Mathematical Preliminaries

Lecture 2

Subir Varma

Overview

- Differentiation
- Introduction to Probability
- Introduction to Tensors

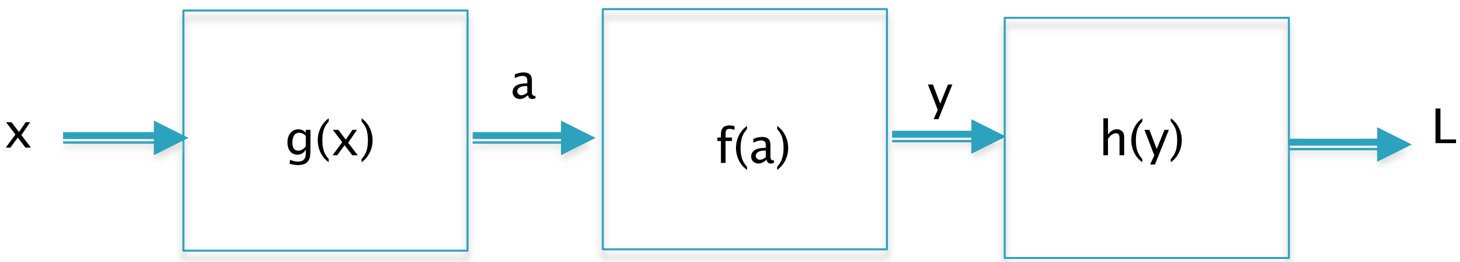
Differentiation

Chain Rule of Derivatives

a = g(x)

y = f(a) = f(g(x))

L = h(y) = h(f(a))
= h(f(g(x)))
= q(x)



$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial x} = \mathrm{h}'(\mathrm{y})\mathrm{f}'(\mathrm{a})\mathrm{g}'(\mathrm{x}) = \mathrm{q}'(\mathrm{x})$$

Easier to evaluate h’, f’, h’ as opposed to q’

Example:

a=g(x) = kx, then g’(x) = $\frac{\partial a}{\partial x} = k$

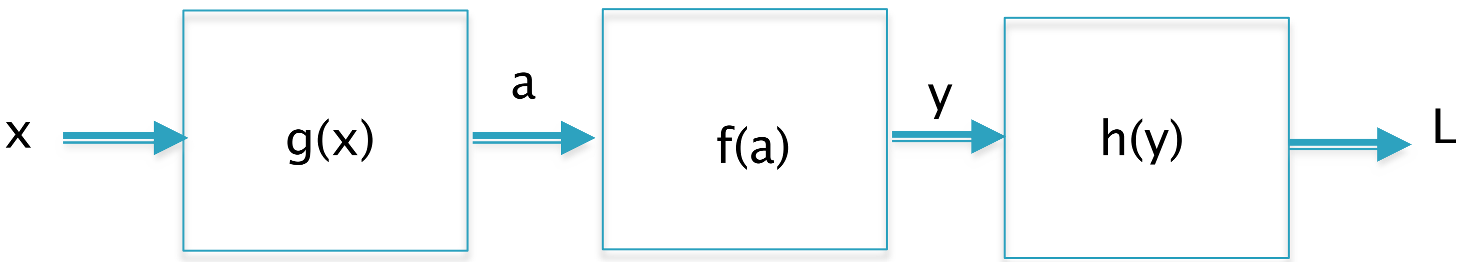
y=f(a) = $\frac{1}{1+\exp(-a)}$, then f’(y)= $\frac{\partial y}{\partial a} = y(1-y)$

L=h(y) = $-[t \log y + (1-t)\log(1-y)], then\ \frac{\partial L}{\partial y} = \frac{y-t}{y(1-y)}$

$$\frac{\partial L}{\partial x} = k(y-t)$$

Backpropagation

$a = g(x)$ $\quad$ $y = f(a) = f(g(x))$ $\quad$ $L = h(y) = h(f(a)) = h(f(g(x))) = q(x)$



Problem: Evaluate $\frac{\partial L}{\partial y}, \frac{\partial L}{\partial a}$ and $\frac{\partial L}{\partial x}$

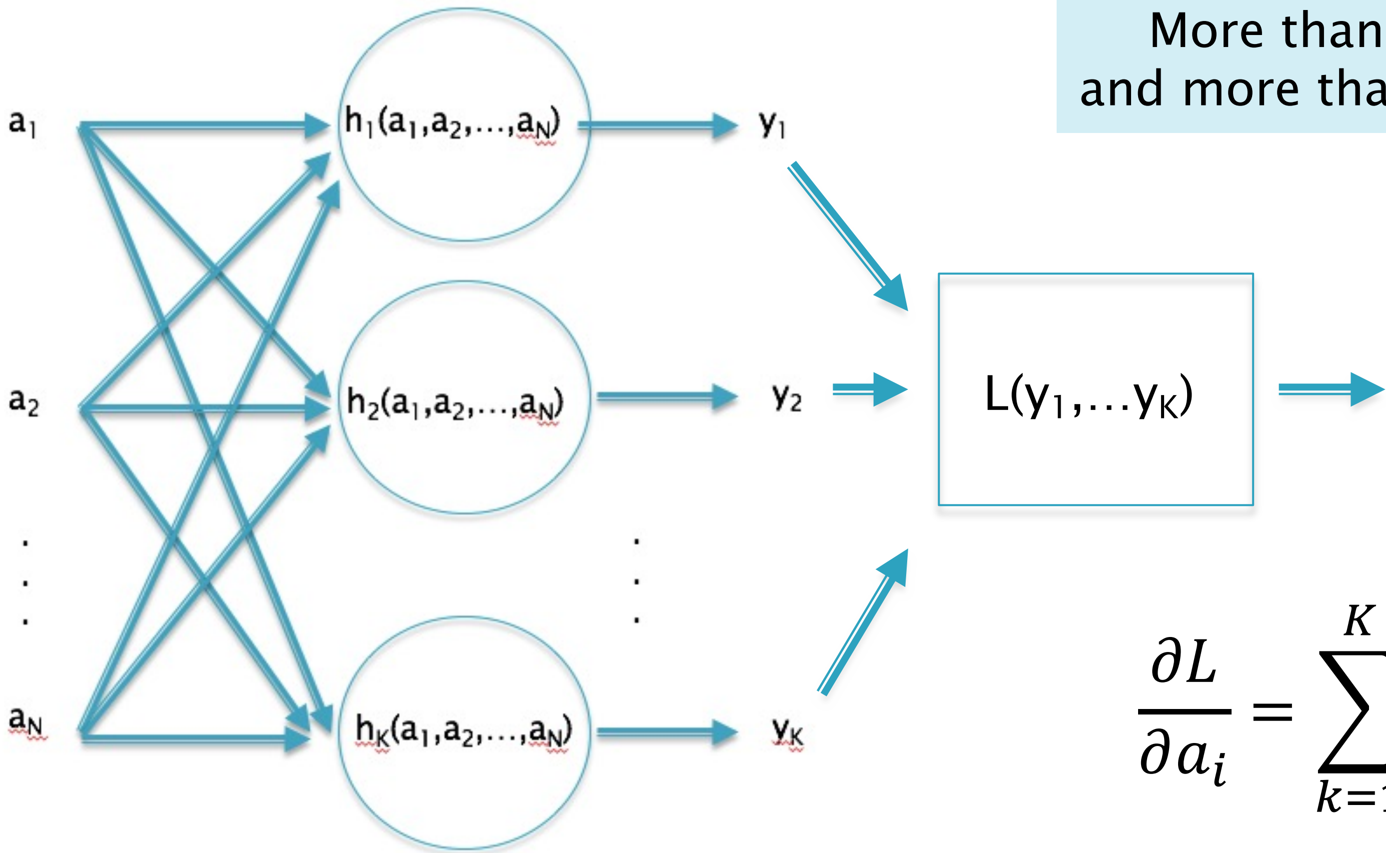
$$\frac{\partial L}{\partial y} = h'(y)$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial a} = h'(y)\ f'(a)$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial x} = h'(y)\ f'(a)\ g'(x)$$

The gradient Calculation Proceeds Backwards!

