Imperial College London

May 11th, 2020

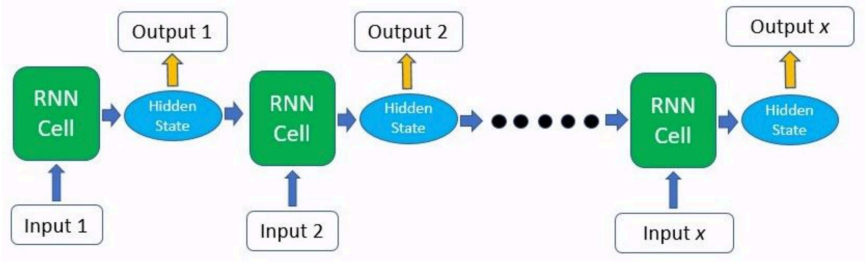
Recurrent Networks and LSTMs

Olivier Dubrule and Navjot Kukreja

1

Imperial College London

Basic Recurrent Network Structure



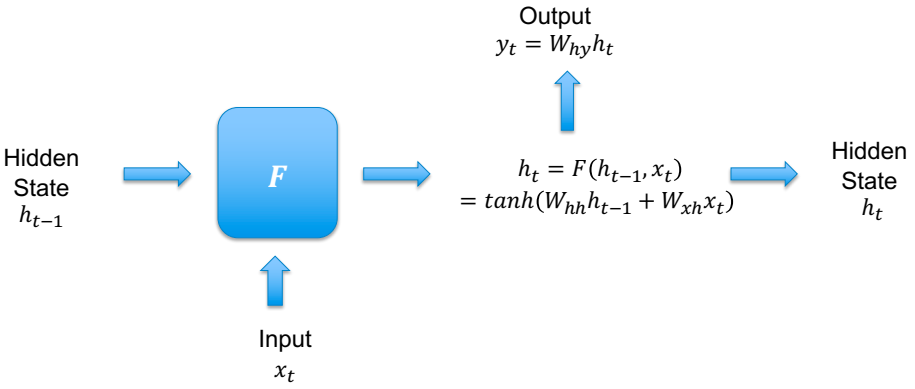
Example:
Based on a (usually long) text used for Training, we wish to generate "similar" text on a character by character basis.

https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

2

Imperial College London

The Basic RNN Cell for time *t*



https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

3

Imperial College London

A Simplistic Character Sequence Example (3)

Training Workflow:

For a sequence of p (say $p = 20$) characters:

- Take first character x_1, combine with current hidden state h_0 (the size of h_0 is a hyper-parameter) to derive h_1, calculate associated output vector y_1 of size equal to vocabulary, transform y_1 into Softmax vector, calculate cross-entropy with second character x_2 of sequence.
- Take this second character as input to re-do what we just did with the first character and again calculate loss function using the value of the third character of the sequence, and so on until we reach the end of the sequence.
- Add up all the above loss functions.
- Do back-propagation to calculate gradients according to each trainable parameter.
- Modify parameters by gradient descent

Move to next sequence of p characters, until end of Training Set is reached.

4

Imperial College London

A Simplistic Character Sequence Example (4)

Text Generation Workflow:

Generate a new sequence of n characters:

Randomly sample first character x_1, combine with initial null state vector value h_0 to derive new vector h_1, calculate associated output vector y_1 of size equal to vocabulary size, transform y_1 into Softmax vector, sample new character x_2 based on Softmax probability, and so on, until the required number of characters have been generated.

