DATA 586: Advanced Machine Learning

2023W2

Shan Du

Course Information

- **Instructor**: Dr. Shan Du FIP 324
  shan.du@ubc.ca
- **Department**: Computer Science
- **Class Time**: Tuesday and Thursday 11:00AM – 12:30PM
- **Location**: EME 1153
- **Office Hour**: Friday 11:00AM – 12:00PM on Canvas
- **Course Website:**

  https://canvas.ubc.ca/courses/133408

Course Information

- **Teaching Assistant**:

  Amanat Ullah, amanat7@mail.ubc.ca

Course Information

- **Course Description**:
  - Neural networks, backpropagation, deep learning. Restricted to students in the MDS program.
  - Prerequisite: DATA 580.

Course Information

- **Learning Materials:**
  - Lecture Notes and Tutorials (available electronically)
  - Recommended Textbooks :
    - An Introduction to Statistical Learning with Applications in Python (Chapter 10), Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor, 2023: https://www.statlearning.com/
    - Deep Learning (Chapter 5-10), Ian Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press, 2016: https://www.deeplearningbook.org
    - Dive into Deep Learning: Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, Cambridge University Press, 2023: https://d2l.ai/

- **Programming Language:** Python

Course Information

Learning Outcomes:

Upon successful completion of this course, students will be able to:

- Demonstrate understanding of the basic concepts of deep neural networks and their applications
- Develop practical skills for designing and training neural networks
- Be comfortable with tools and techniques required in deep learning
- Know how to use a neural network
- Explore the parameters for neural networks

Course Information

Grading:

Programming Assignments: 40%

Project: 60%

Regulations

- Students MUST achieve a passing grade in overall in order to pass the course.
- The course website contains the most up-to-date information and important dates for main events such as assignments and project due dates. Students need to check regularly.
- Attendance is mandatory in lectures and labs.
- **The use of artificial intelligence (AI) assistance for any assessed portions of this course is not permitted.**
- If you feel any mark was unfair or incorrectly recorded, ensure that I am aware of the problem before the last week of classes.

Regulations

- **Late Penalty:** Late assignments and report will be deducted 10% per day up to 3 days (after which they will receive 0 marks).

- **Plagiarism:** is forbidden (0 mark).

Course Contents and Schedule

Week	Contents
1	Introduction to Neural Network, Single Layer NN, Multiple Layer NN (MLP)
2	Convolutional Neural Network (CNN), Recurrent Neural Network (RNN)
3	Optimization, Regularization
4	Modern CNN, Modern RNN
5	Projects and Presentations

Acknowledgment

- The course materials are based on:
  - The recommended textbooks/references
  - Some online resources and similar courses offered in other top-ranked universities
  - Some published papers and datasets

Learning Algorithms

- A machine learning algorithm is an algorithm that is able to learn from data.
- But what do we mean by learning? - “A computer program is said to learn from **experience *E*** with respect to some class of **tasks *T*** and **performance measure *P***, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.”

The Task, *T*

- Machine learning tasks are usually described in terms of how the machine learning system should process an **example**.
- An example is a collection of **features** that have been quantitatively measured from some object or event that we want the machine learning system to process.

The Task, *T*

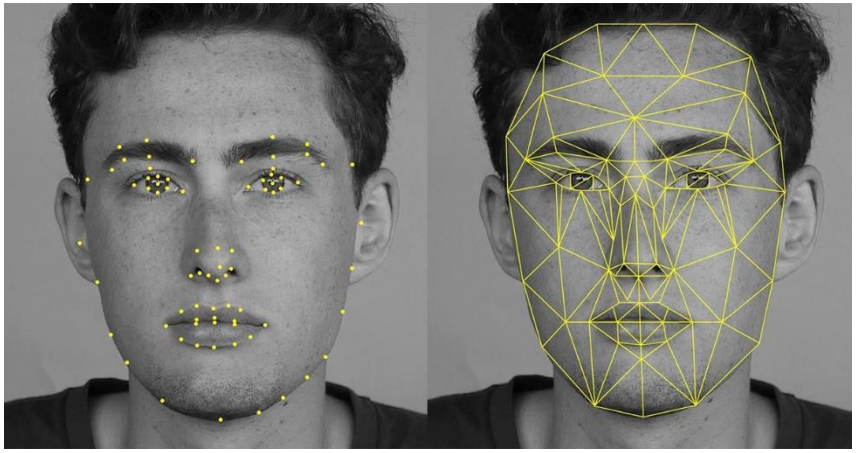
- We typically represent an example as a vector $x \in R^n$ where each entry x_i of the vector is a feature. For example, the features of an image are usually the values of the pixels in the image.
- Some of the most common machine learning tasks include the following:

The Task, *T*

- Classification: Specify which of *k* categories the input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f: R^n \rightarrow \{1, \ldots, k\}$. When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y.

The Task, *T*

– Classification:



The Task, *T*

- Classification with missing inputs: Classification becomes more when some of the inputs may be missing. Rather than providing a single classification function, the learning algorithm must learn a set of functions. Each function corresponds to classifying x with a different subset of its inputs missing. This kind of situation arises frequently in medical diagnosis, because many kinds of medical tests are expensive or invasive. One way to efficiently define such a large set of functions is to learn a probability distribution over all of the relevant variables, then solve the classification task by marginalizing out the missing variables. With n input variables, we can now obtain all 2^n different classification functions needed for each possible set of missing inputs, but we only need to learn a single function describing the joint probability distribution.

The Task, *T*

- Regression: Predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function $f: R^n \rightarrow R$. This type of task is similar to classification, except that the format of output is different.

The Task, *T*

- Transcription: Observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form.
  - For example, in optical character recognition, the computer program is shown a photograph containing an image of text and is asked to return this text in the form of a sequence of characters (e.g., in ASCII or Unicode format).
  - Another example is speech recognition, where the computer program is provided an audio waveform and emits a sequence of characters or word ID codes describing the words that were spoken in the audio recording.

The Task, *T*

- Machine translation: The input consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language. This is commonly applied to natural languages, such as translating from English to French.

The Task, *T*

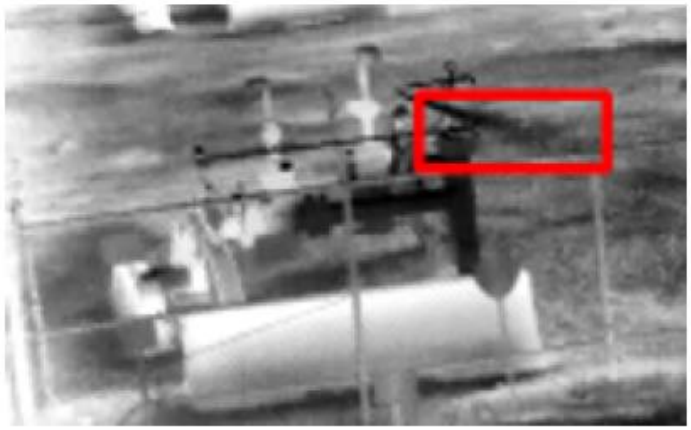
- Structured output: Structured output tasks involve any task where the output is a vector (or other data structure containing multiple values) with important relationships between the different elements. One example is parsing—mapping a natural language sentence into a tree that describes its grammatical structure and tagging nodes of the trees as being verbs, nouns, or adverbs, and so on. Another example is pixel-wise segmentation of images, where the computer program assigns every pixel in an image to a specific category. For example, deep learning can be used to annotate the locations of roads in aerial photographs.