Chain Rule of Derivatives



More than one input
and more than one output

$$\frac{\partial L}{\partial a_i} = \sum_{k=1}^{K} \frac{\partial L}{\partial y_k}\frac{\partial y_k}{\partial a_i}$$

For K=2, $$\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial a_1} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial a_1}$$

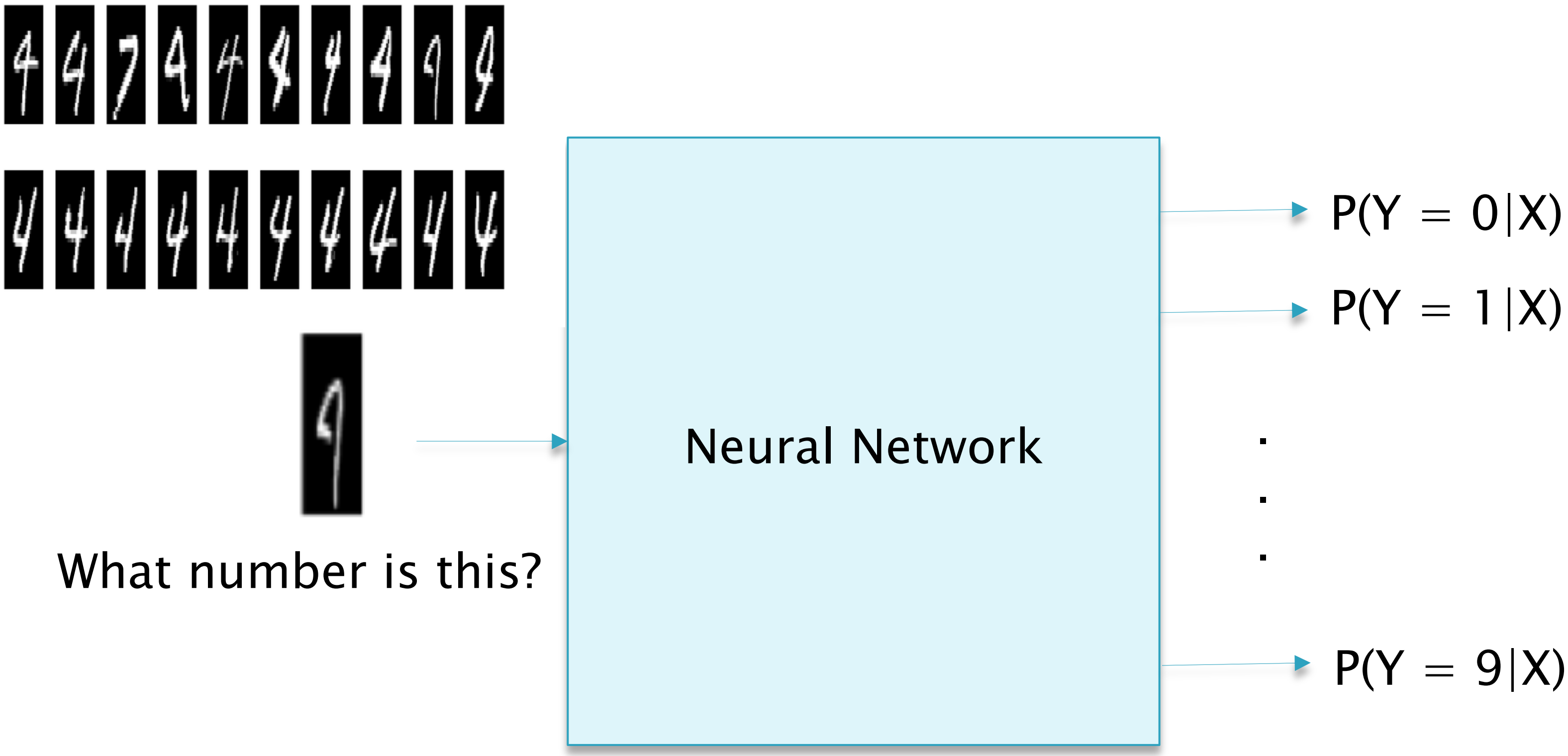
Probability Theory

Probability Theory and ML/DL

Why is Probability Theory useful in ML/DL?

There is an inherent uncertainty in performing tasks such as classification or machine translation.

The Output of a Classification System is a Probability Distribution



Output Y: Probability Distribution

Probabilities in NLP

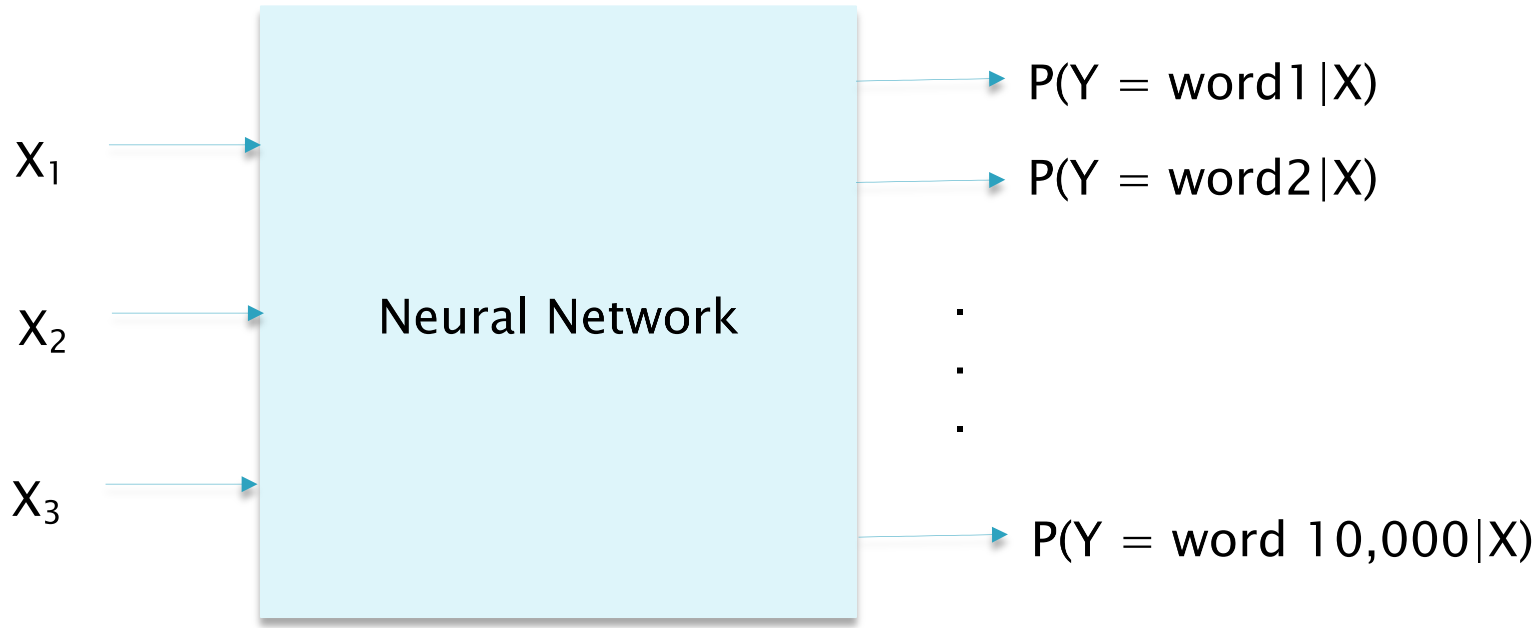
X1, X2, X3 are the first 3 words in a sentence

