University of Idaho
Department of Computer Science
Coeur d'Alene

Lecture #23

Neural Networks

Garrett Wells
revised October 15, 2024

Table of Contents I

Table of Contents I

Table of Contents
Review
Neural Networks vs. Human Brains
Feedforward
Backpropagation
Learning Optimizations
Additional Topics

Review

Machine Learning Paradigms

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Review
Machine Learning Paradigms
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Review

Perceptrons

- Borrowing from the human biological model for learning

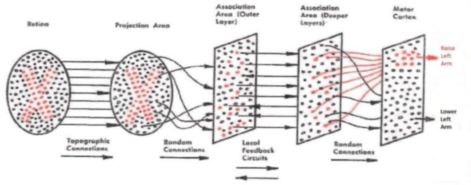


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

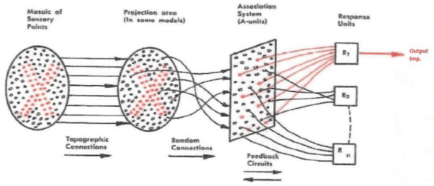


FIG. 2 — Organization of a perceptron.

2024-10-15

Lecture #23
└Review
└Review

Review
Perceptrons
- Borrowing from the human biological model for learning

Review

Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes

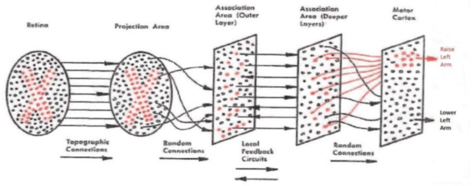


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

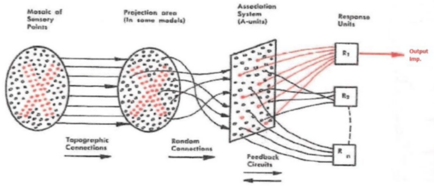
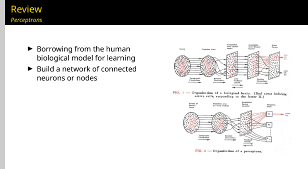


FIG. 2 — Organization of a perceptron.

2024-10-15

Lecture #23
└Review

└Review



Review

Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated

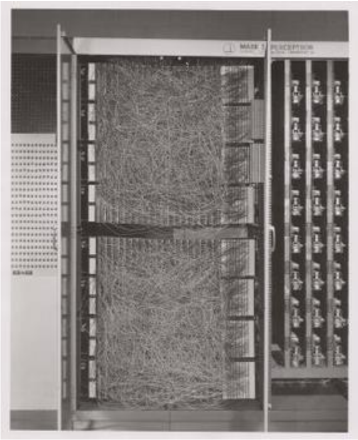


Figure: Mark I Perceptron

Review
Perceptrons
- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated

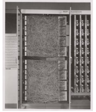


Figure: Mark I Perceptron

Review

Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated
- Activation: output of activation function is passed to child nodes

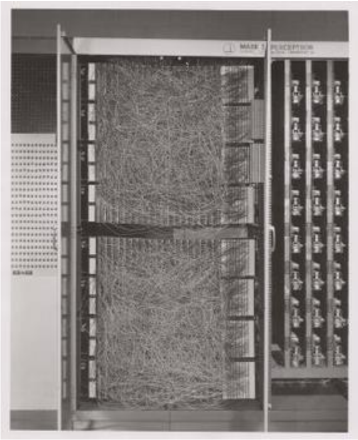


Figure: Mark I Perceptron

Review

Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated
- Activation: output of activation function is passed to child nodes

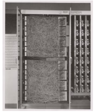


Figure: Mark I Perceptron

Review

Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated
- Activation: output of activation function is passed to child nodes
- Original perceptron only had 3 layers, random neuron connections

