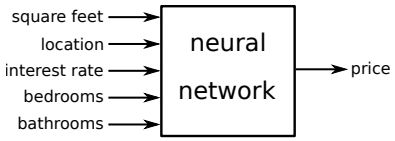
1 Neural Networks as Functions

1 Definition (Neural Network)
A neural network is a function that maps inputs to outputs.

---

2 Example (Housing Prices Neural Network)
Real estate agents want to know what price a house will sell for. A neural network can predict the selling price.



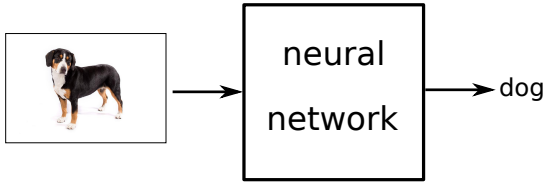
The inputs are called **features** and are application specific. If the output is continuous, the network is a **regression network**. If the output is discrete, then the network is a **classification network**. The function that maps features to prices is learned from a large number of training examples.

---

3 Example (Feature Engineering)
If we had thousands of features (inputs) direct programming of the housing prices neural network would be difficult and require **feature engineering** which is a process of combining input features to construct more abstract and useful features. Deep learning automates feature engineering, i.e. the housing prices neural network can be trained **end-to-end** using the raw data.

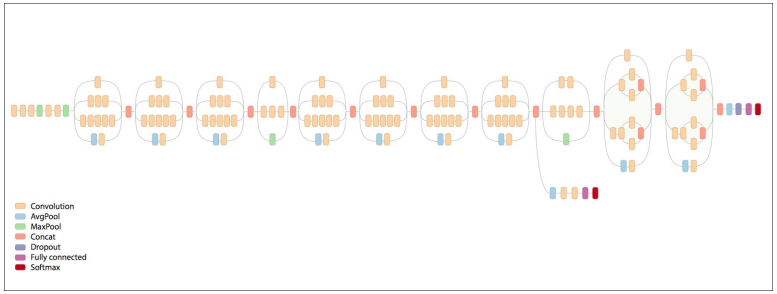
---

4 Example (Image Classifier)



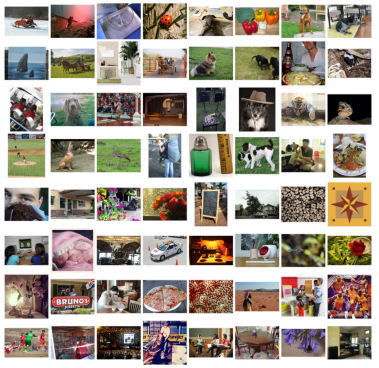
---

5 Example (InceptionV3)



---

6 Example (ImageNet)
ImageNet is a dataset of over 14 million images from 20,000 categories.



---

7 Example (Jupyter Notebook Image Classifier)
Implement a Jupyter notebook image classifier using the InceptionV3 neural network trained on ImageNet.

---

8 Example (Trainable Parameters)
How many trainable parameters does InceptionV3 have?

Hint Use Kera's `.summary()` method.

---

9 Definition (Simple Bias Network)
The simplest possible neural network has no input and has a contant output called the **bias** of the network. The simple bias network has a single trainable parameter called b for bias.

---

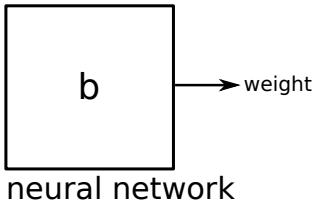
10 Example (Weight Prediction)
Assume we are interested in designing playground equipment. We need to predict the weight of a child. Assume children are defined to be humans who are 18 years old or younger. Below is a list of the weights (in pounds) of 10 randomly selected children. What weight should we predict for a child?[1]

weights
128
123
129
143
132
142
112
118
108
119

---

11 Example (Simple Bias Network)
Lets train a neural network to predict a child's weight. This network will be unusual because we have no features available to use as inputs to the network. It has a single parameter b, the bias of the network.

[1] Data source: SOCR Data



We will learn an appropriate value for b from the data provided. First we need to define a **loss function** that tells us how good the network is at predicting the weight of a child.

---

12 Definition (L^1 Loss Function)
Assume n equals the number of available outputs from our dataset. The L^1 loss function for a neural network is:

$$\frac{1}{n}\sum_{i=1}^{n}|\text{predicted output} - \text{target output}|$$

---

13 Definition (L^2 Loss Function)
Assume n equals the number of available outputs from our dataset. The L^2 loss function (Mean Square Error or MSE) for a neural network is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\text{predicted output} - \text{target output})^2$$

---

14 Example (Simplest Network, L^2 Version)
Repeat the previous Lesson except use an L^2 loss function in place of the L^1 loss function.

In general, training a neural network with an L^2 loss will give a different network than training with an L^1 loss. The L^2 loss is easier to use. Note that the L^2 loss function is smoother than the L^1 loss function.

15 Example (Optimal Bias)
Show that for a simple bias network, the bias, b, that minimizes the MSE (L^2 loss) is equal to the mean of the target values, i.e. the optimal b is equal to $\bar{y}$.

Hint: Set the derivative of the MSE loss function equal to zero.

---

16 Definition (Root Mean Square Error)
The square root of MSE is called the Root Mean Square Error or RMSE.

$$RMSE = \sqrt{MSE}$$

---

17 Definition (Variance and Standard Deviation)
The **sample variance**, s^2, of a set of n numbers, $y_1, y_2, \cdots, y_n$, is given by:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

The **sample standard deviation** is s.