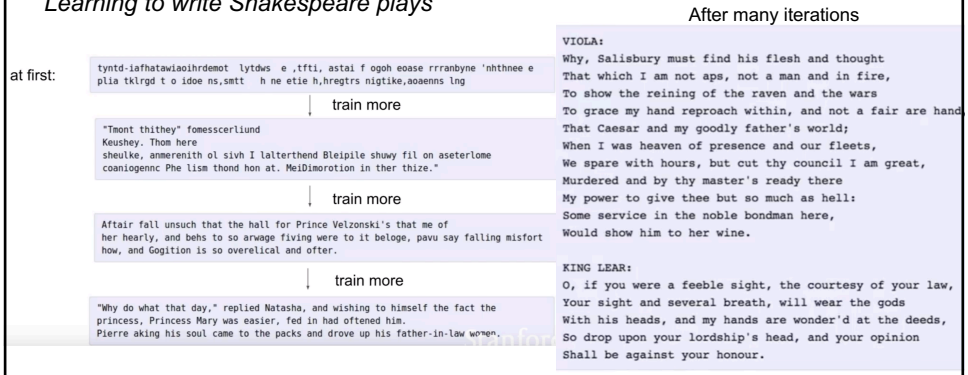
5

Imperial College London

A Simple Language Processing Python Code

https://gist.github.com/karpathy/d4dee566867f8291f086

Learning to write Shakespeare plays

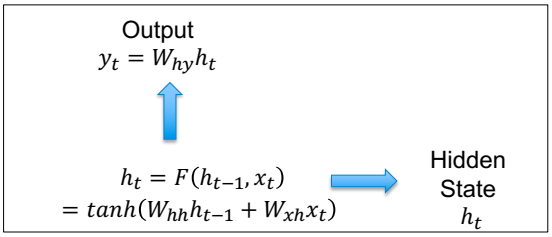


Andrej Karpathy's Blog: *The Unreasonable Effectiveness of Recurrent Neural Networks*

6

Imperial College London

A Summary



The weight matrices $\boldsymbol{W_{hh}}$, $\boldsymbol{W_{xh}}$ and $\boldsymbol{W_{hy}}$ contain the trainable parameters and are the same for each time t .

The hidden vector $\boldsymbol{h_{t-1}}$, the size of which is a hyperparameter, changes with t and contains recent historical information about the $\boldsymbol{x_t}$ sequence before t.

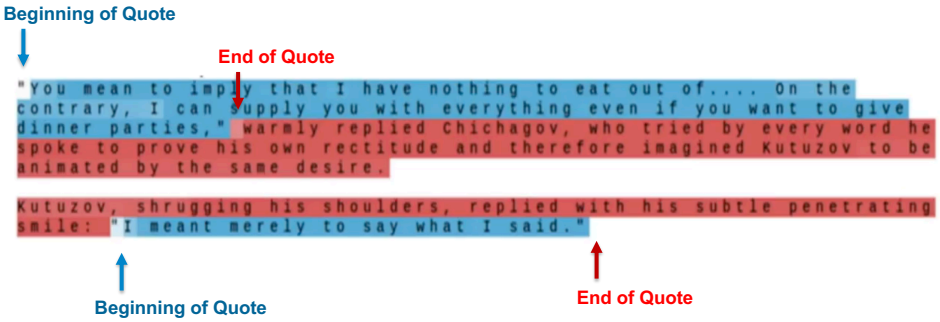
$\boldsymbol{x_t}$ is the new data at time t which is combined with $\boldsymbol{h_{t-1}}$ to predict $\boldsymbol{x_{t+1}}$.

https://www.youtube.com/watch?v=6niqTuYFZLQ

7

Imperial College London

Visualizing one coordinate of h_t as a function of t

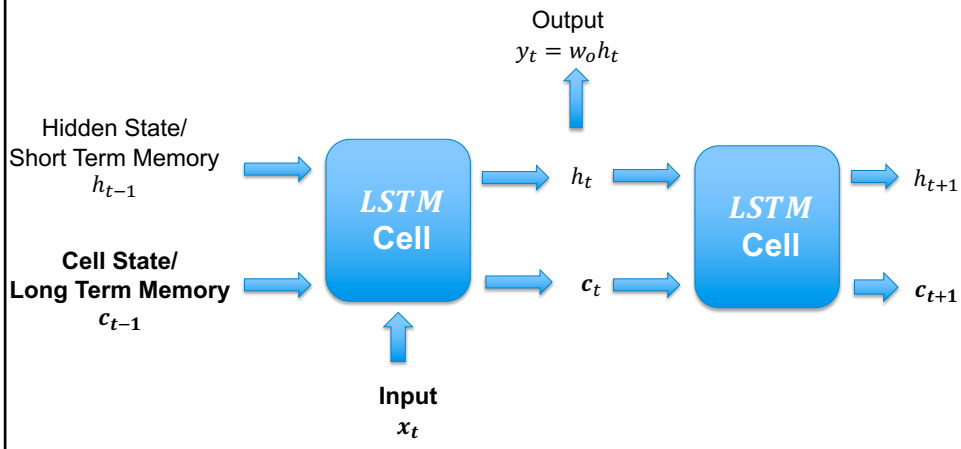


From Karpathy, Johnson and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

8

Imperial College London

The Basic LSTM Cell for time t



https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

9

Imperial College London

Detailed Calculations in an LSTM cell

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

"Input" Gate
"Forget" Gate
"Output" Gate

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W' \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