Ask a Neural Network to guess the next word

The Network computes
$P(Y|X_{1,} X_2, X_3)$ over all possible words Y

Text Generation in NLP

Next Word Prediction



Output Y: Probability Distribution over the Vocabulary

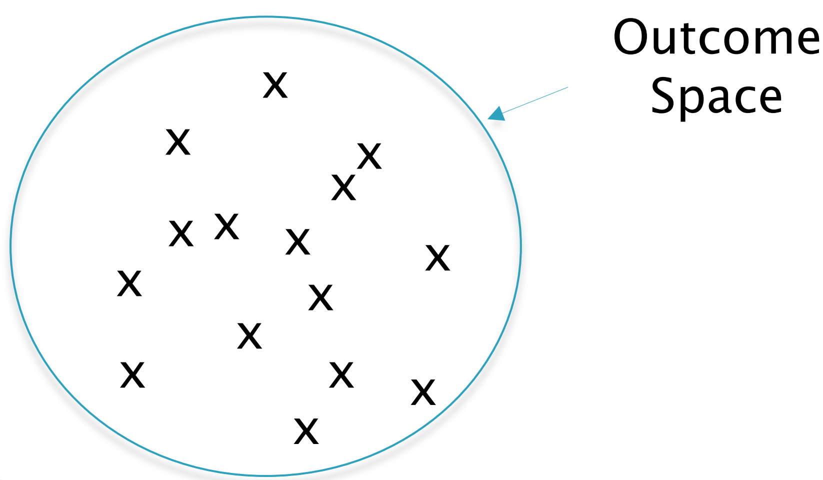
Probability: Outcomes

- <u>Experiment</u>: Example: Pick a card from a deck of cards

Atoms of Probability Theory

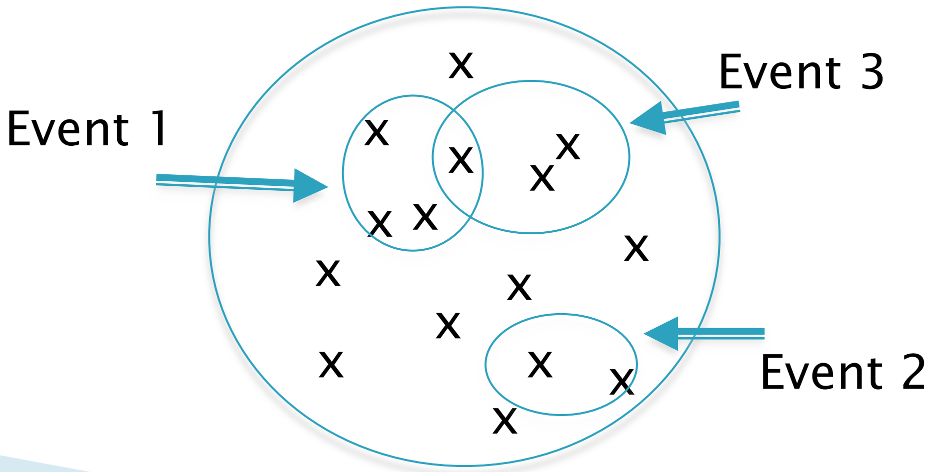
- <u>Outcome of Experiment</u> = One of the 52 cards in the deck
  Outcomes must be:
  Mutually Exclusive – No two outcomes can occur together
  Collectively Exhaustive – At least one of the outcome must occur

- <u>Probability of an Outcome</u>: Likelihood that the outcome will occur at the end of the experiment.
  P(The card is an Ace of Spades) = 1/52



Probability: Events

- Event: A Collection of Outcomes
  Examples:
  E: The card picked is a spade
  F: The card picked is a 7
  G: The card picked is a odd numbered card
  H: The card picked is a Jack, Queen or King
- Two events A and B are Mutually Exclusive if they contain different Outcomes
- Probability of an Event: The Sum of the Probabilities of the Outcomes comprising the Event
  P(E) = 13/52, P(F) = 4/52, P(G) = 20/52, P(H) = 12/52



Laws of Probability

First Law of Probability

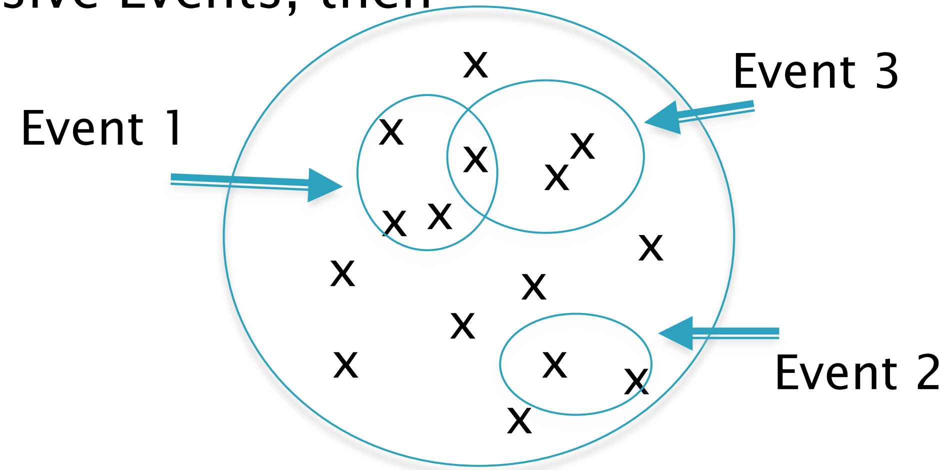
The Probability of an Event is a number between 0 and 1

The sum of the probabilities over all possible outcomes should be 1

Second Law of Probability

If A and B are Mutually Exclusive Events, then

P(A or B) = P(A) + P(B)



Conditional Probability

P(A|B): Probability of the Event A, given that the Event B has occurred

<u>Third Law of Probability</u>

If A and B are two Events, then

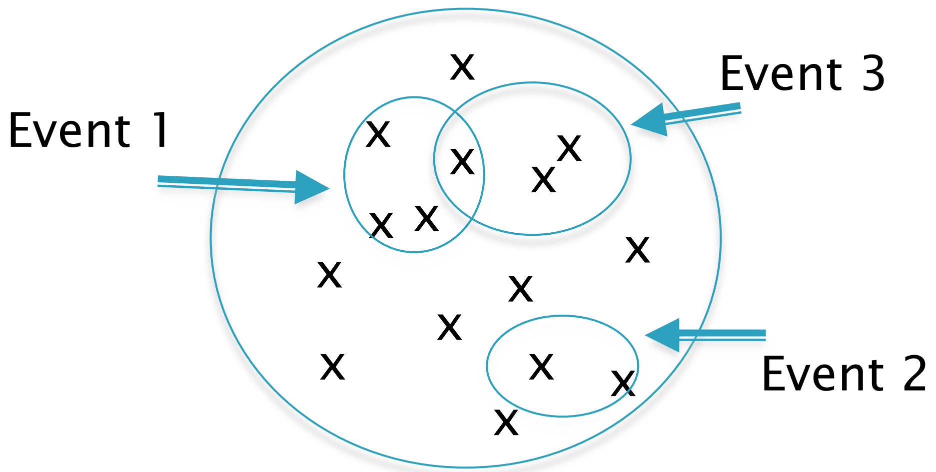
$$P(A|B) = \frac{P(A \;\; and \;\; B)}{P(B)}$$

Example:

A: The card picked is the queen of any suit

B: The card picked is a face card

$$P(A|B) = \frac{4/52}{12/52} = 1/3$$



P(Event 1|Event 3) = 1/3

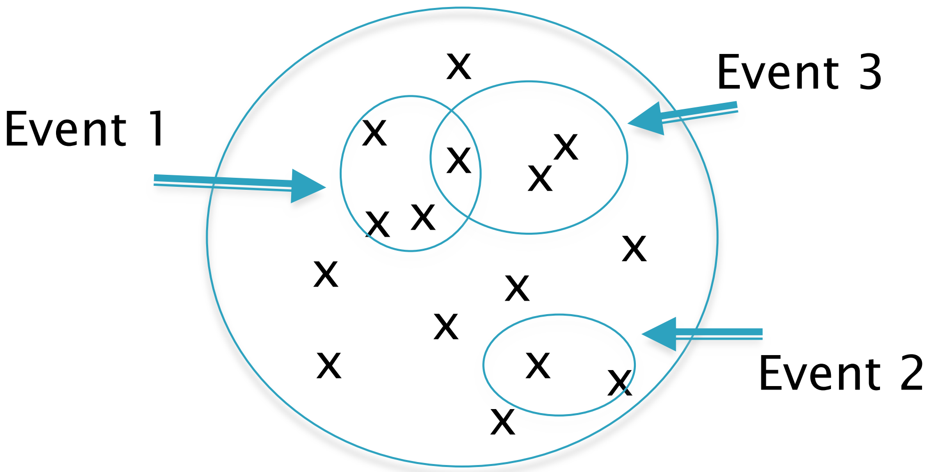
Independent Events

Two Events A and B are Independent if knowing that B has occurred does not influence the probability of A occurring