---

18 Definition (Baseline Performance)
In deep learning it is often important to establish a **baseline performance** to compare to. One of the simplest baseline performances to compute for regression networks is the minimum RMSE of a simple bias network. If you are unable to do better than this simple baseline, you have serious issues to deal with. We show below that the standard deviation of the target values is a good estimate of the optimal performance of a simple bias network.

---

19 Example (Estimating Performance of Simple Bias Network)
Show that the standard deviation of the target values is a good estimate for minimum RMSE of a simple bias network.

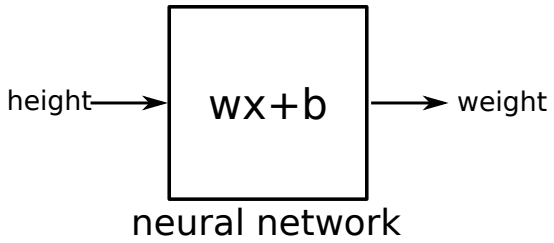
Before training a neural network, its a good idea to establish a baseline for prediction accuracy. If your neural network can't beat the simple bias network, you have a poor result.

2 Simple Linear Regression

20 Example (Simple Linear Regression)
What is the weight of a child who is 70 inches tall? Below is a table of the heights (in inches) and the weights (in pounds) of 10 randomly selected children.[2]

height	weight
67	128
67	123
72	129
69	143
69	132
70	142
67	112
67	118
66	108
68	119

Unlike the previous examples, we now have a feature, the height of a child, that we can use to predict the weight of the child. We will train the neural network shown below.



The variable x represents the value of a feature (in this case height) and w and b are the parameters of the network that must be learned from

[2] Data source: SOCR Data

the data.[3]

---

21 Definition (Dot Product)
Let $\mathbf{u} = \big(u_1, u_2, \ldots, u_n\big)^\top$ and $\mathbf{v} = \big(v_1, v_2, \ldots, v_n\big)^\top$ be two vectors with n components. The dot product of $\mathbf{u}$ and $\mathbf{v}$ is given by

$$\mathbf{u} \circ \mathbf{v} = \mathbf{u}^\top \mathbf{v} = u_1 v_1 + v_2 v_2 \ldots + u_n v_n.$$

---

22 Example (Dot Product)
Let $\mathbf{u} = \big(5, 3, 1\big)^\top$ and $\mathbf{v} = \big(-1, 2, 1\big)^\top$. Compute $\mathbf{u} \circ \mathbf{v} = \mathbf{u}^\top \mathbf{v}$.

---

23 Definition (Matrix Multiplication)
Assume A has m rows and n columns and B has n rows and r columns. Let C be the matrix product of A and B.

$$A_{m\times n} B_{n\times r} = C_{m\times r}.$$

1. The inner dimensions must match for the matrices to be compatible for multiplication.
2. The resulting matrix, C, will have the outer dimensions.
3. Each i, j element in the matrix C is a dot product of row i of A with column j of B.

---

24 Example (Matrix Multiplication)

$$\begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 1 \end{pmatrix}_{3\times 2} \begin{pmatrix} 3 & -1 \\ 2 & 0 \end{pmatrix}_{2\times 2} = \begin{pmatrix} 3 & -1 \\ 4 & 0 \\ 5 & -1 \end{pmatrix}_{3\times 2}$$

---

[3] w should not be confused for the weight of the child. The weight of the child is the output of the network, y.

25 Example (Matrix Form of Simple Linear Regression)
Show that the simple linear regression network can be expressed in matrix form as the function

$$\mathbf{f}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \hat{\mathbf{y}}$$

where

$$\mathbf{X} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \ \mathbf{w} = \begin{pmatrix} w \\ b \end{pmatrix}, \ \hat{\mathbf{y}} = \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_n} \end{pmatrix}$$

Note: The hat is used to indicate that $\hat{\mathbf{y}}$ is predicted value for $\mathbf{y}$.

---

26 Definition (Gradient Vector)
The gradient vector of a scalar-valued function $f(\mathbf{w})$ is denoted by $\frac{df}{d\mathbf{w}}$ and is given by

$$\frac{df}{d\mathbf{w}} = \Big(\tfrac{\partial f}{\partial w_1}, \quad \tfrac{\partial f}{\partial w_2}, \quad \ldots, \quad \tfrac{\partial f}{\partial w_n}\Big)^\top.$$

---

27 Example (Gradient Vector)
Compute the gradient vector of the function

$$f(\mathbf{w}) = f(w_1, w_2, w_3) = w_1^2 + w_1 w_2 + 2 w_2 w_3^2.$$

---

28 Example (Normal Equations)
Show that the parameter values, $\mathbf{w}$, that minimizes the L^2 loss function $MSE(\mathbf{w})$ where

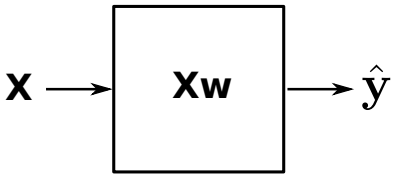
$$MSE(\mathbf{w}) = \frac{1}{n}(\hat{\mathbf{y}} - \mathbf{y})^\top (\hat{\mathbf{y}} - \mathbf{y})$$

is given by the solution to the **normal equations**

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}.$$

---

29 Example (Linear Regression With More Than One Feature)
Linear regression networks with d features



can be expressed in matrix form as the function

$$\mathbf{f}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \hat{\mathbf{y}}$$

where

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} & 1 \end{pmatrix}, \ \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{pmatrix}, \ \hat{\mathbf{y}} = \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_n} \end{pmatrix}$$

Columns j, $j = 1, 2, \dots, d$, of the data matrix $\mathbf{X}$ contain the n values of the jth feature. w_j is the network weight of the jth feature. The last column corresponds to the bias and can be interpreted as a special feature whose value is always equal to 1.