"internal" Cell State: $c_t = f \odot c_{t-1} + i \odot g$

Hidden State: $h_t = o \odot \tanh(c_t)$

Element-wise Multiplication

https://www.youtube.com/watch?v=6niqTuYFZLQ

10

Imperial College London

May 11th, 2020

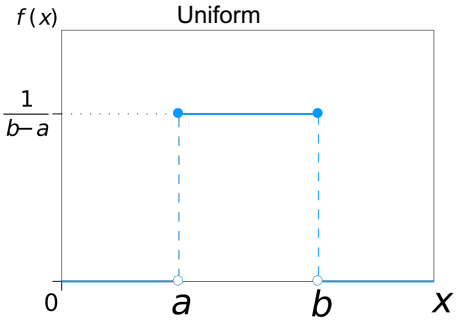
Probabilities for Deep Learning

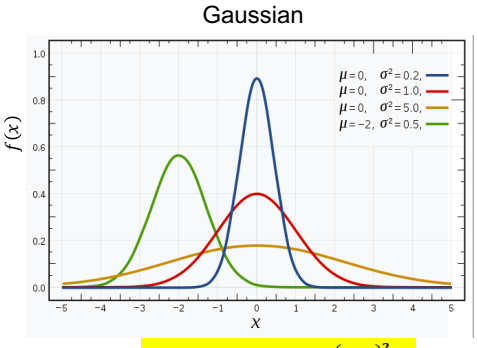
Olivier Dubrule and Navjot Kukreja

11

Imperial College London

The Uniform and the Normal (or Gaussian) pdfs





$f(x) = \frac{1}{b-a}$ *if* $x \in [a, b]$ and $f(x) = 0$ *if* $x \notin [a, b]$

$f(x)$ also written $\mathrm{U}(x; a, b)$

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

$f(x)$ also written $N(x; \mu, \sigma^2)$

https://en.wikipedia.org/wiki/Normal_distribution

12

Imperial College London

General Properties of a PDF $f(x)$

$$\mathrm{P}(a < X < b) = \int_a^b f(x)dx \qquad \int_{-\infty}^{+\infty} f(x)dx = 1$$

Mean $$E(X) = \mu = \int_{-\infty}^{+\infty} xf(x)dx$$

Variance $$Var(X) = \sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)dx$$
$$= E[(X-\mu)^2] = E(X^2) - \mu^2$$

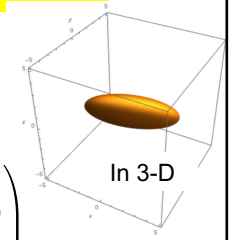
13

Imperial College London

Multivariate Normal pdf of a Random Vector$(X_1, \ldots X_n)$

$$f(x) = \frac{1}{(2\pi)^{n/2}} \frac{1}{\sqrt{\det(\Sigma)}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

$\boldsymbol{x}$ is the $n\times1$ vector $\quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$



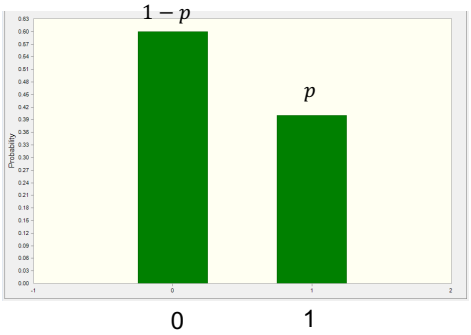
μ is the $n\times1$ expectation vector $\quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_n) \end{pmatrix}$

Σ is the $n\times n$ variance-covariance matrix $\quad \Sigma = \left(\left(Cov(X_i, X_j)\right)_{i,j=1,\ldots,n}\right)$

14

Imperial College London

Probability Distribution of a Bernouilli Random Variable



A Bernouilli variable takes the value 1 with probability p and 0 with probability $(1-p)$, and we can write that the probability that it is equal to x is:

$$b(x) = p^x(1-p)^{1-x}$$

15

Imperial College London

Independent and Identically Distributed (IID)

*In probability theory, a sequence or collection of random variables is independent and identically distributed (**i.i.d.** or **iid** or **IID**) if each random variable has the same probability distribution as the others and they all are mutually independent.*

When treating m samples from a training or test dataset (for instance a set of images), it is assumed they are IID.