<u>Fourth Law of Probability</u>

If A and B are independent Events, then

$$P(A|B) = P(A) \quad \text{and} \quad P(A \text{ and } B) = P(A) \times P(B)$$



Random Variables

Random Variables

- When the Outcomes in a Probability Model are numbers, then these numbers are referred to as Random Variables.
- A Random Variable can be:
  - Discrete
    Example: Outcome of a die toss
  - Continuous
    Example: Annual Rainfall in Santa Clara

Probability Density Functions: Discrete Random Variables

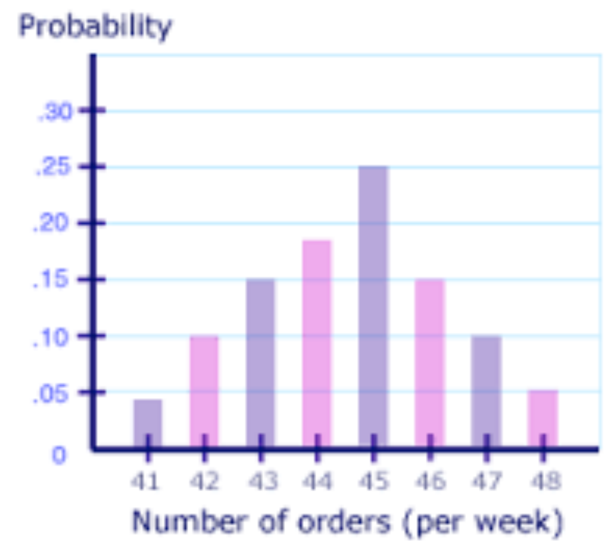
$P(X = x_1) = p_1$

$P(X = x_2) = p_2$

.

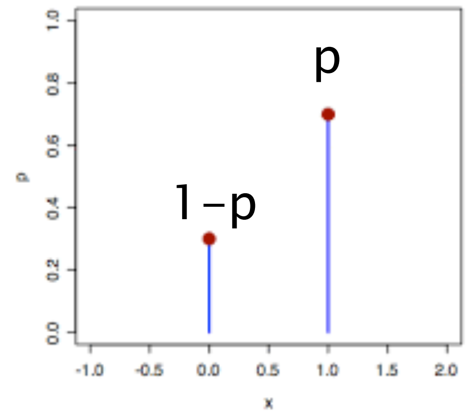
.

$P(X = x_n) = p_n$



$p_1 + p_2 + \ldots + p_n = 1$

Probability Density Functions: Discrete RVs



Bernoulli Distribution X = {0,1}

$$P(X=1)=p$$

$$P(X=0)=1-p=q$$

Outcome of a single (biased) coin toss

Probability Density Functions: Binomial Distribution

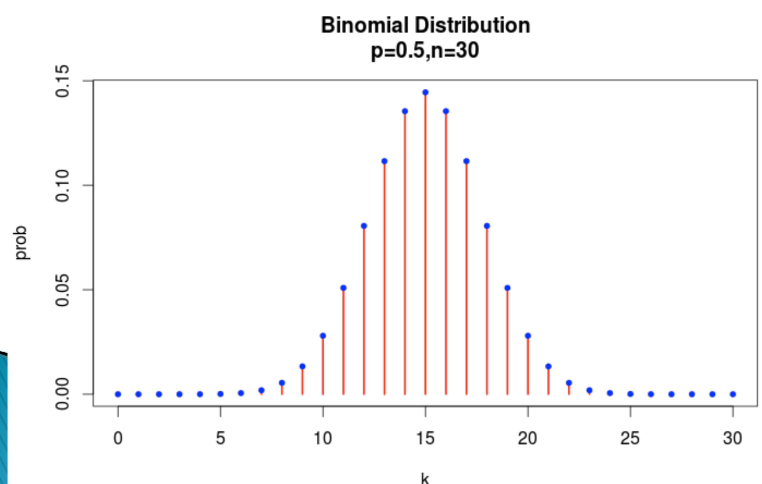
<u>Binomial Distribution</u>

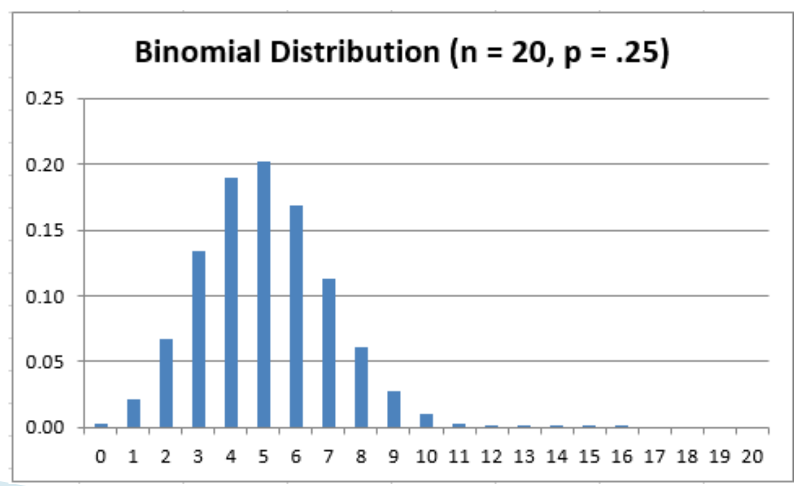
X = {0,1, 2, …, n}

$$P(X=r)=\frac{n!}{r!(n-r)!}p^r(1-p)^{n-r} \quad r=0,1,\ldots,n$$

$$X = \sum_{i=1}^{n} Y_i$$

Toss a coin n times
Y_i: Outcome of the i^{th} toss = {0,1}
X: Total number of Heads (1's)





Probability Density Functions: Continuous RVs

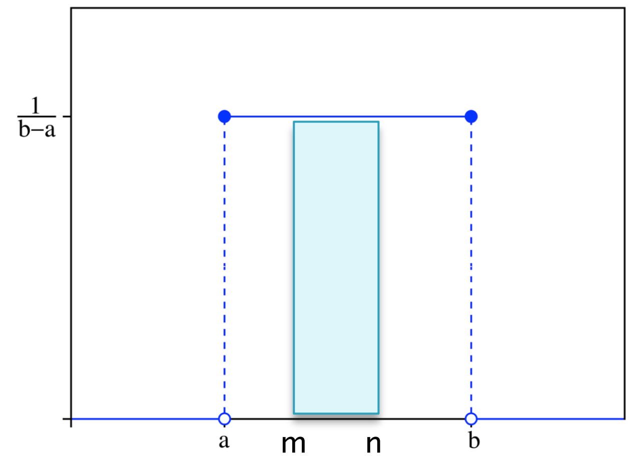
Probabilities are defined over Intervals

Examples:

- Uniform Distribution

$X = [a, b]$

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if} \quad a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