---

30 Example (Linear Regression for Red Wine)

(a) Load the data set `wine_quality_red.csv` into a dataframe. Assume wine quality is the feature to be predicted. How many input features does the data set contain? How many data points?

(b) Determine the feature weights and bias that minimize MSE. Hint: Add a column of 1's to the data matrix $\mathbf{X}$ to represent the bias. Then solve the normal equations to obtain the optimal weights and bias.

(c) Compare RMSE of the linear regression network with the RMSE of a simple bias network.

---

31 Example (Red Wine Training vs Testing RMSE)
Shuffle then split the red wine dataset into two datasets, 80% for training and 20% for testing. Train on the training set and compute RMSE for training and testing sets. Which do you expect to be larger training RMSE or testing RMSE?

---

32 Definition (One-Hot-Encoding)
One-Hot-Encoding is a procedure that converts categorical data into numerical data that can be used as input to machine learning algorithms. The converted data can be interpreted as a probability distribution.

---

33 Example (One-Hot-Encoding Abalone)
Data Source[4]

Abalone is a type of snail. How accurately can the age of abalone be predicted from physical measurements? Data on the features listed in the table below is given in the file `Abalone.csv`.

feature	measurement	description
sex	M, F, I	male, female or infant
length	mm	longest shell measurement
diameter	mm	perpendicular to length
height	mm	with meat in shell
weight_whole	gms	whole abalone
weight_shucked	gms	weight of meat
weight_viscera	gms	gut weight (after bleeding)
weight_shell	gms	after being dried
rings		+1.5 gives the age in years

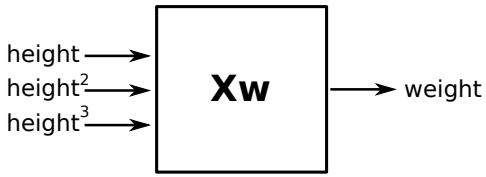
---

34 Definition (Feature Engineering)
Feature engineering is a process of creating new features from old ones in an effort to improve the performance of a machine learning al-

[4] Marine Resources Division, Marine Research Laboratories-Taroona, Department of Primary Industry and Fisheries, GPO Box 619F, Hobart, Tasmania 7001, Australia

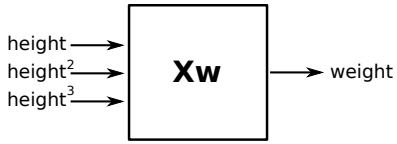
gorithm.

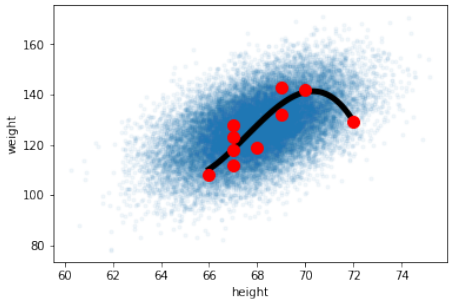
35 Example (Feature Engineering Height vs Weight)
In order to improved the prediction accuracy of weight using height, we can include two new features: height^2 and height^3.



36 Example (Height vs Weight Feature Engineering)

height	weight
67	128
67	123
72	129
69	143
69	132
70	142
67	112
67	118
66	108
68	119





37 Example (Hyper-Parameters)
Consider the problem of predicting a child's weight, y, from the child's height, x. Assume we have created a series of engineered feature, x^k, $k = 2, 3, \ldots, d$ so that

$$\hat{y} = b + w_1 x + w_2 x^2 + \cdots + w_d x^d.$$

The parameter d is called a **hyper-parameter**. The parameters $b, w_1, w_2, \ldots, w_d$ are selected to minimize training MSE by solving the normal equations. The hyper-parameter d, on the other hand, is selected to minimize the test MSE. In the *Lesson (Feature Engineering)*, the optimal value for the hyper-parameter d is $d = 1$.

38 Example (Stock Prices)
Can past stock prices be used to predict future stock prices?

The file `AdjustedClosingPrices.csv`[5] contains data on 467 U.S. stocks over the course of 4173 days. Train a linear regression neural network that uses yesterday's and today's stock prices to predict tomorrow's stock prices. Train the network using stock price data for IBM. Normalize the data by computing log price ratios as shown below:

$$\text{log price ratio} = \ln\left(\frac{\text{price today}}{\text{price yesterday}}\right).$$

Evaluate the accuracy of your trained neural network.

39 Example (Iris Dataset)
R. A. Fisher's iris data set is one of the oldest, most used examples in pattern recognition. The data set contains three class (50 instances each) of the iris plant. Each class corresponds to one of three plant species: Setosa, Versicolour and Virginica. Plant attributes consist of four measurements (in centimeters) of plant leaves: sepal length and width and petal length and width.

[5] Courtesy Dylan Vener. The data was downloaded from Yahoo Finance.



iris plant setosa versicolor virginica

Use the dataset `Iris-cleaned.csv` to predict plant species from sepal length and sepal width measurements.

One of the problems of using linear regression to solve classification problems like the Iris problem described above is that the predictions, $\hat{\mathbf{Y}}$, of linear regression are not true probability distributions like one-hot-encoding. We can fix this problem by using a special function called the **softmax function** that converts the output, $\hat{\mathbf{Y}}$, of linear regression into a genuine probability distribution, $\hat{\mathbf{P}}$. The resulting classifier is called **logistic regression**. Unfortunately, logistic regression, is nonlinear, so the normal equations can no long be applied.

40 Definition (Linearity)
A function $\mathbf{y} = f(\mathbf{x})$ with n inputs and m outputs is is linear if there exits an $m \times n$ matrix $\mathbf{A}$ such that $f(\mathbf{x}) = \mathbf{Ax}$.

