CSC520 - Artificial Intelligence

Lecture 22

Dr. Scott N. Gerard

North Carolina State University

Apr 8, 2025

Class Exercise

- Below is the confusion matrix for a model that classifies user activities as fraudulent or benign.

	Predicted Fraudulent	Predicted Benign
Actual Fraudulent	20	50
Actual Benign	10	5000

- Compute the accuracy, precision, recall and F1-score.

Accuracy	Precision	Recall	F1

Class Exercise

- Below is the confusion matrix for a model that classifies user activities as fraudulent or benign.

	Predicted Fraudulent	Predicted Benign
Actual Fraudulent	20	50
Actual Benign	10	5000

- Compute the accuracy, precision, recall and F1-score.

Accuracy	Precision	Recall	F1
$\frac{TP + TN}{TP + FP + FN + TN}$	$\frac{TP}{TP + FP}$	$\frac{TP}{TP + FN}$	$\frac{2 * P * R}{P + R}$

Class Exercise

- Below is the confusion matrix for a model that classifies user activities as fraudulent or benign.

	Predicted Fraudulent	Predicted Benign
Actual Fraudulent	20	50
Actual Benign	10	5000

- Compute the accuracy, precision, recall and F1-score.

Accuracy	Precision	Recall	F1
$\frac{TP+TN}{TP+FP+FN+TN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2*P*R}{P+R}$
98.8%	66.7%	28.8%	40.0%

Class Exercise

- Below is the confusion matrix for a model that classifies user activities as fraudulent or benign.

	Predicted Fraudulent	Predicted Benign
Actual Fraudulent	20	50
Actual Benign	10	5000

- Compute the accuracy, precision, recall and F1-score.

Accuracy	Precision	Recall	F1
$\frac{TP+TN}{TP+FP+FN+TN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2*P*R}{P+R}$
98.8%	66.7%	28.8%	40.0%

- In this case, is accuracy is a good measure of performance?

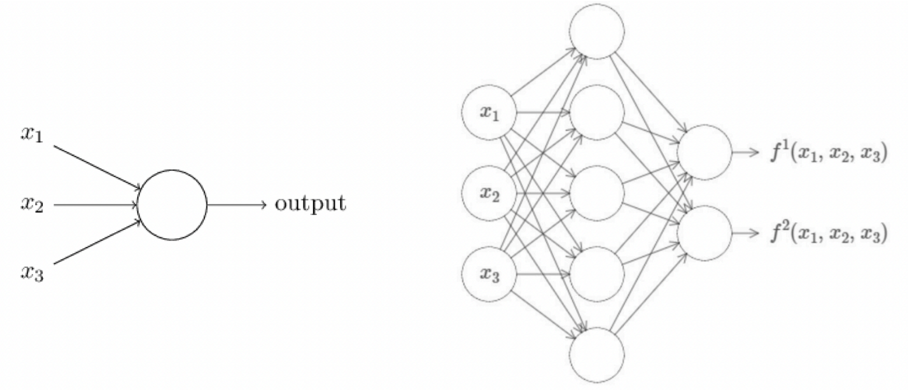
Agenda

- K-Means Clustering
- Naive Bayes Model
- Neural Networks
- Neural Network Computations
- Logistic Regression as Single Neuron
- Training NN and Activation Functions