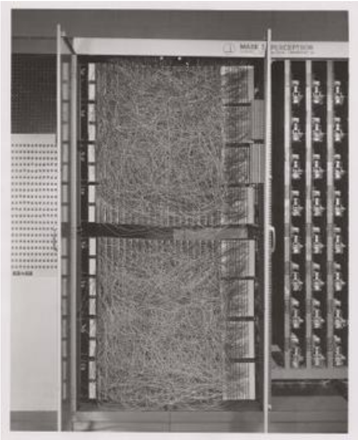


Figure: Mark I Perceptron

Review
Perceptrons

- Borrowing from the human biological model for learning
- Build a network of connected neurons or nodes
- If inputs to nodes exceed threshold → node is activated
- Activation: output of activation function is passed to child nodes
- Original perceptron only had 3 layers, random neuron connections

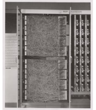


Figure: Mark I Perceptron

Neural Network

An artificial mathematical model used to approximate non-linear functions[1].

Remeber. . .

Neural Networks
Neural Network
An artificial mathematical model used to approximate non-linear functions[1].
Remeber. . .

Neural Network

An artificial mathematical model used to approximate non-linear functions[1].

Remeber...

- Perceptrons accused of not being able to learn non-linear functions

Neural Networks
Neural Network
An artificial mathematical model used to approximate non-linear functions[1].
Remeber...
► Perceptrons accused of not being able to learn non-linear functions

Neural Networks

Neural Network

An artificial mathematical model used to approximate non-linear functions[1].

Remeber...

- This was mostly due to the fixed depth(number of layers) of the perceptron

Neural Networks

Neural Network

An artificial mathematical model used to approximate non-linear functions[1].

Remeber...

- This was mostly due to the fixed depth(number of layers) of the perceptron

Neural Network

An artificial mathematical model used to approximate non-linear functions[1].

Remeber...

- Neural Networks a broad class of ML models

Neural Networks
Neural Network
An artificial mathematical model used to approximate non-linear functions[1].
Remeber...
Neural Networks a broad class of ML models

Neural Networks

Characteristics & Parameters. . .

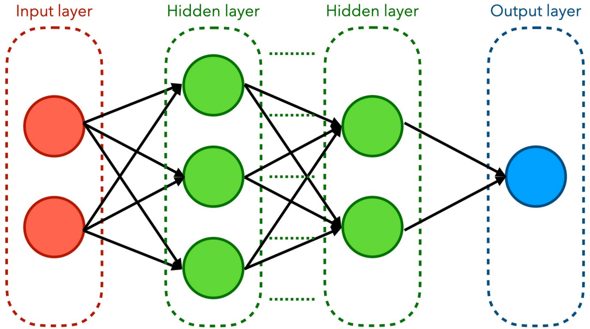
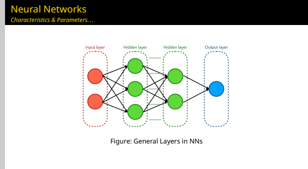


Figure: General Layers in NNs



Neural Networks

Characteristics & Parameters...

Input Layer $\boldsymbol{x}$, input

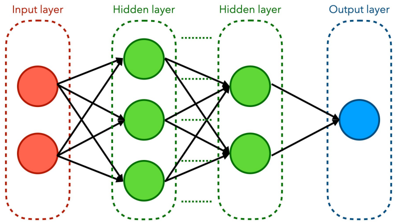


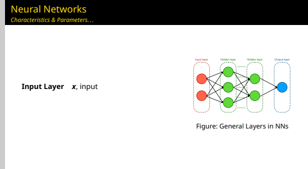
Figure: General Layers in NNs

2024-10-15

Lecture #23

└─Review

└─Neural Networks



Neural Networks

Characteristics & Parameters...