Note that a characteristic of linear function is that $f(\mathbf{0}) = \mathbf{0}$.

---

41 Definition (Training vs Inference)
Linear regression networks have two functional forms:

$$\begin{array}{rl} \text{Training} & \hat{\mathbf{y}} = f_{\mathbf{X}}(\mathbf{w}) = \mathbf{Xw} \\ \text{Inference} & \hat{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{X}) = \mathbf{Xw} \end{array}$$

During training, the data matrix $\mathbf{X}$, is constant while during inference, the weights $\mathbf{w}$ are constant.

---

42 Definition (Linear Combination)
Linear regression can be interpreted as a **linear combination** of features plus a bias.

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{Xw} \\ &= \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_d & \mathbf{1} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \end{pmatrix} \\ &= w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + \cdots + w_d\mathbf{x}_d + b\mathbf{1} \end{aligned}$$

The predictions for the target, $\hat{\mathbf{y}}$, is just a weighted sums of the d features (columns of $\mathbf{X}$) plus a bias.

For the case of r targets we have that $\hat{\mathbf{Y}}$ and $\hat{\mathbf{W}}$ are matrices where

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{XW} \\ \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_r \end{pmatrix} &= \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_d & \mathbf{1} \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1r} \\ w_{21} & w_{22} & \cdots & w_{2r} \\ \vdots & \vdots & & \vdots \\ w_{d1} & w_{d2} & \cdots & w_{dr} \\ b_1 & b_2 & \cdots & b_r \end{pmatrix} \end{aligned}$$

---

3 Optimization

43 Definition (Increasing/Decreasing Function)
A function is increasing if its derivative is positive and it is decreasing if its derivative is negative.

---

44 Example (Minimum of a Function)

(a) Explain why the minimum value of a smooth function can only occur when the derivative of the function equals zero.

(b) If the derivative of a smooth function equals zero, must the function value be at either a maximum or minimum value? Explain.

---

45 Example (Inflection Point)
Draw the graph of a smooth function with an inflection point.

---

46 Example (Local vs Global Minimums)
Draw the graph of a function that has a two minimum values, a local minimum and a global minimum.

One strategy for minimizing a loss function is to perform a grid search. If the function is smooth, a gradient descent algorithm is likely to be much more efficient.[6]

47 Definition (Gradient Descent Algorithm)
The gradient descent algorithm for minimizing a smooth scalar function, $f(x)$, is outlined below.

Choose a step size, h.
Choose a starting point, x_0.
Define $f_{\text{min}} = f(x_0)$.
Initialize $k = 0$.
while $f(x_k) \leq f_{\text{min}}$:
    $f_{\text{min}} = f(x_k)$
    $x_{k+1} = x_k - f'(x_k)h$
    $k = k + 1$
return x_k

---

48 Definition (Gradient Vector)
The gradient vector, $\frac{dy}{d\mathbf{x}}$, of the function $y = f(\mathbf{x})$ is given by the vector

$$\frac{dy}{d\mathbf{x}} = \left(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_n} \right)^\top.$$

---

[6] The gradient of a scalar function equals the ordinary derivative of the function.

49 Theorem (Direction of Maximum Increase/Decrease)
The direction in the domain of the function $y = f(\mathbf{x})$ in which $f(\mathbf{x})$ increases the fastest is in the direction of the gradient vector $\frac{dy}{d\mathbf{x}}$. The direction in which $f(x)$ decreases the fastest is $-\frac{dy}{d\mathbf{x}}$.

---

50 Example (Gradient Vector)
Consider the function

$$y = f(x_1, x_2) = x_1 + x_1 x_2 + x_2^2.$$

(a) Compute the gradient vector of $f(x_1, x_2)$.

(b) In which direction does $f(x_1, x_2)$ increase the fastest at the point (2,1).

(c) In which direction does $f(x_1, x_2)$ decrease the fastest at the point (2,1).

---

51 Definition (Central Difference Approximation)
The partial derivative of the function $y = f(\mathbf{x})$ with respect to the variable x_i can be approximated by the central difference formula

$$\frac{\partial y}{\partial x_i} \approx \frac{f(x_1, \ldots, x_i + \epsilon, \ldots, x_n) - f(x_1, \ldots, x_i - \epsilon, \ldots, x_n)}{2\epsilon}$$

where ϵ is a small number (e.g. $\epsilon = 10^{-5}$).

---

52 Example (Numerical Approximation of the Gradient Vector)
Use a central difference approximation to numerically approximate the gradient vector $\frac{df(1,1)}{d\mathbf{x}}$ for the function given below using the values $\epsilon = 10^{-1}, 10^{-3}, 10^{-5}$. Compare your approximation to the exact answer.

$$f(x_1, x_2) = e^{1 - x_1 x_2}$$

The gradient descent algorithm can be easily adapted to functions with multiple input variables. Simply replace the scalar quantity $\frac{dy}{dx}$ with the vector quantity $\frac{dy}{d\mathbf{x}}$.

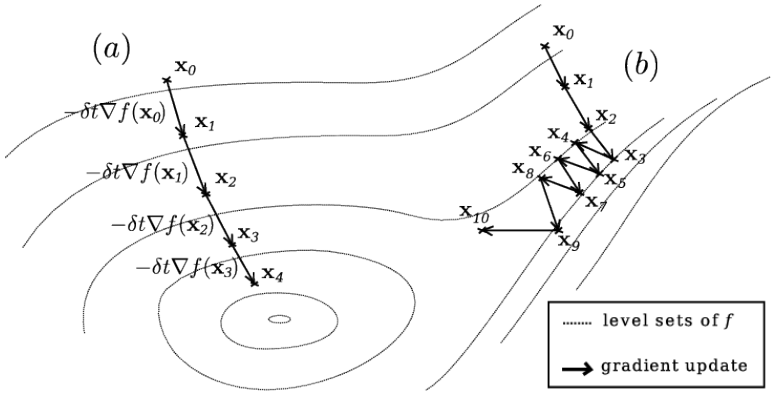
53 Example (Gradient Vector for Linear Regression)

Show that the gradient vector $\frac{d\ell}{d\mathbf{w}}$ for the MSE loss function (L^2 loss function) for linear regression $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ is given by

$$\frac{d\ell}{d\mathbf{w}} = \frac{2}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}).$$

In certain situations, the gradient descent algorithm for finding the minimum of a function can converge very slowly. Consider the following example.