Neural Networks

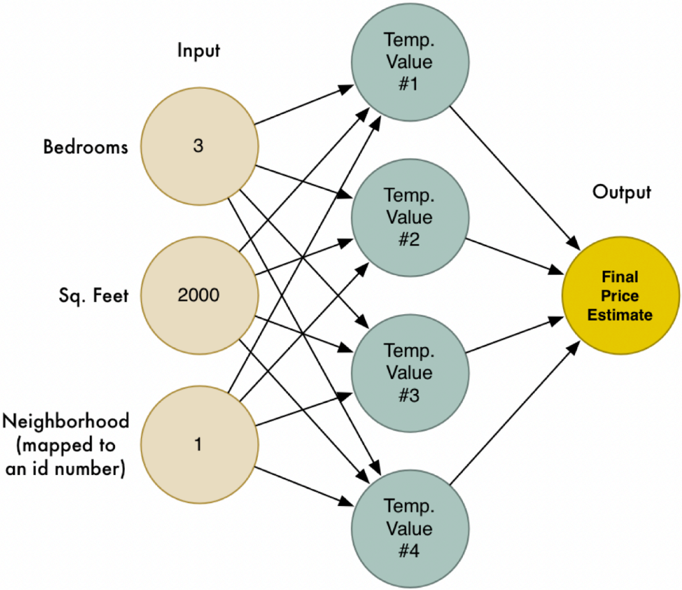
- Inspired by study on how brain works
- Early work on brain models
  - (1943) Warren McCulloch and Walter Pitts defined a basic model of neuron function and a calculus for computing functions
  - (1950) Minsky and Edmonds built the first neural network machine
- Deep learning (neural networks) has become very popular
  - Advancement in computing hardware: math co-processors and GPUs
  - Amount of data available over the internet for training
  - Advancement in the learning algorithms

Neural Networks

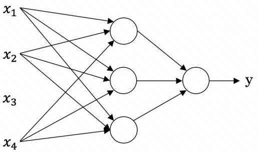


- For any mathematical function you can design a NN that approximates it to a sufficient degree

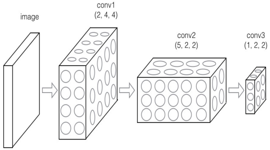
Neural Networks



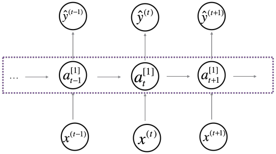
Neural Network Architectures



Fully-connected NN

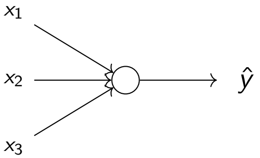


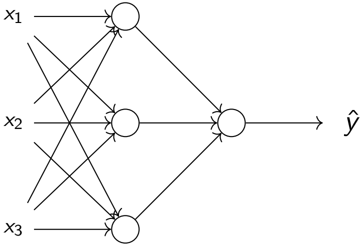
Convolutional NN



Recurrent NN

Neural Network Computation





$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$
$$z^{[1]} = w^{[1]} x + b^{[1]}$$
$$a^{[1]} = \hat{y} = g(z^{[1]})$$

$$z^{[1]} = W^{[1]} x + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$
$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$
$$a^{[2]} = \hat{y} = g(z^{[2]})$$

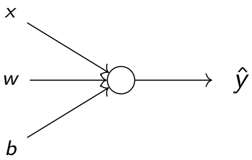
Logistic Regression

- Given data: $(x_1, y_1), (x_2, y_2), \ldots$ where y_i is either 0 or 1
- Hypothesis is a logistic function: $h(x) = \hat{y} = g(wx + b)$, where $g(z) = \dfrac{1}{1 + e^{-z}}$
- Cross-entropy Loss function

$$L(\hat{y}, y) = -\frac{1}{m}\sum_{i=1}^{m}(y_i log \hat{y}_i + (1 - y_i) log(1 - \hat{y}_i))$$

- Goal of the learning is to: $\min_{w,b} L(\hat{y}, y)$
- Use gradient descent like in linear regression

Logistic Regression as Single Neuron



$$z = wx + b$$

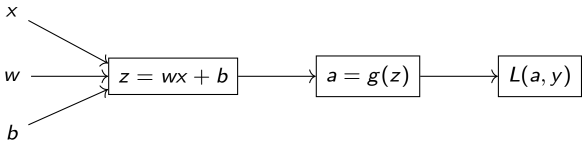
$$\hat{y} = a = g(z)$$

- Gradient descent

Start with initial values for w,b
while not converged **do**

$$w = w - \alpha \frac{\partial}{\partial w} L(\hat{y}, y)$$
$$b = b - \alpha \frac{\partial}{\partial b} L(\hat{y}, y)$$