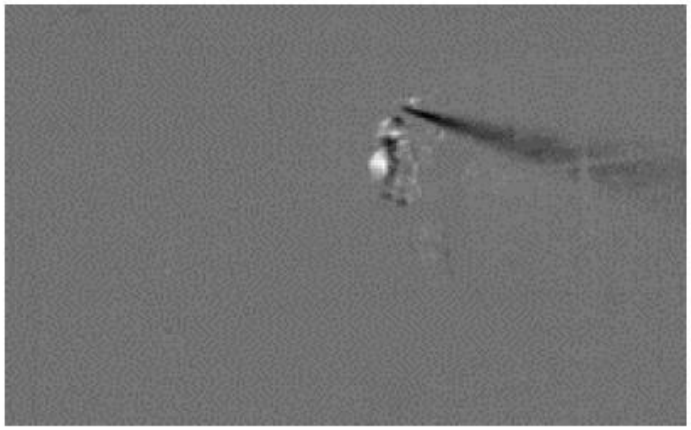
The Task, *T*

- Anomaly detection: Sift through a set of events or objects, and flags some of them as being unusual or atypical. An example of an anomaly detection task is credit card fraud detection. By modeling your purchasing habits, a credit card company can detect misuse of your cards. If a thief steals your credit card or credit card information, the thief's purchases will often come from a different probability distribution over purchase types than your own. The credit card company can prevent fraud by placing a hold on an account as soon as that card has been used for an uncharacteristic purchase.

The Task, *T*

– Anomaly detection:



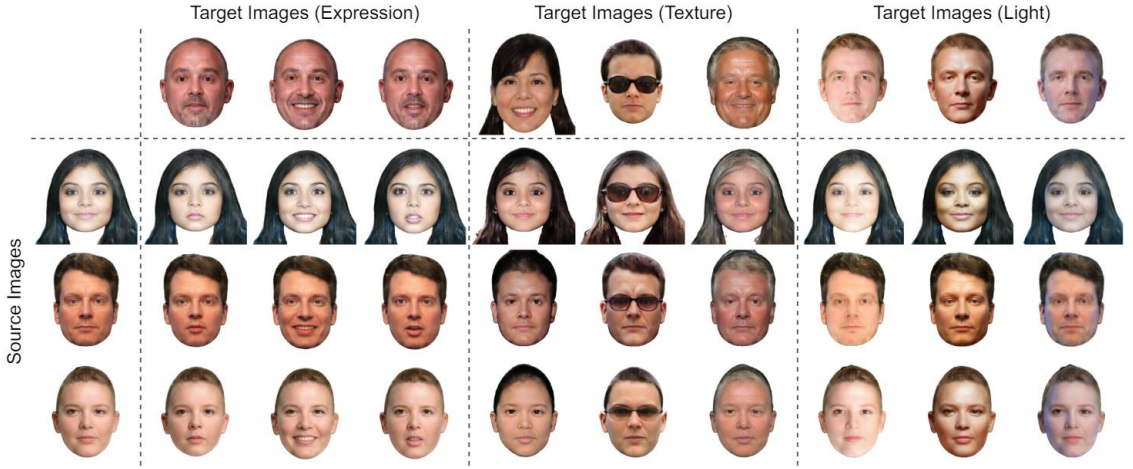


The Task, *T*

- Synthesis and sampling: Generate new examples that are similar to those in the training data. Synthesis and sampling via machine learning can be useful for media applications where it can be expensive or boring for an artist to generate large volumes of content by hand.

The Task, *T*

- Synthesis and sampling:



The Task, *T*

- Synthesis and sampling:

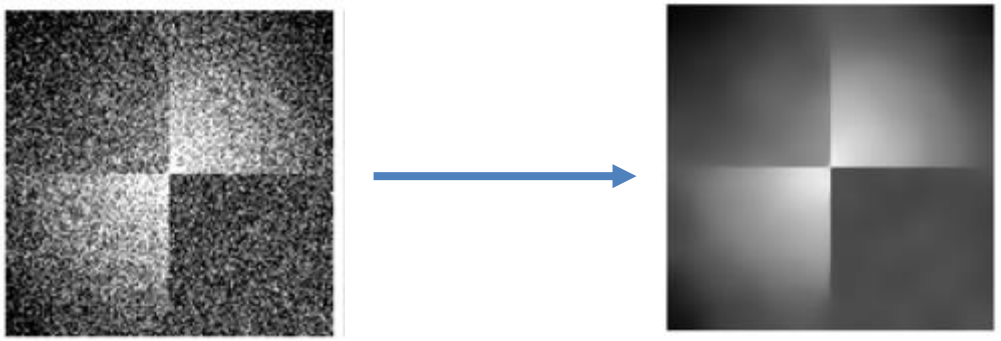
- Creating video from text https://openai.com/sora

The Task, *T*

– Imputation of missing values: In this type of task, the machine learning algorithm is given a new example $x \in R^n$, but with some entries x_i of $\mathbf{x}$ missing. The algorithm must provide a prediction of the values of the missing entries.