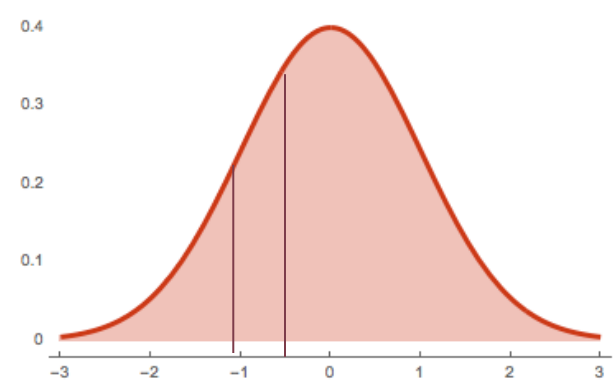


$$P(X \in [m,n]) = \frac{n-m}{b-a}$$

- Normal Distribution

$X = (-\infty, +\infty)$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

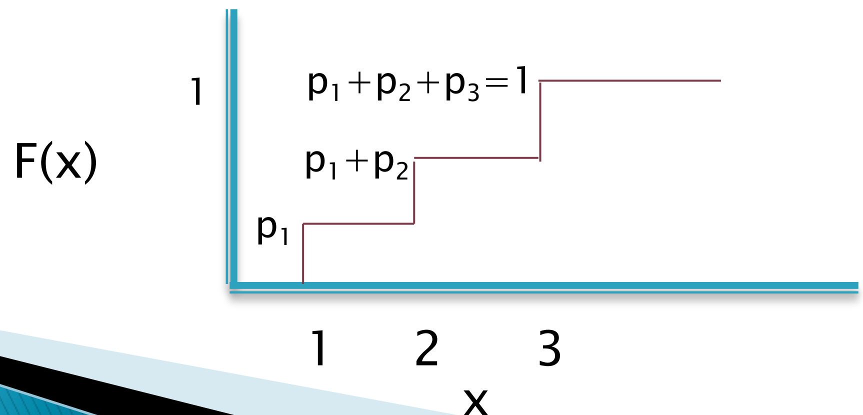


Probability Distributions Functions

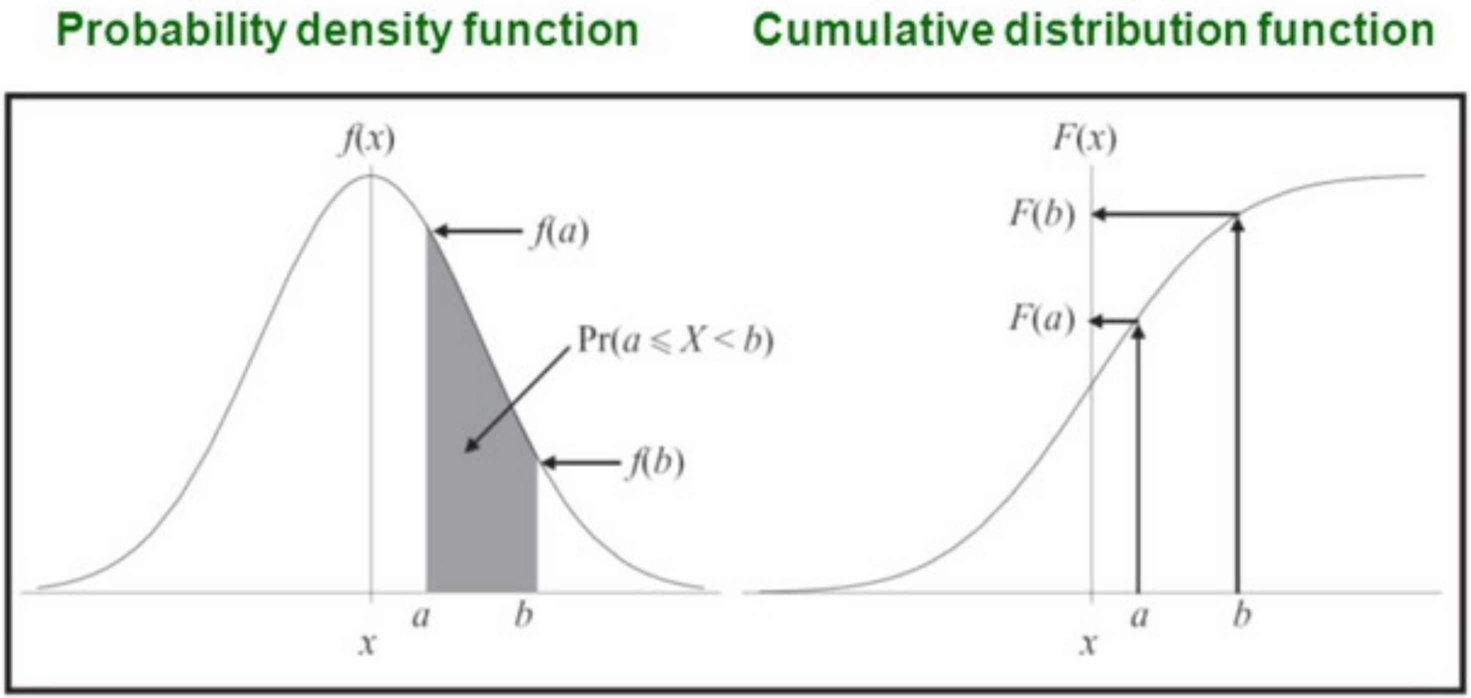
- Given a Probability Density
  X={1,2,…,n}
  $P(X = k)=p_k$

- The Probability Distribution of X is defined as

$$F(X \le k) = \sum_{i=1}^{k} p_k$$



Probability Distribution Function



$$P(a \leq X \leq b) = F(b) - F(a)$$

$$F(x) = P(X \leq x)$$

$$0 \leq F(x) \leq 1$$

Joint Probability Distributions

- Given two RVs (X,Y), their Joint Probability Distribution is given by

$$p_{ij} = P((X,Y) = (i,j))$$

$$\sum_{i} \sum_{j} p_{ij} = 1$$

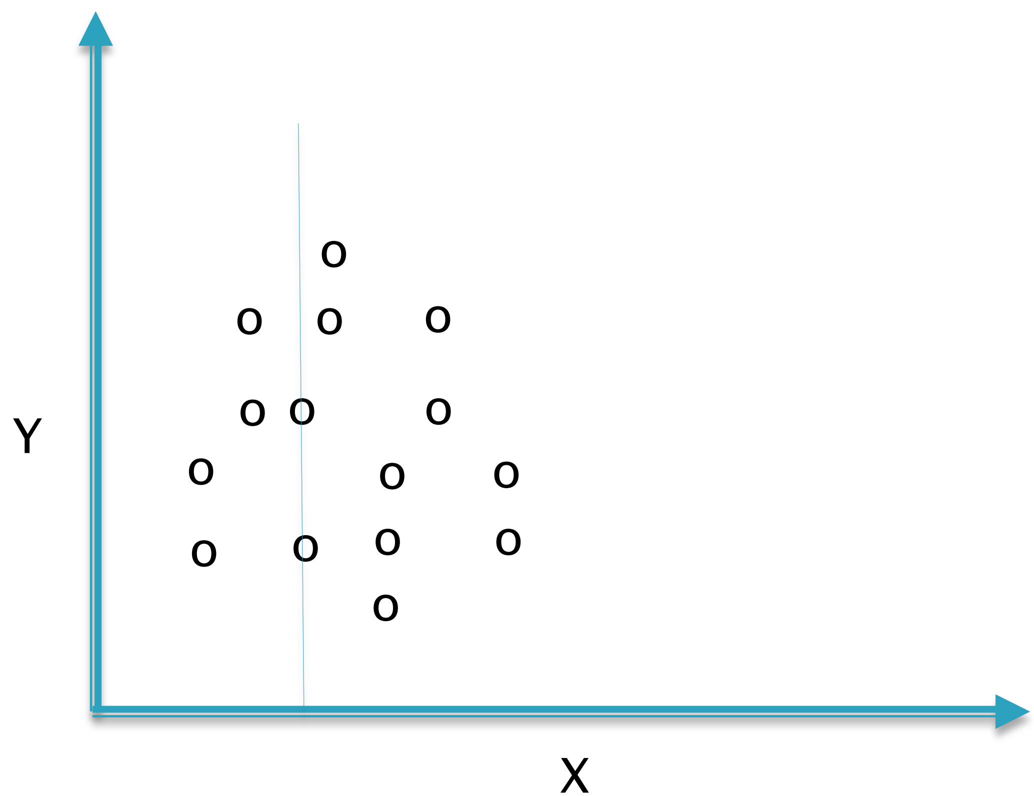
- Sum Rule

$$P(X) = \sum_{Y} P(X,Y)$$

Marginal Probability

- Conditional Probability

$$P(Y|X) = \frac{P(X,Y)}{P(X)} = \frac{P(X,Y)}{\sum_Y P(X,Y)}$$



Joint Probability Distributions

- Product Rule

$$P(X,Y) = P(Y|X)p(X) = P(X|Y)p(Y)$$

The above two rules imply that

$$P(X) = \sum_{Y} P(X,Y) = \sum_{Y} P(X|Y)p(Y)$$

Law of Marginal Probabilities

Sometimes P(X) is intractable but P(X|Y) is easier to compute

Independence and Conditional Independence

Independence

$$P(Y|X) = P(Y)$$
$$P(X,Y) = P(X)P(Y)$$

Conditional Independence

$$P(X,Y|Z) = P(X|Z)P(Y|Z)$$

Expectation, Variance, Co-Variance

- Discrete RVs

$$E(X) = \mu_X = \sum_i p_i x_i$$

$$Var(X) = \sigma_X^2 = \sum_i p_i (x_i - \mu_X)^2$$

$$\mathrm{cov}(X,Y) = E\left[(X - E(X))(Y - E(Y))\right] = \sum_i \sum_j p_{ij} (x_i - \mu_X)(y_j - \mu_Y)$$