54 Example (Slow Convergence of Gradient Descent)
Which initial condition, (a) or (b), in the diagram below is likely to result in slow convergence of the gradient descent algorithm?[7]



[7] http://ludovicarnold.altervista.org/teaching/

Standardizing input data to a neural network often accelerates convergence.

55 Definition (Standardizing Data)
A data set can be standardized by shifting and scaling the data set as follows:

$$z = \frac{x - \overline{x}}{s_x}$$

where $\overline{x}$ is the mean of x and s_x is the standard deviation of x and z is the standardized data.

---

56 Example (Standardize Linear Regression for Wine)
Which features are most important for determining the quality of (a) red wine (b) white wine? Standardize the features, perform linear regression and examine the feature weights.

Optimization problems can be divided into two types:
(i) convex optimization problems
(ii) non-convex optimization problems

Convex optimization problems are much easier than non-convex optimization problems. Much of classical machine learning deals with convex optimization problems. Training a neural network is a non-convex optimization problem. Until recently, many people believed that it was intractable to train large neural networks.

57 Definition (Convex Function)
A function is convex if

$$f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2).$$

---

58 Example (Line Segment)
The set of all the points, $\mathbf{x}$, on the line segment connecting point $\mathbf{x}_1$ to point $\mathbf{x}_2$ can be specified as

$$\mathbf{x} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \text{ for } 0 \leq \alpha \leq 1.$$

---

59 Example (Convex vs Nonconvex Functions)
Draw some examples of single variable functions that are convex and that are nonconvex.

---

60 Example (Minimum Value of Convex Functions)
(a) Can a strictly convex function have more than one minimum value?

(b) Can a strictly convex function have inflection points?

---

61 Definition (L^2 Regularization)
L^2 **regularization** is a procedure for preventing over-fitting. The term $\alpha\mathbf{w}^\top\mathbf{w}$ is added to the loss function, $\ell(\mathbf{w})$. This term prevents the weights, $\mathbf{w}$, from becoming too large.

$$\begin{aligned}\ell(\mathbf{w}) &= \text{MSE}(\mathbf{w}) + \alpha\mathbf{w}^\top\mathbf{w} \\ &= \frac{1}{n}(\hat{\mathbf{y}}-\mathbf{y})^\top(\hat{\mathbf{y}}-\mathbf{y}) + \alpha\mathbf{w}^\top\mathbf{w}\end{aligned}$$

If the value of the hyper-parameter α is too large, we will have under-fitting, resulting in a large training error. If the value is too small, we may or may not have over-fitting. Over-fitting is characterized by a small training error, but a large test error. The optimal value of α is chosen to minimizes the test error.

---

62 Example (L^2 Regularization of Linear Regression)
Recall that for linear regression $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$. To apply L^2 regularization, we must minimize

$$\ell(\mathbf{w}) = \frac{1}{n}(\hat{\mathbf{y}}-\mathbf{y})^\top(\hat{\mathbf{y}}-\mathbf{y}) + \alpha\mathbf{w}^\top\mathbf{w}.$$

$$\begin{aligned}\frac{d\ell}{d\mathbf{w}} &= \left(\frac{2}{n}(\hat{\mathbf{y}}-\mathbf{y})^\top\frac{d\hat{\mathbf{y}}}{d\mathbf{w}} + 2\alpha\mathbf{w}^\top\right)^\top \\ &= \left(\frac{2}{n}(\mathbf{X}\mathbf{w}-\mathbf{y})^\top\mathbf{X} + 2\alpha\mathbf{w}^\top\right)^\top \\ &= \frac{2}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w}-\mathbf{y}) + 2\alpha\mathbf{w}\end{aligned}$$

Setting $\frac{d\ell}{d\mathbf{w}} = 0$ implies

$$\begin{aligned}\mathbf{X}^\top\mathbf{X}\mathbf{w} + n\alpha\mathbf{w} &= \mathbf{X}^\top\mathbf{y} \\ (\mathbf{X}^\top\mathbf{X} + \alpha_o\mathbf{I})\mathbf{w} &= \mathbf{X}^\top\mathbf{y}\end{aligned}$$

where $\alpha_o = n\alpha$. Therefore, we must repeatedly solve

$$\mathbf{A}(\alpha_o)\mathbf{w} = \mathbf{b}$$

where $\mathbf{A}(\alpha_o) = \mathbf{X}^\top\mathbf{X} + \alpha_o\mathbf{I}$ and $\mathbf{b} = \mathbf{X}^\top\mathbf{y}$ in order to determine α_o that minimize the test error.

---

4 Logistic Regression

- Prediction problems with continuous target variables are called *regression* problems.
- Prediction problems with discrete target variables are called *classification* problems.

Logistic regression is used for classification problems.

63 Example (Regression vs Classification)
Regression or Classification?

(a) Use a person's age to predict their resting heart rate.

(b) Use a person's handwriting to predict if they are male or female.

(c) Spam Filter

(d) Forecast next year's prime interest rate.