The Task, *T*

– Denoising



– Super-resolution

The Task, *T*

– Density estimation or probability mass function estimation: Learn a function $p_{model}: R \rightarrow R^n$, where $p_{model}(\mathbf{x})$ can be interpreted as a probability density function (if $\mathbf{x}$ is continuous) or a probability mass function (if $\mathbf{x}$ is discrete) on the space that the examples were drawn from. The algorithm needs to learn the structure of the data it has seen. It must know where examples cluster tightly and where they are unlikely to occur.

The Performance Measure, *P*

- In order to evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance.
- Usually this performance measure *P* is specific to the task *T* being carried out by the system.
- For tasks such as classification, classification with missing inputs, and transcription, we often measure the **accuracy** of the model. Accuracy is just the proportion of examples for which the model produces the correct output.

The Performance Measure, *P*

- We can also obtain equivalent information by measuring the *error rate*, the proportion of examples for which the model produces an incorrect output.
- We often refer to the error rate as the expected 0-1 loss. The 0-1 loss on a particular example is 0 if it is correctly classified and 1 if it is not.

The Performance Measure, *P*

- For tasks such as density estimation, it does not make sense to measure accuracy, error rate, or any other kind of 0-1 loss.
- Instead, we must use a different performance metric that gives the model a continuous-valued score for each example.
- The most common approach is to report the average log-probability the model assigns to some examples.

The Performance Measure, *P*

- Usually, we are interested in how well the machine learning algorithm performs on data that it has not seen before.
- We therefore evaluate these performance measures using a *test set* of data that is separate from the data used for training the machine learning system.

The Experience, *E*

- Machine learning algorithms can be broadly categorized as u*nsupervised* or *supervised* by what kind of experience they are allowed to have during the learning process.
- The experience can be given by an entire *dataset*. A dataset is a collection of many *examples*. Sometimes we will also call examples *data points*.

Unsupervised learning algorithms

- Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.
- In the context of deep learning, we usually want to learn the entire probability distribution that generated a dataset, whether explicitly as in density estimation or implicitly for tasks like synthesis or denoising. Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the dataset into clusters of similar examples.

Supervised learning algorithms

- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a *label* or *target*.

The Experience, *E*

- Roughly speaking, unsupervised learning involves observing several examples of a random vector x and attempting to implicitly or explicitly learn the probability distribution $p(x)$, or some interesting properties of that distribution.
- Meanwhile supervised learning involves observing several examples of a random vector x and an associated value or vector y, and learning to predict y from x, usually by estimating $p(y|x)$.

The Experience, *E*

- The term supervised learning originates from the view of the target y being provided by an instructor or teacher who shows the machine learning system what to do.
- In unsupervised learning, there is no instructor or teacher, and the algorithm must learn to make sense of the data without this guide.

The Experience, *E*

- Traditionally, people refer to regression, classification and structured output problems as supervised learning.
- Density estimation in support of other tasks is usually considered unsupervised learning.
- Other variants of the learning paradigm are possible. For example, in semi-supervised learning, some examples include a supervision target but others do not. In multi-instance learning, an entire collection of examples is labeled as containing or not containing an example of a class, but the individual members of the collection are not labeled.

The Experience, *E*

- Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.
- One common way of describing a dataset is with a **design matrix**. A design matrix is a matrix containing a different example in each row. Each column of the matrix corresponds to a different feature.

History of Neural networks

- Neural networks rose to fame in the late 1980s.
- Then along came SVMs, boosting, and random forests, and neural networks fell somewhat from favor.
  - Neural networks required a lot of tinkering, while the new methods were more automatic.
  - New methods outperformed poorly-trained neural networks.

History of Neural networks

- Neural networks resurfaced after 2010 with the new name deep learning.
  - Ever-larger training datasets
  - High performance computing devices

Single Layer Neural Networks

- A neural network takes an input vector of p variables $X = (X_1, X_2, \ldots, X_p)$ and builds a nonlinear function $f(X)$ to predict the response Y.