- Statistics of a Random Variable
- Allows you to infer properties of the RV without knowing the entire distribution

- Continuous RVs

$$\mu_X = \int_\Omega f(x)dx$$

$$\sigma_X^2 = \int_\Omega (x - \mu_X)^2 f(x) dx$$

$$\mathrm{cov}(X,Y) = \iint f(x,y)(x - \mu_X)(y - \mu_Y) dx\, dy$$

Problems in Statistics

Estimation Problem

Given Data Samples $\{X_1, X_2, \ldots, X_n\}$, find the statistics of the Random Variable X that generated this data

- Distribution of X (has complete information about X)
- Mean, Variance of X

This an Important Problem to solve in Deep Learning:
Once we know the Distribution f(X) of X, we can generate new samples of X

How?

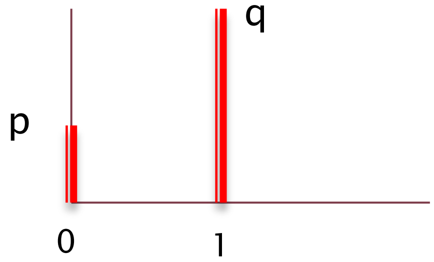
Problems in Statistics

The Sampling Problem

Given a Random Variable X with a known distribution, generate data samples $\{X_1,X_2,\ldots,X_n\}$, that follow this distribution

Sampling from a Discrete Distribution

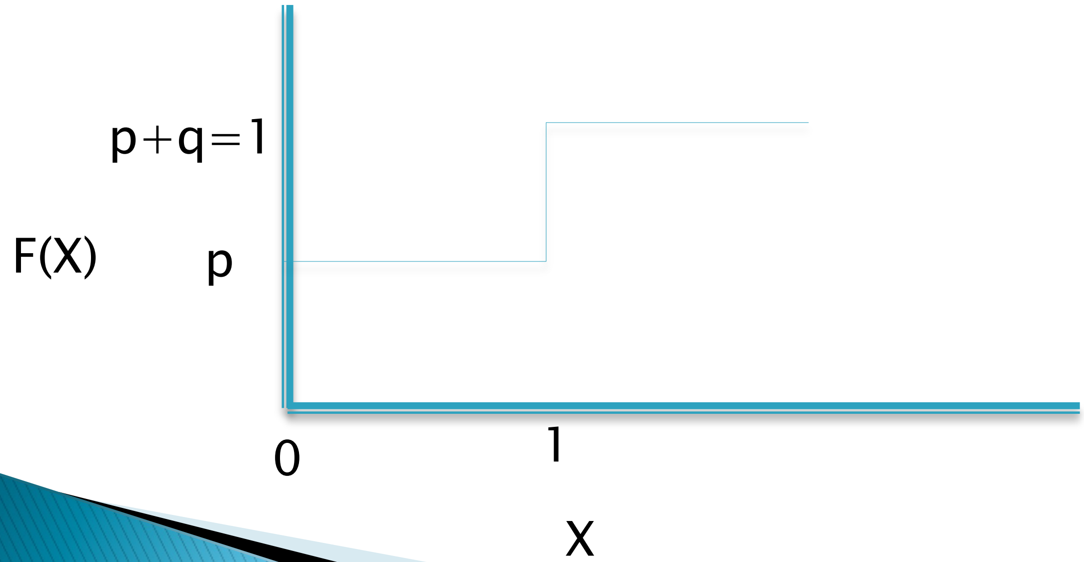
Sampling Problem: Generate random numbers which follow this distribution



Example: Bernoulli Distribution

$P(X=0) = p$

$P(X=1) = q = 1-p$



Solution:
Compute the Distribution Function F(X) for X

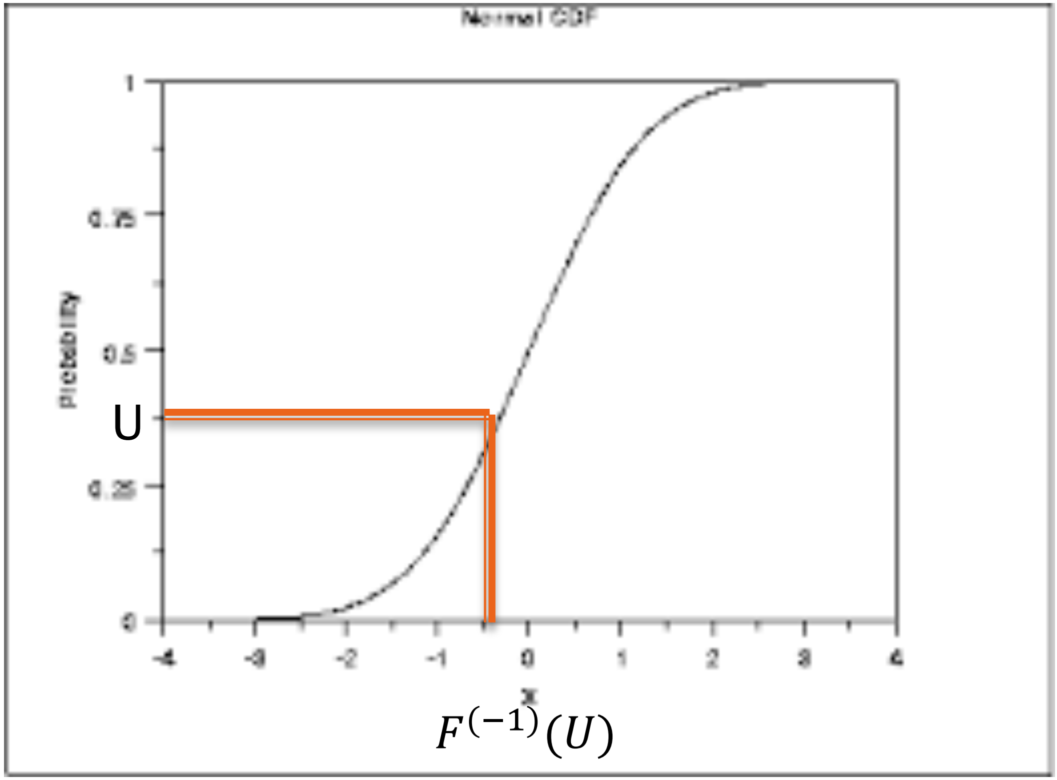
Generate a Uniform U in [0,1]

If U is on [0,p] then generate X=0
Otherwise generate X=1

Sampling from a Continuous Distribution

F(x)

$F(x) = P(X \leq x)$



$$Y = F^{(-1)}(U)$$

$$P(Y \leq y) = P\left(F^{(-1)}(U) \leq y\right) = P(U \leq F(y) = F(y)$$

Random Sequences

Random Sequences

- Sequence of Random Variables

$$X_1, X_2, ..., X_n$$

For example successive toss of a dice

- Joint Probability Distribution

$$P(X_1, X_2, ..., X_n)$$

Examples:
X is a word, $(X_1, X_2, \ldots, X_n)$ is a sentence

X is a pixel, $(X_1, X_2, \ldots, X_n)$ is an image

Random Sequences: Special Cases

- Sequence of <u>Independent Random Variables</u>

$$P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2)\ldots P(X_n)$$

Example: Tossing a die multiple times

Chain Rule of Probabilities

$$P(X_1,X_2) = P(X_2|X_1)P(X_1)$$

$$P(X_1,X_2,X_3) = P(X_3|X_1,X_2)P(X_1,X_2)$$
$$= P(X_3|X_1,X_2)P(X_2|X_1)P(X_1)$$

More Generally for a Sequence of RVs

$$P(X_1,X_2,\ldots,X_n) = P(X_n|X_1,\ldots,X_{n-1})\ P(X_{n-1}|X_1,\ldots,X_{n-2})\ldots P(X_2|X_1)\ P(X_1)$$