Logistic Regression as Single Neuron



$$
\begin{aligned}
\frac{\partial L}{\partial w} &= \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w} \\
&= \frac{a-y}{a(1-a)} * a(1-a) * x \\
&= x(a-y) \\
\frac{\partial L}{\partial b} &= \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial b} \\
&= \frac{a-y}{a(1-a)} * a(1-a) \\
&= a-y
\end{aligned}
$$

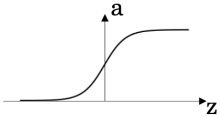
$$
\begin{aligned}
\frac{\partial L}{\partial a} &= \frac{\partial}{\partial a} - (y \log a + (1-y)\log(1-a)) \\
&= \frac{-y}{a} + \frac{1-y}{1-a} = \frac{a-y}{a(1-a)} \\
\frac{\partial a}{\partial z} &= a(1-a) \\
\frac{\partial z}{\partial w} &= x \\
\frac{\partial z}{\partial b} &= 1
\end{aligned}
$$

Training Neural Networks

- Gradient descent is used
  - Forward pass to compute activations
  - Backward pass to compute gradients
- Hyperparameters need to be manually specified and tuned
  - Activation function ($g(z)$)
  - Number of hidden layers
  - Number of units per layer
  - Learning rate (α)
  - ...

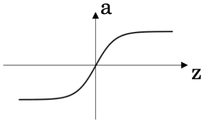
Activation Functions

Sigmoid: $a = \dfrac{1}{1 + e^{-z}}$



- Advantages: Derivable at every point, output is between 0 to 1
- Disadvantages: Vanishing gradient problem
- Typically used as output of binary classification

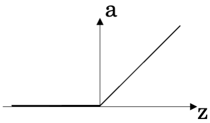
tanh: $a = \dfrac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})}$



- Advantages: Derivable at every point, output is between -1 to $+1$, centered on 0
- Disadvantages: Vanishing gradient problem
- Typically used in the hidden layers

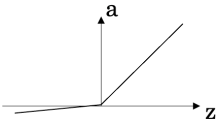
Activation Functions

ReLU: $a = \max(0, z)$



- Advantages: Addresses vanishing gradient problem, computationally effective
- Disadvantages: Dying ReLU issue (discards negative values)
- Typically used in the hidden layers

Leaky ReLU:
$a = \max(\alpha z, z)$, where α is typically 0.01



- Advantages: Same as ReLU but addresses dying ReLU issue
- Typically used in the hidden layers

Variants of Gradient Descent

- Batch: Forward and backward propagation on the entire training dataset
  - Slow if training dataset is huge
- Mini-batch: Forward and backward propagation on a smaller batch of training dataset
  - Training dataset is divided into smaller batches
  - Forward and backward propagation on each of the smaller batch
- Stochastic: Forward and backward propagation on one example at a time

Class Exercise

Naive Bayes Model:

$$P(C|f_1,\ldots,f_n) = \alpha P(f_1,\ldots,f_n|C)P(C)$$
$$= \alpha \prod_i P(f_i|C)P(C)$$

Age	Prescription	Astigmatism	TearRate	Lenses
Young	Myope	No	Reduced	Noncontact
Young	Myope	No	Normal	Softcontact
Young	Myope	Yes	Reduced	Noncontact
Young	Myope	Yes	Normal	Hardcontact
Young	Hypermetrope	No	Reduced	Noncontact
Young	Hypermetrope	No	Normal	Softcontact
Young	Hypermetrope	Yes	Reduced	Noncontact
Young	Hypermetrope	Yes	Normal	Hardcontact
Prepresbyopic	Myope	No	Reduced	Noncontact
Prepresbyopic	Myope	No	Normal	Softcontact

Compute the following probability:

$P(Softcontact|Prespresbyopic, Hypermetrope, No, Normal) =$