Hidden Layers Flexible, changes between architectures

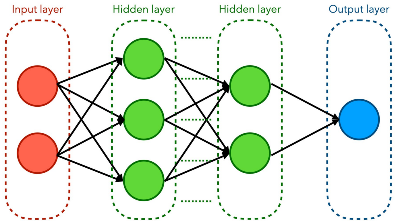


Figure: General Layers in NNs

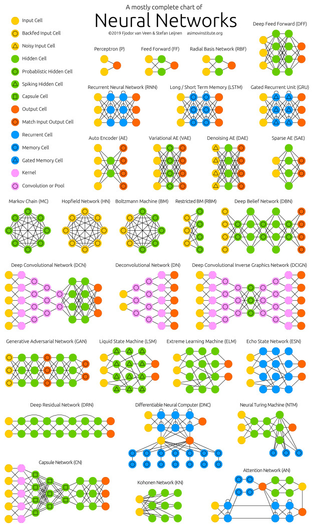
Neural Networks
Characteristics & Parameters...
Hidden Layers Flexible, changes between architectures
Figure: General Layers in NNs

Neural Networks

Characteristics & Parameters…

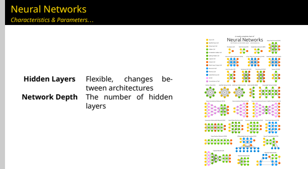
Hidden Layers Flexible, changes between architectures

Network Depth The number of hidden layers



2024-10-15

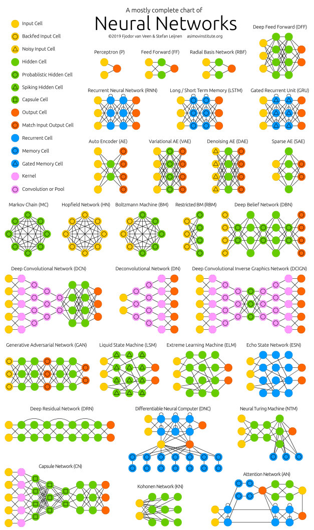
Lecture #23
└Review
└Neural Networks


Neural Networks
Characteristics & Parameters…
Hidden Layers Flexible, changes between architectures
Network Depth The number of hidden layers

Neural Networks

Characteristics & Parameters…

Hidden Layers	Flexible, changes between architectures
Network Depth	The number of hidden layers
Layer Breadth/Width	How many nodes per layer

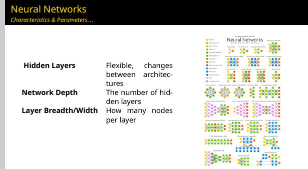


2024-10-15

Lecture #23

└Review

└Neural Networks


Neural Networks
Characteristics & Parameters…

Neural Networks

Characteristics & Parameters...

Output Layer ***y***, Same number of nodes as classes

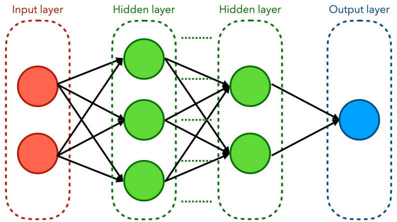
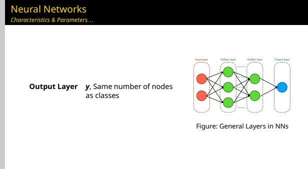


Figure: General Layers in NNs

2024-10-15

Lecture #23
└Review
└Neural Networks



Neural Networks

Setting up a neural network...

1. Specify architecture
   - Number of layers
   - Number of nodes per layer
   - Activation function for nodes in layer
   - Specify output layer & activation function
2. Initialize weight values
3. Initialize bias values

Neural Networks
Setting up a neural network...
1. Specify architecture
- Number of layers
- Number of nodes per layer
- Activation function for nodes in layer
- Specify output layer & activation function
2. Initialize weight values
3. Initialize bias values

Neural Network Complexity vs. Human Brain Complexity

Are neural networks equivalent to the human brain?