Joint Probability Distribution: What we are really interested in

Product of Conditional Probability Distributions: Easier to compute with a Neural Network

This formula is Fundamental in Deep Learning

In NLP, enables the system to generate text

Tensors

(NumPy Arrays)

Scalars, Vectors

- Scalars: A single number

```
>>> import numpy as np
>>> x = np.array(12)
>>> x
array(12)
>>> x.ndim
0
```

- Vectors: A 1-D array of numbers

Column Vector: $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

Row Vector: $\mathbf{x} = (x_1\ x_2\ \ldots\ x_n)$

```
>>> x = np.array([12, 3, 6, 14])
>>> x
array([12, 3, 6, 14])
>>> x.ndim
1
```

Matrices

<u>Matrix</u>: A 2-D array of numbers

$$\boldsymbol{W} = \begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{pmatrix}$$

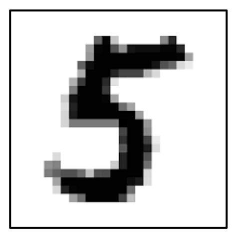
$\boldsymbol{W}$ is a matrix with m rows and n columns

$w_{i,j}$ The element at the i^{th} row and j^{th} column

$W_{i,:}$ The i^{th} row of the matrix

$W_{:,j}$ The j^{th} column of the matrix

```
>>> x = np.array([[5, 78, 2, 34, 0],
                  [6, 79, 3, 35, 1],
                  [7, 80, 4, 36, 2]])
>>> x.ndim
2
```

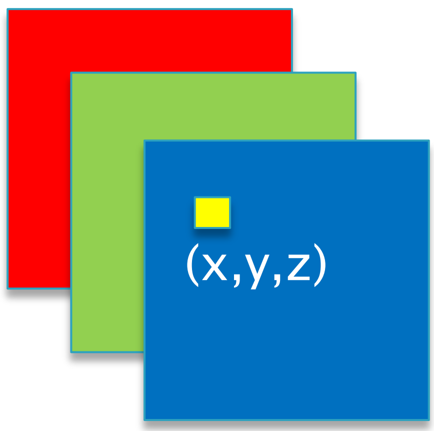


Example of a matrix:
Pixels in a grayscale image
28x28 array of numbers

Tensors

<u>Tensor</u>: A generalization of matrices to n dimensions $(x_1,x_2,\ldots,x_n)$

Example:



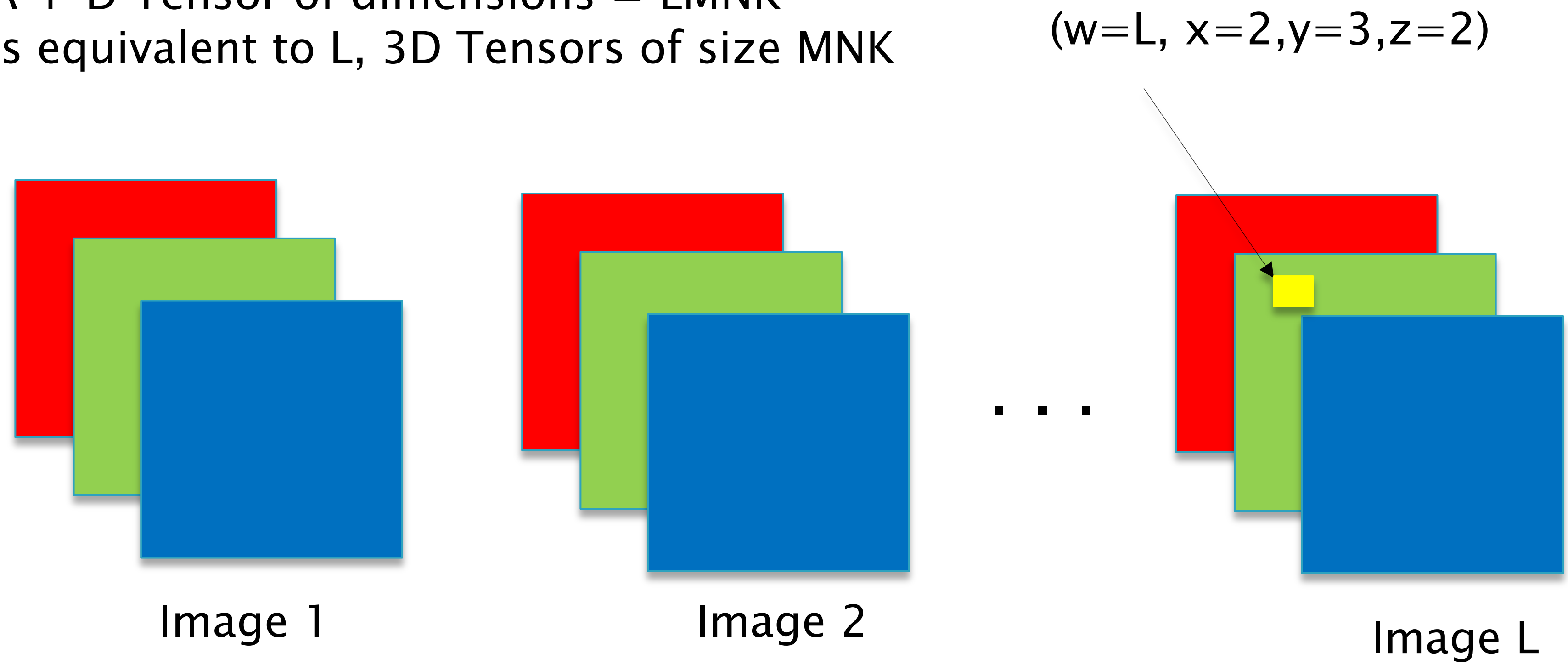
Example of a 3D tensor:
RGB pixels in a color image

A 3-D Tensor of dimensions = KMN is equivalent to K matrices of size MN

```
>>> x = np.array([[[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]]])
>>> x.ndim
3
```

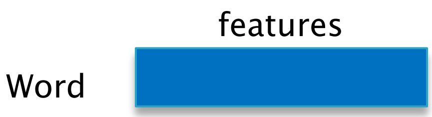
Example: 4D Tensors in Image Processing

A 4-D Tensor of dimensions = LMNK
is equivalent to L, 3D Tensors of size MNK

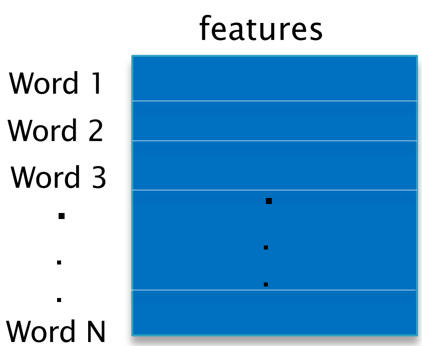


Example of a 4D tensor:
A sample of L color images
(sample #, height, width, channels)

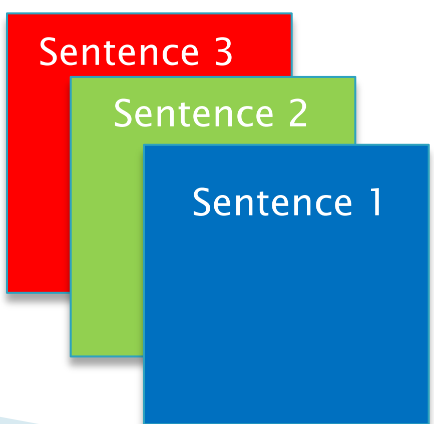
Example: Tensors in NLP



A word can be represented by a vector of features



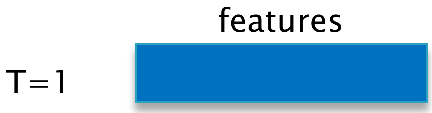
A sentence can be represented by a 2D matrix