Single Layer Feed-Forward Neural Networks

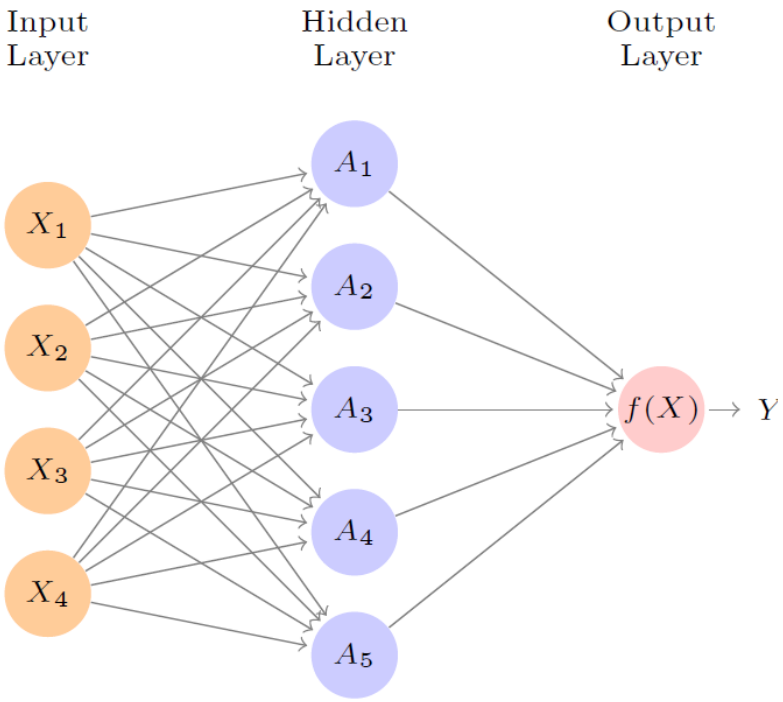


FIGURE 10.1. *Neural network with a single hidden layer. The hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs $X_1, X_2, \ldots, X_p$. Hence these A_k are not directly observed. The functions $h_k(\cdot)$ are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_k as inputs, resulting in a function $f(X)$.*

Single Layer Neural Networks

- The p variables (predictors, features) $X_1, X_2, \ldots, X_p$ make up the units in the *input layer*.
- The arrows indicate that each of the inputs from the input layer feeds into each of the K hidden input layer units (we get to pick K).
- The neural network model has the form:

Single Layer Neural Networks

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

$$= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

It is built up in two steps.

Single Layer Neural Networks

- First, the K *activations* $A_k, k = 1, \ldots, K$, in the hidden layer are computed as functions of the input features $X_1, X_2, \ldots, X_p$,

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j),$$

where $g(z)$ is a nonlinear *activation function* that is specified in advance.

We can think of each A_k as a different transformation $h_k(X)$ of the original features.

Single Layer Neural Networks

- Then the K activations from the hidden layer then feed into the output layer, resulting in

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k \,,$$

A linear regression model in the K activations.

All the parameters $\beta_0, \ldots, \beta_K$ and $w_{10}, \ldots, w_{Kp}$ need to be estimated from data.

Activation Functions

- In the early instances of neural networks, the *sigmoid* activation function was favored,

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}},$$

which is to convert a linear function into probabilities between zero and one.

- The preferred choice in modern neural networks is the *ReLU* (*rectified linear unit*) activation function, which takes the form

Activation Functions

$$g(z) = \begin{cases} 0 & if\ z < 0 \\ z & otherwise \end{cases}$$

A ReLU activation can be computed and stored more efficiently than a sigmoid activation.

Although it thresholds at zero, because we apply it to a linear function, the constant term w_{k0} will shift this inflection point.