- Human Brain: 10^{11} [2], $\geq$ 80 billion [3]
- **2018** – 16 million, analagous to frog brain[4]
- **2021** Natural Language Processing(NLP): approx. 110 million
- GPT-3: 125 million – 125 billion

Neural Network Complexity vs. Human Brain Complexity
Are neural networks equivalent to the human brain?
Human Brain: 10^{11} [2], $\geq$ 80 billion [3]
2018 – 16 million, analagous to frog brain[4]
2021 Natural Language Processing(NLP): approx. 110 million
GPT-3: 125 million – 125 billion

Neural Network Operation

Resources...

1. Deep Learning – Goodfellow *et. al.* [5]
2. Neural Networks and Deep Learning – [6]
3. Numerous online resources

Neural Network Operation
Resources...
1. Deep Learning – Goodfellow et. al. [5]
2. Neural Networks and Deep Learning – [6]
3. Numerous online resources

Neural Network Operation

Feedforward...

Feedforward

1. Feed input feature vector into the network input layer.
2. Inputs pass through linear function to calculate an activation value.
   - ☞ $a = wx + b$
3. The activation function of each node is run on the activation value producing the output value of the node.
4. Final output values from output layer determine model prediction.

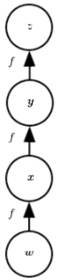


Figure: Minimal NN from [5]

Neural Network Operation
Feedforward...
Feedforward
1. Feed input feature vector into the network input layer.
2. Inputs pass through linear function to calculate an activation value.
☞ $a = wx + b$
3. The activation function of each node is run on the activation value producing the output value of the node.
4. Final output values from output layer determine model prediction.
Figure: Minimal NN from [5]

Neural Network Operation

Feedforward...

Feedforward

Network prediction, z, expressed as:

$$z = f(y) = f(f(x)) \tag{1}$$

$$y = f(x) = f(f(w)) \tag{2}$$

w	is the input to the network(scalar)
f	is the activation function

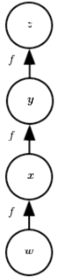


Figure: Minimal NN from [5]

2024-10-15

Lecture #23
└Feedforward
└Neural Network Operation

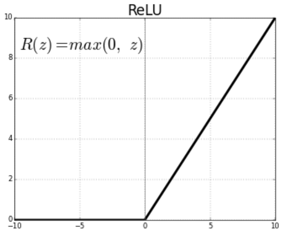
Neural Network Operation
Feedforward...
Feedforward
Network prediction, z, expressed as:
$z = f(y) = f(f(x))$ (1)
$y = f(x) = f(f(w))$ (2)
w is the input to the network(scalar)
f is the activation function
Figure: Minimal NN from [5]

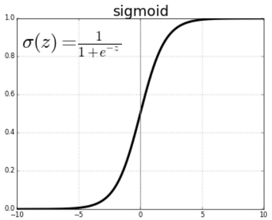
Neural Network Operation

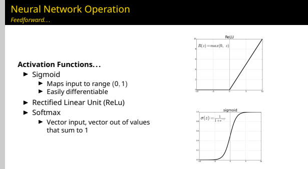
Feedforward…

Activation Functions…

- Sigmoid
  - Maps input to range $(0, 1)$
  - Easily differentiable
- Rectified Linear Unit (ReLu)
- Softmax
  - Vector input, vector out of values that sum to 1







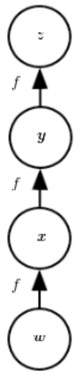
1. Activation function impacts other network functions. For example, with ReLu, bias must be non-zero value when initialized because if zero there is no gradient.
2. Sigmoid is sensitive to change when value is near to zero, however slope means that it is slow to respond to updates when maxed out.
3. Activation functions are not always linear (ReLu) introducing problems when differentiating.

Neural Network Operation

Feedforward…

Implications

1. We can calculate a gradient(partial derivative) between each layer of the network
2. The gradient indicates how much each node contributed to the final prediction



Neural Network Operation
Feedforward…
Implications
1. We can calculate a gradient(partial derivative) between each layer of the network
2. The gradient indicates how much each node contributed to the final prediction

Neural Network Operation

Feedforward...

