters $W$. 

- The output $y$ is connected to the RNN.
Unfolded RNN
RNN: Computational Graph

Reminder: Re-use the same weight matrix at every time-step
RNN: Computational Graph: Many to Many

[Diagram showing the computational graph of an RNN with many to many connections, including input $x$, hidden states $h_0, h_1, h_2, h_3, \ldots, h_T$, and outputs $y_1, y_2, y_3, \ldots, y_T$, with transformations $f_w$ and weight matrix $W$.]
RNN: Computational Graph: Many to One
RNN: Example Training

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
RNN: Example Training

Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
RNN: Example Training

Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Example:
Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model
RNN: Example Test

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
RNN: Example Test

Example:
Character-level Language Model Sampling

Vocabulary: [h,e,i,o]

At test-time sample characters one at a time, feed back to model
References

• Artificial Neural Network. Chapter 5.4 and 5.5. Introduction to Data Mining.

• Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow. A practical handbook to start wrestling with Machine Learning models (2nd ed).

Exercises - Neural Network
Predict with a Neural Network

- Given the following NN with
  - assigned weights (see figure)
  - activation function $f(S) = \text{sign}(S-0.2)$ for all nodes

- Label the test set on the right, then compute accuracy, and precision & recall for both classes

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
<td>+1</td>
<td></td>
</tr>
</tbody>
</table>
Predict with a Neural Network - Solution

- **Given the following NN with**
  - assigned weights (see figure)
  - activation function \( f(S) = \text{sign}(S-0.2) \) for all nodes

- **Label the test set on the right, then compute accuracy and precision & recall for both classes**

\[
\begin{align*}
H_1 &= \text{sign}(0.4 \times -1 + 0.1 \times 1 - 0.2) = \\
&= \text{sign}(-0.5) = -1 \\
H_2 &= \text{sign}(0.0 \times -1 + -0.4 \times 1 - 0.2) = \\
&= \text{sign}(-0.6) = -1 \\
H_3 &= \text{sign}(-0.1 \times -1 + 0.4 \times 1 - 0.2) = \\
&= \text{sign}(0.3) = 1 \\
Y_1 &= \text{sign}(0.2 \times -1 + 0.2 \times -1 + 0.3 \times 1 - 0.2) = \\
&= \text{sign}(-0.3) = -1
\end{align*}
\]

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>
Predict with a Neural Network

Given the neural network below (on the left), apply it to the test set provided (on the right). The weights are reported beside each connection, while the activation function is simply $f(S) = \text{sign}(S)$, i.e. -1 for positive values, +1 for positive ones and 0 for $S=0$. For each case, show the output also of the nodes on the hidden layer.

```
<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td>+0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Predict with a Neural Network - Solution

Answer:

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>-1</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>+0</td>
<td></td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td></td>
<td>+1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input1</th>
<th>0</th>
<th>1</th>
<th>-1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input2</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>H1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>H2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>H3</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>Output</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>