---

64 Definition (Probability Distribution)
The vector $\mathbf{p} = \begin{pmatrix} p_1, \ldots, p_N \end{pmatrix}^\top$ is a probability distribution if:

(i) $0 \leq p_k \leq 1$ for $k = 1, \ldots, N$

(ii) $\sum_{k=1}^{N} p_k = p_1 + \cdots + p_N = 1$

---

65 Example (One-Hot-Encoding)
Does one-hot-encoding generate probability distributions? Explain?

The softmax function $\mathbf{p} = f_{\text{smax}}(\mathbf{x})$ (defined below) is useful for creating discrete probability distributions from continuous values. The softmax function makes it possible to apply the regression techniques we have developed so far to classification problems. Logistic regression refers to the method of applying a softmax function to the output of linear regression.

66 Definition (Softmax Function)
The **softmax function** $\mathbf{p} = f_{\text{smax}}(\mathbf{x})$ is defined to be

$$f_{\text{smax}}(\mathbf{x}) = \begin{pmatrix} \sigma_1(\mathbf{x}) & \cdots & \sigma_N(\mathbf{x}) \end{pmatrix}^\top$$

where

$$\sigma_k(\mathbf{x}) = \frac{e^{x_k}}{e^{x_1} + e^{x_2} + \cdots + e^{x_N}}$$

---

67 Example (Softmax Function)
(a) Show that for any input $\mathbf{x}$, the output $\mathbf{p} = f_{\text{smax}}(\mathbf{x})$ defines a probability distribution.

(b) Is it possible for $p_k = 0$? $p_k = 1$?

(c) What is $\mathbf{p}$ when $x_1 = x_2 = \cdots = x_N$?

(d) What is $\mathbf{p}$ when one of the x_k is a very large positive number and the other x_k's are very large negative numbers?

(e) Compare the softmax function to one-hot-encoding?

---

68 Example (Softmax Gives Argmax)
Explain why the softmax function should be called the softargmax function. Hint: Explain how the softmax function "selects" the largest x_k value.

---

69 Example (Linear Regression)

$$\mathbf{X}_{n\times(d+1)} \rightarrow \boxed{\mathbf{w}_{(d+1)\times 1}} \rightarrow \hat{\mathbf{y}}_{n\times 1}$$

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

---

70 Example (Logistic Regression)

$$\mathbf{X}_{n\times(d+1)} \rightarrow \boxed{\mathbf{W}_{(d+1)\times N}} \rightarrow \hat{\mathbf{Y}}_{n\times N} \rightarrow \boxed{f_{\text{smax}}} \rightarrow \hat{\mathbf{P}}_{n\times N}$$

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{X}\mathbf{W} \\ \hat{\mathbf{P}} &= f_{\text{smax}}(\hat{\mathbf{Y}}) \end{aligned}$$

---

71 Example (Introduction to Keras)
Use `Keras` to predict wine quality of white wine using:
(a) linear regression.
(b) logistic regression.

---

72 Definition (Jacobian Matrix)
Consider the function

$$\mathbf{y} = f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix}$$

The **Jacobian matrix** of $f(\mathbf{x})$ is defined to be

$$\frac{d\mathbf{y}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

---

73 Example (Jacobian matrix)
Compute the Jacobian matrix $\frac{d\mathbf{y}}{d\mathbf{x}}$ of the function

$$\mathbf{y} = f(\mathbf{x}) = \begin{pmatrix} x_1 x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

---

74 Definition (Chain Rule)
Consider the functions $y = f(z)$ and $z = g(x)$. Define the function $h = f \circ g$ defined below:

$$y = h(x) = (f \circ g)(x) = f(g(x))$$

The derivative $\frac{dy}{dx}$ is given by the chain rule:

$$\frac{dy}{dx} = \frac{dy}{dz}\frac{dz}{dx}.$$

---

75 Example (Chain Rule)
Let $y = h(x) = \sin(x^2)$. Compute $\frac{dy}{dx}$ using the chain rule

---

The Jacobian matrix makes it conveniently possible to apply the chain rule to multi-variable, vector valued functions $\mathbf{y} = f(\mathbf{x})$.

76 Theorem (Chain Rule Using Jacobian Matrices)
Consider the functions $\mathbf{y} = f(\mathbf{z})$ and $\mathbf{z} = g(\mathbf{x})$. Define the function $h = f \circ g$ defined below:

$$\mathbf{y} = h(\mathbf{x}) = (f \circ g)(\mathbf{x}) = f(g(\mathbf{x}))$$

$$\mathbf{x} \to \boxed{\text{g}} \to \mathbf{z} \to \boxed{\text{f}} \to \mathbf{y}$$

The derivative $\frac{d\mathbf{y}}{d\mathbf{x}}$ is given by the matrix product of Jacobian matrices:

$$\frac{d\mathbf{y}}{d\mathbf{x}} = \frac{d\mathbf{y}}{d\mathbf{z}}\frac{d\mathbf{z}}{d\mathbf{x}}.$$

Proof for 2×2 Jacobian matrices:

$$\begin{aligned} \frac{d\mathbf{y}}{d\mathbf{z}}\frac{d\mathbf{z}}{d\mathbf{x}} &= \begin{pmatrix} \frac{\partial y_1}{\partial z_1} & \frac{\partial y_1}{\partial z_2} \\ \frac{\partial y_2}{\partial z_1} & \frac{\partial y_2}{\partial z_2} \end{pmatrix} \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial y_1}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial y_1}{\partial z_2}\frac{\partial z_2}{\partial x_1} & \frac{\partial y_1}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial y_1}{\partial z_2}\frac{\partial z_2}{\partial x_2} \\ \frac{\partial y_2}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial y_2}{\partial z_2}\frac{\partial z_2}{\partial x_1} & \frac{\partial y_2}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial y_2}{\partial z_2}\frac{\partial z_2}{\partial x_2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{pmatrix} \\ &= \frac{d\mathbf{y}}{d\mathbf{x}} \end{aligned}$$

---

77 Example (Cardinal Sin)
What is the cardinal sin of machine learning?

Answer: Overfitting.