- Iterate through all layers in network
- For each node in current layer calculate $a_i = \sum_i x_i w_i + b$
  - x_i input/output from node i in previous layer
  - w_i the weight associated with the i layer
  - b, bias, which is the same for all nodes in layer
- Output of node, $u_i = f(a_i)$
  - f, activation function

Neural Network Operation
Feedforward...
- Iterate through all layers in network
- For each node in current layer calculate $a_i = \sum_i x_i w_i + b$
  - x_i input/output from node i in previous layer
  - w_i the weight associated with the i layer
  - b, bias, which is the same for all nodes in layer
- Output of node, $u_i = f(a_i)$
  - f, activation function

Neural Network Operation

Feedforward...

- Now we have a prediction, y and a label(ground truth) value $\hat{y}$.
- Multiple nodes contributed to (wrong) prediction.
- How do we blame those nodes and correct behavior?
- Phrased differently, how do we tune our model to fit the input function?

Neural Network Operation
Feedforward...
- Now we have a prediction, y and a label(ground truth) value $\hat{y}$.
- Multiple nodes contributed to (wrong) prediction.
- How do we blame those nodes and correct behavior?
- Phrased differently, how do we tune our model to fit the input function?

Neural Network Operation

Feedforward...

1. Need to quantify distance between prediction y and answer $\hat{y}$.

Neural Network Operation
Feedforward...
1. Need to quantify distance between prediction y and answer $\hat{y}$.

Neural Network Operation

Feedforward...

1. Need to quantify distance between prediction y and answer $\hat{y}$.
   ☞ *Error/Loss calculation*

2024-10-15

Lecture #23
└─Feedforward

└─Neural Network Operation

Neural Network Operation
Feedforward...
1. Need to quantify distance between prediction y and answer $\hat{y}$.
☞ *Error/Loss calculation*

Neural Network Operation

Feedforward...

1. Need to quantify distance between prediction y and answer $\hat{y}$.
   - ☞ *Error/Loss calculation*
2. Gives us lump sum of "blame" to distribute. How do we distribute intelligently?

Neural Network Operation
Feedforward...
1. Need to quantify distance between prediction y and answer $\hat{y}$.
☞ Error/Loss calculation
2. Gives us lump sum of "blame" to distribute. How do we distribute intelligently?

Neural Network Operation

Feedforward...

1. Need to quantify distance between prediction y and answer $\hat{y}$.
   - ☞ *Error/Loss calculation*
2. Gives us lump sum of “blame” to distribute. How do we distribute intelligently?
   - ☞ *Backpropagation*

Neural Network Operation
Feedforward...
1. Need to quantify distance between prediction y and answer $\hat{y}$.
☞ Error/Loss calculation
2. Gives us lump sum of “blame” to distribute. How do we distribute intelligently?
☞ Backpropagation

Backpropagation

Using the prediction from the feedforward algorithm and the ground truth label, calculate the loss/cost/error.

- Loss is distributed among weights and bias, updating them to optimize model performance. This is done by calculating the gradient of the loss w.r.t. each node.

Neural Network Operation
Backpropagation...
Backpropagation
Using the prediction from the feedforward algorithm and the ground truth label, calculate the loss/cost/error.
Loss is distributed among weights and bias, updating them to optimize model performance. This is done by calculating the gradient of the loss w.r.t. each node.

Neural Network Operation

Backpropagation...

Mean Squared Error (MSE)

"MSE...is calculated as teh average of the squared differences between the predicted and actual values." [7]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \quad (1)$$

Cross-Entropy

Penalty is logarithmic, penalizing heavily for large deviance from expected value.

$$H(p, q) = -\sum_{x} p(x)log(q(x)) \quad (2)$$

Neural Network Operation
Backpropagation...
Mean Squared Error (MSE)
"MSE...is calculated as teh average of the squared differences between the predicted and actual values." [7]
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \quad (1)$$
Cross-Entropy
Penalty is logarithmic, penalizing heavily for large deviance from expected value.
$$H(p, q) = -\sum_{x} p(x)log(q(x)) \quad (2)$$

Neural Network Operation

Backpropagation...