Activation Functions

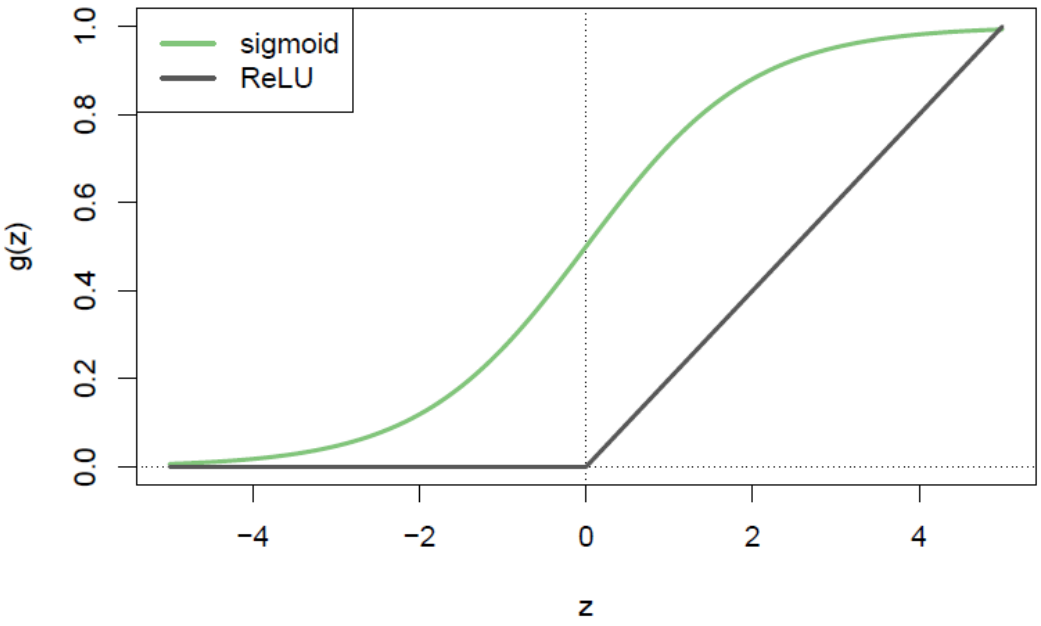


FIGURE 10.2. *Activation functions.* *The piecewise-linear* `ReLU` *function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison.*

Single Layer Neural Networks

- The model depicted in Figure 10.1 derives five new features by computing five different linear combinations of X, and then squashes each through an activation function $g(\cdot)$ to transform it. The final model is linear in these derived variables.

Single Layer Neural Networks

- The name $neural\ network$ originally derived from thinking of these hidden units as analogous to neurons in the brain — values of the activations $A_k = h_k(X)$ close to 1 are *firing*, while those close to 0 are *silent* (using the sigmoid activation function).
- The nonlinearity in the activation function $g(\cdot)$ is essential, since without it the model $f(X)$ would collapse into a simple linear model in $X_1, X_2, \ldots, X_p$.

Single Layer Neural Networks

- Moreover, having a nonlinear activation function allows the model to capture complex nonlinearities and interaction effects.

An Example

- Consider a very simple example with $p = 2$ input variables $X = (X_1, X_2)$, and $K = 2$ hidden units $h_1(X)$ and $h_2(X)$ with $g(z) = z^2$.
- We specify the other parameters as

$$\beta_0 = 0, \beta_1 = \frac{1}{4}, \beta_2 = -\frac{1}{4},$$
$$w_{10} = 0, w_{11} = 1, w_{12} = 1,$$
$$w_{20} = 0, w_{21} = 1, w_{22} = -1.$$

An Example

- Therefore,

$$h_1(X) = (0 + X_1 + X_2)^2$$
$$h_2(X) = (0 + X_1 - X_2)^2$$

Then we get

$$f(X)$$
$$= 0 + \frac{1}{4} \cdot (0 + X_1 + X_2)^2 - \frac{1}{4} \cdot (0 + X_1 - X_2)^2$$
$$= \frac{1}{4}[(X_1 + X_2)^2 - (X_1 - X_2)^2] = X_1 X_2$$

Single Layer Neural Networks

- Fitting a neural network requires estimating the unknown parameters $\beta_0, \ldots, \beta_K$ and $w_{10}, \ldots, w_{Kp}$.
- For a quantitative response, typically squared-error loss is used, so that the parameters are chosen to minimize

$$\sum_{i=1}^{n} (y_i - f(x_i))^2$$