---

function $H(\mathbf{p}, \mathbf{q})$ where

$$\begin{aligned} H(\mathbf{p}, \mathbf{q}) &= -(p_1 \ln(q_1) + \cdots + p_N \ln(q_N)) \\ &= -\sum_{k=1}^{N} p_k \ln(q_k). \end{aligned}$$

---

79 Theorem (Jacobian Matrix of Softmax)
The Jacobian matrix, $\frac{d\mathbf{p}}{d\mathbf{x}}$, of the softmax function $\mathbf{p} = f_{\text{smax}}(\mathbf{x})$ is given by

$$\frac{d\mathbf{p}}{d\mathbf{x}} = \mathbf{D} - \mathbf{p}\mathbf{p}^\top$$

where **D** is a diagonal matrix with **p** along the main diagonal.

The softmax function $\mathbf{p} = f_{\text{smax}}(\mathbf{x})$ is defined to be

$$f_{\text{smax}}(\mathbf{x}) = \begin{pmatrix} \sigma_1(\mathbf{x}) & \cdots & \sigma_N(\mathbf{x}) \end{pmatrix}^\top$$

where

$$\sigma_k(\mathbf{x}) = \frac{e^{x_k}}{s} = p_k$$

where

$$s = e^{x_1} + e^{x_2} + \cdots + e^{x_N}.$$

Now for $i \neq k$,

$$\frac{\partial \sigma_k(\mathbf{x})}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\frac{e^{x_k}}{s}\right) = -\frac{e^{x_i}\, e^{x_k}}{s^2} = -p_i p_k.$$

For $i = k$,

$$\frac{\partial \sigma_k(\mathbf{x})}{\partial x_i} = \frac{\partial}{\partial x_k}\left(\frac{e^{x_k}}{s}\right) = \frac{s e^{x_k} - e^{x_k} e^{x_k}}{s^2} = p_k - p_k p_k.$$

It thus follows that

$$\frac{d\mathbf{p}}{d\mathbf{x}} = \mathbf{D} - \mathbf{p}\mathbf{p}^\top$$

where **D** is a diagonal matrix with **p** along the main diagonal.

---

80 Definition (Cross-Entropy Loss Function)
Let $\mathbf{P}_{n\times N}$ be a one-hot-encoding of N classes. Then, the **cross-entropy loss function** $\ell(\hat{\mathbf{P}}, \mathbf{P})$ is defined to be

$$\ell(\hat{\mathbf{P}}, \mathbf{P}) = -\sum_{i=1}^{n}\sum_{k=1}^{N} p_{ik} \ln(\hat{p}_{ik}).$$

---

81 Example (Stochastic Gradient Descent)
The stochastic gradient descent algorithm has several advantages over the gradient descent algorithm algorithm. ,

- It is simplier.
- It is faster.
- It is less likely to over-fit.

---

82 Example (Stochastic Gradient Descent for Logistic Regression)
Recall that for logistic regression we have:

$$\mathbf{X}_{n\times(d+1)} \rightarrow \boxed{\mathbf{W}_{(d+1)\times N}} \rightarrow \hat{\mathbf{Y}}_{n\times N} \rightarrow \boxed{f_{\text{smax}}} \rightarrow \hat{\mathbf{P}}_{n\times N}$$

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{X}\mathbf{W} \\ \hat{\mathbf{P}} &= f_{\text{smax}}(\hat{\mathbf{Y}}) \end{aligned}$$

We choose the weight matrix **W** to minimize the cross-entropy loss function

$$\ell(\hat{\mathbf{P}}, \mathbf{P}) = \sum_{i=1}^{n} \ell_i(\hat{\mathbf{p}}_i, \mathbf{p}_i)$$

where

$$\ell_i(\hat{\mathbf{p}}_i, \mathbf{p}_i) = -\sum_{k=1}^{N} p_{ik} \ln(\hat{p}_{ik}).$$

Define the "gradient matrix"

$$\frac{d\ell}{d\mathbf{W}} = \begin{pmatrix} \frac{\partial \ell}{\partial W_{11}} & \cdots & \frac{\partial \ell}{\partial W_{1N}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \ell}{\partial W_{(d+1)1}} & \cdots & \frac{\partial \ell}{\partial W_{(d+1)N}} \end{pmatrix}$$

Then we have the following types of iterations:

$$\mathbf{W} = \mathbf{W} - h\frac{d\ell}{d\mathbf{W}} \quad \text{gradient descent}$$

$$\mathbf{W} = \mathbf{W} - h\frac{d\ell_i}{d\mathbf{W}} \quad \text{stochastic gradient descent}$$

For large data sets, the gradient descent algorithm is slow because it takes a long time to compute a single gradient. It typically is much faster to compute an approximate gradient using only a small random sample of the data. This approach is called the **stochastic gradient descent** algorithm. The speed with which an approximate gradient can be computed enables many more gradient steps to be computed in the time it would take for a single full (exact) gradient vector to be computed. Typically, multiple approximate gradient vector steps reduce the loss function faster than a single accurate gradient vector step.

In the stochastic gradient descent algorithm, random samples (without replacement) are taken from the data set and used to compute the gradient vector of the loss function. The size of the random samples (called **minibatches**) is called the batch size. Once all the data in the data set has been used, we say one **epoch** has been completed. Typically, it takes multiple epochs for a stochastic gradient algorithm to converge. The optimal batch size is computing device dependent is typically determined using trial-and-error.