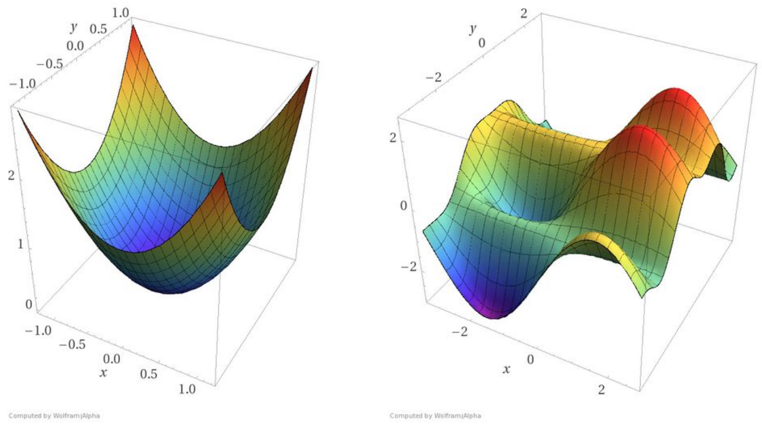
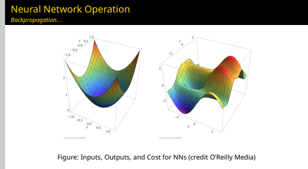


Figure: Inputs, Outputs, and Cost for NNs (credit O'Reilly Media)



1. error function can be thought of as a 3D surface
2. error is *z* axis, with respect to two independent variables *x* and *y*

Neural Network Operation

Backpropagation...

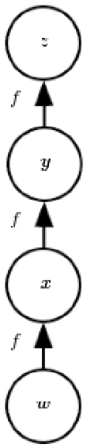
Remember... our prediction as a chain of functions:

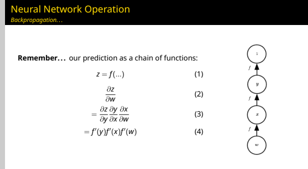
$$z = f(\dots) \tag{1}$$

$$\frac{\partial z}{\partial w} \tag{2}$$

$$= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} \tag{3}$$

$$= f'(y)f'(x)f'(w) \tag{4}$$





1. We can determine the gradient(change) between each node using the chain rule.
2. This gradient is a quantification of how much each node's output impacts our final prediction.
3. Backpropagation uses this chain to distribute error among participating nodes.
4. All that is missing is our loss calculation.

Neural Network Operation

Backpropagation…

Including Loss[1]**…**

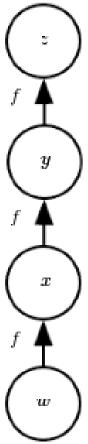
$$\boldsymbol{L}^2 = \frac{1}{2}\sum_i (y - \hat{y})^2 \tag{1}$$

Then, for output node z:

$$\frac{\partial L}{\partial z} = z - \hat{z} \tag{2}$$

Thus…

$$\frac{\partial L}{\partial u} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} \tag{3}$$



[1] Using $\boldsymbol{L}^2$ Norm Loss for simplicity in differentiation

Neural Network Operation
Backpropagation…
Including Loss[1]…
Then, for output node z:
Thus…
[1] Using L² Norm Loss for simplicity in differentiation

1. In equation 4, each partial derivative is taking the derivative of the activation function. For examples with weights and bias, this gradient at the node is then differentiated with respect to the weights and bias.
2. To calculate gradient to x and update parameters, for example, just need to calculate up to $\frac{\partial L}{\partial x}$.

Neural Network Operation

Backpropagation. . .

- Introduced in 1970s, not popular until 1986 [8]

Neural Network Operation
Backpropagation. . .
- Introduced in 1970s, not popular until 1986 [8]

Neural Network Operation

Backpropagation...