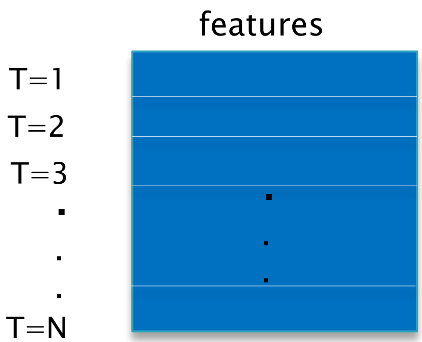
A collection of sentences can be represented by a 3D tensor

(samples, words, features)

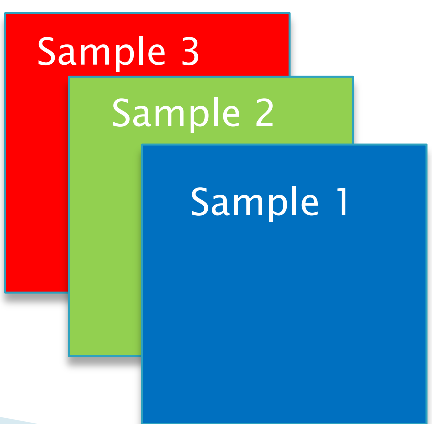
Example: Time Series



Set of features at T = 1



A single sample from T=1 to T=N



A collection of samples can be represented by a 3D tensor

(samples, timesteps, features)

Key Attributes of a Tensor

- Number of Axes (Rank): X.ndim in NumPy
- Shape: Tuple of integers that describes how many dimensions the tensor has along each axis. Obtained with X.shape in NumPy. Examples:
  - 2D Tensor: (3,5)
  - 3D Tensor: (3,3,5)
  - Vector: (5,)
  - Scalar: ( )
- Data Type: Could be float32, uint8, float64 etc. Obtained by X.dtype in NumPy

Example: MNIST Dataset

Download the MNIST dataset

```
from keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Get its shape

```
>>> print(train_images.shape)
(60000, 28, 28)
```

Display an element of the dataset

```
digit = train_images[4]

import matplotlib.pyplot as plt
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```

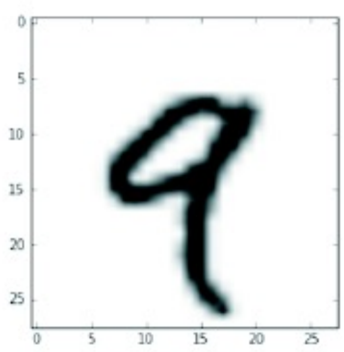


Figure 2.2 The fourth sample in our dataset

Manipulating Tensors: Slicing

Selecting specific elements in a tensor is called tensor slicing

```
>>> my_slice = train_images[10:100]
>>> print(my_slice.shape)
(90, 28, 28)
```

Select images #10 to #99

```
>>> my_slice = train_images[10:100, :, :]
>>> my_slice.shape
(90, 28, 28)
```

```
>>> my_slice = train_images[10:100, 0:28, 0:28]
>>> my_slice.shape
(90, 28, 28)
```

These are both equivalent

```
my_slice = train_images[:, 14:, 14:]
```

Select 14x14 pixels in the bottom right corner of all images

```
my_slice = train_images[:, 7:-7, 7:-7]
```

Crop images to patches of 14x14 pixels centered in the middle

Tensor Reshaping

Reshaping: Re-arranging the tensor's rows and columns to match a target shape

```
>>> x = np.array([[0., 1.],
                  [2., 3.],
                  [4., 5.]])
>>> print(x.shape)
(3, 2)
```

Original tensor

```
>>> x = x.reshape((6, 1))
>>> x
array([[ 0.],
       [ 1.],
       [ 2.],
       [ 3.],
       [ 4.],
       [ 5.]])
```

After reshape operation 1

```
>>> x = x.reshape((2, 3))
>>> x
array([[ 0.,  1.,  2.],
       [ 3.,  4.,  5.]])
```

After reshape operation 2

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.]])
>>> a.flatten()
array([ 1.,  2.,  3.,  4.,  5.,  6.])
```

Flatten operation

Creates 1D array

Tensor Operations

Elementwise Operations: Operations applied independently to each entry of the tensor

```
z = x + y               <- Element-wise addition
z = np.maximum(z, 0.)   <- Element-wise relu
```

Broadcasting

If the shape of the two tensors is different, the smaller tensor is broadcasted to match the shape of the larger tensor, in 2 steps:

1. Axes are added to the smaller tensor to match the ndim of the larger tensor
2. The smaller tensor is repeated alongside these new axes

```
import numpy as np
x = np.random.random((64, 3, 32, 10))   <- x is a random tensor with shape (64, 3, 32, 10).
y = np.random.random((32, 10))          <- y is a random tensor with shape (32, 10).
z = np.maximum(x, y)                    <- The output z has shape (64, 3, 32, 10) like x.
```

Tensor Dot Product

```
import numpy as np

z = np.dot(x, y)
```

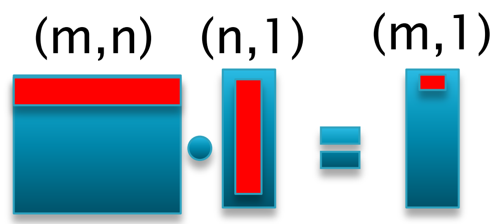
If x and y are vectors

```
z = 0.
for i in range(x.shape[0]):
    z += x[i] * y[i]
return z
```

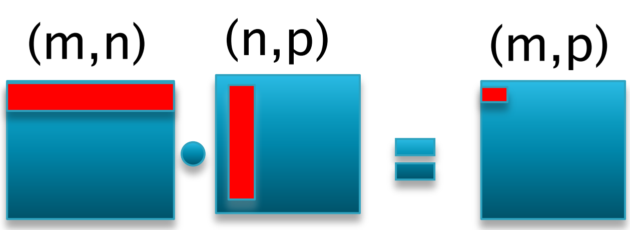
Returns a Scalar

If x is a matrix and y is a vector

```
z = np.zeros(x.shape[0])
for i in range(x.shape[0]):
    for j in range(x.shape[1]):
        z[i] += x[i, j] * y[j]
return z
```



Both x and y are matrices



More Generally

```
(a, b, c, d) . (d,) -> (a, b, c)
(a, b, c, d) . (d, e) -> (a, b, c, e)
```

Creating Arrays

```
>>> np.arange(5, dtype=float)
array([ 0.,  1.,  2.,  3.,  4.])
>>> np.arange(1, 6, 2, dtype=int)
array([1, 3, 5])
```

Arrays with consecutive integers

```
>>> np.ones((2,3), dtype=float)
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> np.zeros(7, dtype=int)
array([0, 0, 0, 0, 0, 0, 0])
```

Arrays with 1's or 0's

```
>>> np.random.rand(2,3)
array([[ 0.50431753,  0.48272463,  0.45811345],
       [ 0.18209476,  0.48631022,  0.49590404]])
>>> np.random.rand(6).reshape((2,3))
array([[ 0.72915152,  0.59423848,  0.25644881],
       [ 0.75965311,  0.52151819,  0.60084796]])
```

```
>>> np.random.randint(5, 10)
9
```

Generating Random Numbers

```
>>> l = range(10)
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> np.random.shuffle(l)
>>> l
[4, 9, 5, 0, 2, 7, 6, 8, 1, 3]
```

Random Shuffle

Transpose of a Matrix

$$\boldsymbol{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{pmatrix}$$

then

$$\boldsymbol{W}^T = \begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \\ w_{1,3} & w_{2,3} \end{pmatrix}$$

In general

$$(\boldsymbol{W}^T)_{i,j} = w_{j,i}$$

Supplementary Reading

- For Probability Theory: Chapters 2 of “Deep Learning” by Goodfellow, Bengio and Courville. http://www.deeplearningbook.org/
- For Tensors: Chapter 2, Sections 2.2 and 2.3 of Chollet