16

Imperial College London

Problem Addressed by the Maximum Likelihood Method

Assume we have m *independent (IID)* data points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Each data point is $x^{(i)}$ is composed of n features.

We want to fit a multivariate (for instance multivariate Gaussian) pdf $p_\theta(x)$ of dimension n to these m samples.

Maximum likelihood consists of calculating the parameters θ such that the m samples maximize their likelihood.

But what is the likelihood? The likelihood is the probability to obtain the m sample values $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ assuming that $p_\theta(x)$ is the probability of x.

17

Imperial College London

Likelihood and Maximum Likelihood for a Dataset of Images

The pdf $p_\theta(x_1, x_2, \ldots\ldots, x_n)$ is parametrized by θ. If there is just one image $x^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, \ldots\ldots, x_n^{(1)}\right)$ in the dataset, its likelihood is defined as:

$$Likelihood\ of\ image\ x^{(1)} = p_\theta\left(x^{(1)}\right) = p_\theta\left(x_1^{(1)}, x_2^{(1)}, \ldots\ldots, x_n^{(1)}\right)$$

The Maximum Likelihood estimate θ_{ML} of θ is calculated as

$$\theta_{ML} = argmax\left(p_\theta\left(x^{(1)}\right)\right) \quad \text{or} \quad \theta_{ML} = argmax\left(\log p_\theta\left(x^{(1)}\right)\right)$$

If there are m *IID* images $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ in the dataset

$$\theta_{ML} = argmax\left(p_\theta\left(x^{(1)}\right)p_\theta\left(x^{(2)}\right)\ldots p_\theta\left(x^{(m)}\right)\right)$$

$$= argmax\left(\log\left(p_\theta\left(x^{(1)}\right)p_\theta\left(x^{(2)}\right)\ldots p_\theta\left(x^{(m)}\right)\right)\right) = \theta_{ML} = argmax\left(\sum_{i=1}^{m}\log p_\theta\left(x^{(i)}\right)\right)$$

18

Imperial College London

Maximum Likelihood for a Bernouilli Distribution

A Bernouilli distribution has just one parameter p

Suppose we have just one sample x_1 (which has the value 1 or 0)

Its likelihood is: $b(x_1) = p^{x_1}(1-p)^{1-x_1}$

Its log-likelihood is: $\log b(x_1) = x_1 \log p + (1-x_1)\log(1-p)$

If we have m samples x_i, their log-likelihood is:

$\sum_{i=1\ldots m}(x_i \log p + (1-x_i)\log(1-p))$ **(this is equal to minus the cross-entropy!)**

The maximization leads, unsurprisingly, to : $p_{ML} = \frac{1}{m}\sum_{i=1\ldots m} x_i$.
(p_{ML} is the proportion of samples equal to 1)

19

Imperial College London

Compare two pdfs: the Kullback-Leibler (KL) Divergence

For two pdfs $p(x)$ and $q(x)$:

$$D_{KL}(p\|q) = \int_{-\infty}^{+\infty} p(x)\log p(x)\,dx - \int_{-\infty}^{+\infty} p(x)\log q(x)\,dx$$

Two Fundamental Properties

- $D_{KL}(p\|q)$ is positive, and 0 if $p(x)$ and $q(x)$ identical
- Asymmetry: $D_{KL}(p\|q) \neq D_{KL}(q\|p)$

20

Imperial College London

KL Divergence versus Maximum Likelihood

If the Training Set consists of m *IID* data points $x^{(1)}, x^{(2)}, \dots, x^{(m)}$, the KL Divergence $D_{KL}(p_d \| p_\theta)$ between the experimental distribution p_d of the Training Set and any theoretical distribution p_θ is:

$$D_{KL}(p_d \| p_\theta) = \int_{-\infty}^{+\infty} p_d(x) \log p_d(x)\, dx - \int_{-\infty}^{+\infty} p_d(x) \log p_\theta(x)\, dx$$

Minimizing $D_{KL}(p_d \| p_\theta)$ in the parameters θ is equivalent to maximizing the second term:

$$\int_{-\infty}^{+\infty} p_d(x) \log p_\theta(x)\, dx$$

But this is the expression of the expectation of $\log p_\theta(x)$calculated over the Training Set! Hence

$$\theta = argmax\left(\frac{1}{m}\sum_{i=1}^{m} \log p_\theta\left(x^{(i)}\right)\right)$$

Minimizing the KL Divergence is equivalent to maximizing the Likelihood!