Backpropagation...

1. Calculate loss
2. Calculate derivative of loss
3. Starting with the output layer, iterate over network layers.
4. For each node in layer ☞ calculate $\frac{\partial L}{\partial u_i}$
   - In other words, calculate the gradient of the loss w.r.t. node u_i
   - $\frac{\partial L}{\partial u_i}$ is then used to calculate the loss w.r.t. each weight and bias
5. Update each weight: $w = w - \lambda\frac{\partial L}{\partial w}$ [1]
6. Update the bias $b = b - \lambda\frac{\partial L}{\partial b}$

[1] λ is the learning rate, scaling parameter

2024-10-15

Lecture #23
└Backpropagation
└Neural Network Operation

Neural Network Operation
Backpropagation...

Backpropagation...

1. Calculate loss
2. Calculate derivative of loss
3. Starting with the output layer, iterate over network layers.
4. For each node in layer ☞ calculate $\frac{\partial L}{\partial u_i}$
   - In other words, calculate the gradient of the loss w.r.t. node u_i
   - $\frac{\partial L}{\partial u_i}$ is then used to calculate the loss w.r.t. each weight and bias
5. Update each weight: $w = w - \lambda\frac{\partial L}{\partial w}$ [1]
6. Update the bias $b = b - \lambda\frac{\partial L}{\partial b}$

[1] λ is the learning rate, scaling parameter

Learning Optimizations

Gradient Descent	Update after calculating average gradient of entire dataset
Stochastic Gradient Descent(SGD)	Update model parameters after every prediction
Mini-batch SGD	Update model after evaluating a randomly selected batch of samples

Learning Optimizations
Gradient Descent Update after calculating average gradient of entire dataset
Stochastic Gradient Descent(SGD) Update model parameters after every prediction
Mini-batch SGD Update model after evaluating a randomly selected batch of samples

1. Gradient descent is enormously expensive computationally. Has higher accuracy.
2. SGD is resource sipping, but can have erratic convergence behavior.
3. Mini-batch considered the cross-over between GD & SGD

Other Topics of Neural Network Design

- Weight Initialization
  - Simplest would be to set all to zero, but then they learn the same thing
  - Setting all weights to same value means they will all be far from a solution
  - So have heuristics for this [9]
- Learning Rate Selection
  - Too large → fail to converge
  - Too small → converges extremely slowly
  - So strategies to reduce learning rate as training progresses

Other Topics of Neural Network Design
- Weight Initialization
  - Simplest would be to set all to zero, but then they learn the same thing
  - Setting all weights to same value means they will all be far from a solution
  - So have heuristics for this [9]
- Learning Rate Selection
  - Too large → fail to converge
  - Too small → converges extremely slowly
  - So strategies to reduce learning rate as training progresses

Other Topics of Neural Network Design

- Normalizing Inputs
  - Can have inputs on different ranges
  - Binary ☞ [0, 1], discrete
  - Pixel values ☞ 8-bit, 16-bit float?
- Input data dimensionality
  - We have looked at scalar inputs
  - Everything gets more complex for vector inputs
  - But we don't stop there ☞ tensor(s) (3D)

Other Topics of Neural Network Design

- Normalizing Inputs
  - Can have inputs on different ranges
  - Binary ☞ [0, 1], discrete
  - Pixel values ☞ 8-bit, 16-bit float?
- Input data dimensionality
  - We have looked at scalar inputs
  - Everything gets more complex for vector inputs
  - But we don't stop there ☞ tensor(s) (3D)

Bibliography I

[1] *Neural network*, in *Wikipedia*, Page Version ID: 1249531506, Oct. 5, 2024. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Neural_network&oldid=1249531506` (visited on 10/15/2024).