83 Example (Derivation of Stochastic Gradient for Logistic Regression) Show that

$$\frac{\partial \ell_i}{\partial w_{jk}} = -x_{ij}\sum_{r=1}^{N}\frac{p_{ir}}{\hat{p}_{ir}}\frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{ik}}$$

where $\frac{d\hat{\mathbf{p}}_i}{d\hat{\mathbf{y}}_i} = \left(\frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{ik}}\right)$ is the Jacobian matrix of the softmax function.

$$\begin{aligned} \ell_i &= -\sum_{r=1}^{N} p_{ir}\ln(\hat{p}_{ir}) \\ \frac{\partial \ell_i}{\partial w_{jk}} &= -\sum_{r=1}^{N}\frac{p_{ir}}{\hat{p}_{ir}}\frac{\partial \hat{p}_{ir}}{\partial w_{jk}} \end{aligned}$$

Now,

$$\frac{\partial \hat{p}_{ir}}{\partial w_{jk}} = \sum_{s=1}^{N}\frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{is}}\frac{\partial \hat{y}_{is}}{\partial w_{jk}}.$$

Since

$$\frac{\partial \hat{y}_{is}}{\partial w_{jk}} = \begin{cases} 0 & s \neq k \\ x_{ij} & s = k \end{cases}$$

we have that

$$\begin{aligned} \frac{\partial \hat{p}_{ir}}{\partial w_{jk}} &= \frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{ik}}\frac{\partial \hat{y}_{ik}}{\partial w_{jk}} \\ &= \frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{ik}}x_{ij}. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{\partial \ell_i}{\partial w_{jk}} &= -\sum_{r=1}^{N}\frac{p_{ir}}{\hat{p}_{ir}}\frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{jk}}x_{ij} \\ &= -x_{ij}\sum_{r=1}^{N}\frac{p_{ir}}{\hat{p}_{ir}}\frac{\partial \hat{p}_{ir}}{\partial \hat{y}_{jk}} \end{aligned}$$

---

5 Fully Connected Neural Networks

84 Example (Concrete Strength)

(a) Use linear regression to predict the compressive strength of concrete using the features listed in the file `Concrete_train`[8]. Compare the speeds of the Adam optimizer to stochasitic gradient descent (SGD). Would you characterize linear regression as under or over-fitting the data.

(b) Add one or more 8 node (output) fully-connected layers and repeat part (a).

(c) Add one or more 8 node (output) fully-connected layers separated by ReLU activation layers and repeat part (a).

Since the composition of linear functions is linear, not much is gained by stacking linear layers.

85 Example (Stacking Linear Layers)

$$\mathbf{X} \to \boxed{\mathbf{W}_0} \to \mathbf{Z}_1 \to \boxed{\mathbf{w}_1} \to \hat{\mathbf{y}}$$

$$\begin{aligned} \mathbf{Z}_1 &= \mathbf{X}\mathbf{W}_0 \\ \hat{\mathbf{y}} &= \mathbf{Z}_1\mathbf{w}_1 \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{W}_0\mathbf{w}_1 \\ &= \mathbf{X}\mathbf{w} \end{aligned}$$

where $\mathbf{w} = \mathbf{W}_0\mathbf{w}_1$. Thus, two stacked linear layers can be represented by a single linear layer.

---

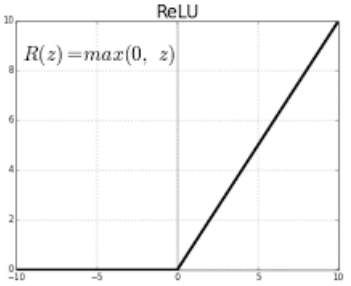
[8]Source: Prof. I-Cheng Yeh, Department of Information Management Chung-Hua University, Hsin Chu, Taiwan 30067, R.O.C., e-mail:icyeh@chu.edu.tw, TEL:886-3-518651y1

To increase the learning capacity of a network, linear layers must be separated by nonlinear layers. One of the simplest possible nonlinear layers is the **ReLU layer**.

86 Definition (ReLU)

Rectified Linear Units (ReLU's) are among the simplest of nonlinear functions:

$$\text{ReLU}(x) = \max\{0, x\}$$



---

87 Example (ReLU)

Assume $Y = \text{ReLU}(X)$ where X is the input data shown below. Determine Y.

x_1	x_2	x_3	y_1	y_2	y_3
3.1	−1.2	0.1			
−4.4	−0.2	0.4			

---

88 Definition (Fully-Connected (Dense) Regression Networks)

A **fully-connected (dense) regression network** is a neural network consisting of L linear layers separated by nonlinear layers such as the ReLU layer. Each linear layer has N_k, $k = 0, 1, \ldots, L-1$, **nodes** (also called **outputs**).

$$\underset{n\times(d+1)}{\mathbf{X}} \to \boxed{\underset{(d+1)\times N_0}{\mathbf{W}_0}} \to \underset{n\times(N_0+1)}{\mathbf{Z}_1} \to \boxed{f_{\text{ReLU}}} \to \underset{n\times(N_0+1)}{\mathbf{A}_1} \to$$

$$\boxed{\underset{(N_0+1)\times N_1}{\mathbf{W}_1}} \to \underset{n\times(N_1+1)}{\mathbf{Z}_2} \to \boxed{f_{\text{ReLU}}} \to \underset{n\times(N_1+1)}{\mathbf{A}_2} \to \cdots \to$$

$$\underset{n\times(N_{L-2}+1)}{\mathbf{A}_{L-1}} \to \boxed{\underset{(N_{L-2}+1)\times N_{L-1}}{\mathbf{W}_{L-1}}} \to \underset{n\times N_{L-1}}{\hat{\mathbf{Y}}}$$