[2] F. Shao and Z. Shen, "How can artificial neural networks approximate the brain?" *Frontiers in Psychology*, vol. 13, p. 970 214, Jan. 9, 2023, ISSN: 1664-1078. DOI: `10.3389/fpsyg.2022.970214`. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9868316/` (visited on 10/15/2024).

Bibliography I

[1] *Neural network*, in *Wikipedia*, Page Version ID: 1249531506, Oct. 5, 2024. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Neural_network&oldid=1249531506` (visited on 10/15/2024).

[2] F. Shao and Z. Shen, "How can artificial neural networks approximate the brain?" *Frontiers in Psychology*, vol. 13, p. 970 214, Jan. 9, 2023, ISSN: 1664-1078. DOI: `10.3389/fpsyg.2022.970214`. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9868316/` (visited on 10/15/2024).

Bibliography II

[3] K. Le, “Can artificial neural networks truly replicate the complexity and functionality of real neurons and…,” Eyes-of-Ai-Journal, (Aug. 2, 2023), [Online]. Available: `https://medium.com/eyes-of-ai-journal/can-artificial-neural-networks-truly-replicate-the-complexity-and-functionality-of-real-neurons-and-291e8746d4bd` (visited on 10/15/2024).

[4] E. U. o. Technology, “New AI method increases the power of artificial neural networks,” (), [Online]. Available: `https://phys.org/news/2018-06-ai-method-power-artificial-neural.html` (visited on 10/15/2024).

[5] “Deep learning,” (), [Online]. Available: `https://www.deeplearningbook.org/` (visited on 10/15/2024).

Bibliography II

[3] K. Le, “Can artificial neural networks truly replicate the complexity and functionality of real neurons and…,” Eyes-of-Ai-Journal, (Aug. 2, 2023), [Online]. Available: `https://medium.com/eyes-of-ai-journal/can-artificial-neural-networks-truly-replicate-the-complexity-and-functionality-of-real-neurons-and-291e8746d4bd` (visited on 10/15/2024).

[4] E. U. o. Technology, “New AI method increases the power of artificial neural networks,” (), [Online]. Available: `https://phys.org/news/2018-06-ai-method-power-artificial-neural.html` (visited on 10/15/2024).

[5] “Deep learning,” (), [Online]. Available: `https://www.deeplearningbook.org/` (visited on 10/15/2024).

Bibliography III

[6] M. A. Nielsen, “Neural networks and deep learning,”, 2015, Publisher: Determination Press. [Online]. Available: `http://neuralnetworksanddeeplearning.com` (visited on 10/15/2024).

[7] J. Brownlee, “Loss and loss functions for training deep learning neural networks,” MachineLearningMastery.com, (Jan. 27, 2019), [Online]. Available: `https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/` (visited on 10/15/2024).

[8] M. A. Nielsen, “Neural networks and deep learning,”, 2015, Publisher: Determination Press. [Online]. Available: `http://neuralnetworksanddeeplearning.com` (visited on 10/15/2024).

Bibliography III

[6] M. A. Nielsen, “Neural networks and deep learning,”, 2015, Publisher: Determination Press. [Online]. Available: http://neuralnetworksanddeeplearning.com (visited on 10/15/2024).

[7] J. Brownlee, “Loss and loss functions for training deep learning neural networks,” MachineLearningMastery.com, (Jan. 27, 2019), [Online]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/ (visited on 10/15/2024).

[8] M. A. Nielsen, “Neural networks and deep learning,”, 2015, Publisher: Determination Press. [Online]. Available: http://neuralnetworksanddeeplearning.com (visited on 10/15/2024).

Bibliography IV

[9] J. Guo, “AI notes: Initializing neural networks,” deeplearning.ai, (), [Online]. Available: https://www.deeplearning.ai/ai-notes/initialization/ (visited on 10/16/2024).

Bibliography IV
[9] J. Guo, “AI notes: Initializing neural networks,” deeplearning.ai, (), [Online]. Available: https://www.deeplearning.ai/ai-notes/initialization/ (visited on 10/16/2024).