Imperial College London

ACSE 2020

Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTM)

Olivier Dubrule, Lluís Guasch

Objectives

- Basic Recurrent Neural Networks (RNNs) model
- Introduction to Long Short-Term Memory (LSTM) Networks

RNNs

Examples of sequential data:

- Text considered as a sequence of words or characters
- Continuous parameter which is a function of time (ie stock price)
- Sequence of images in a video-clip
- Sequence of labels in a Genome sequence
- Vertical sequences of lithologies in the subsurface
- … plus many others

RNNs

Applications of RNNs to sequential data:

- Text Generation “in the style of”
- Provide “Sentiment analysis” on a piece of text
- Automatic “Speech-to-Text” as in MS Teams
- Automatic Foreign language translation
- Prediction of stock price on the basis of historical data
- Label sequence of images in a video-clip

RNNs

Quick reminder of a feed-forward network

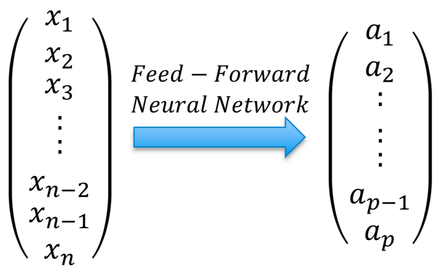
Fixed-sized inputs mapped to fixed-sized outputs don't provide enough flexibility to work on sequential data problems

A Simple Way to See Supervised Neural Networks

Suppose we have m pairs of data.
Each pair is composed of a vector of dimension n and a vector of dimension p (the labels).

A neural network is simply a function that maps any vector of dimension n into a (discrete or continuous) vector of dimension p.

In order to calculate the parameters of this function, we train the parameters of the neural network by back-propagation using the m pairs of data as Training Set.



Input Vector

Output Labels

RNNs

Quick reminder of a feed-forward network

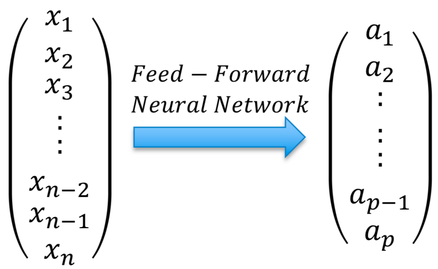
Fixed-sized inputs mapped to fixed-sized outputs don't provide enough flexibility to work on sequential data problems

A Simple Way to See Supervised Neural Networks

Suppose we have m pairs of data.
Each pair is composed of a vector of dimension n and a vector of dimension p (the labels).

A neural network is simply a function that maps any vector of dimension n into a (discrete or continuous) vector of dimension p.

In order to calculate the parameters of this function, we train the parameters of the neural network by back-propagation using the m pairs of data as Training Set.

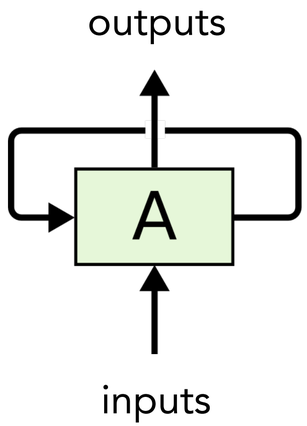


Input Vector

Output Labels

RNNs

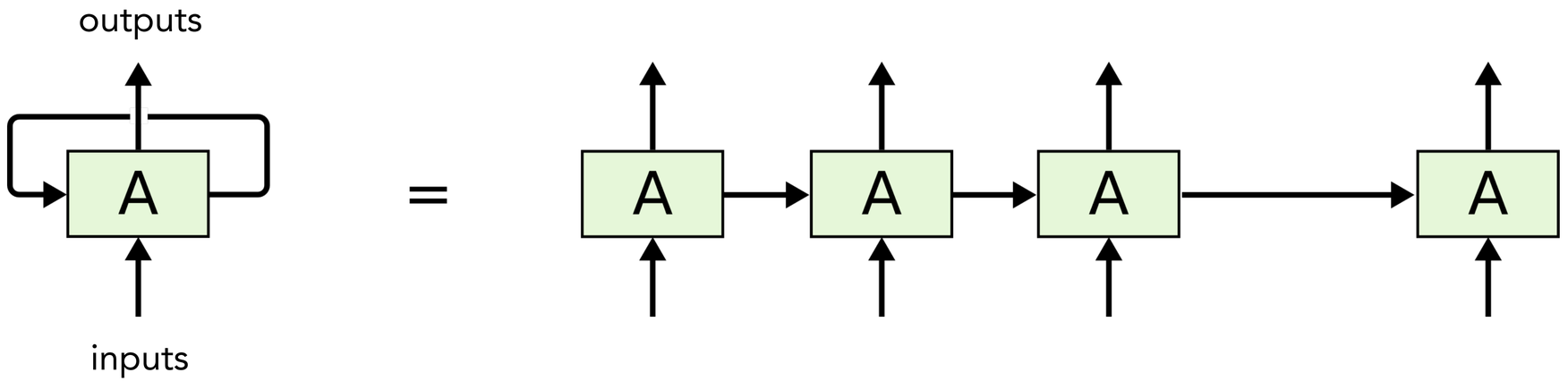
Basic structure of RNNs:



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs

Basic structure of RNNs:



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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs

Simple (unrealistic) example:

Given a string: `Hello world!`

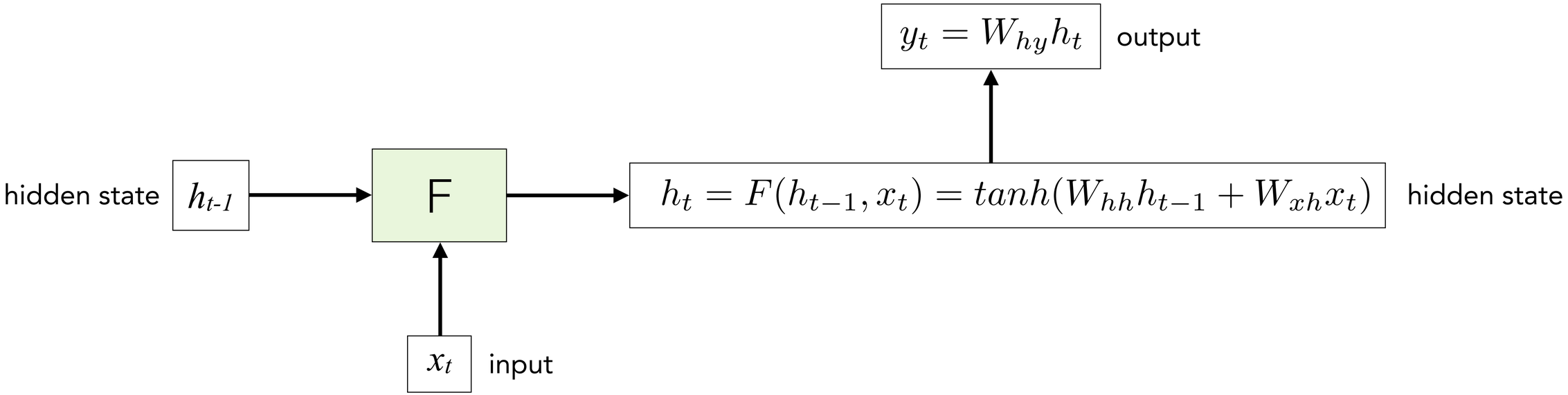
We identify the dictionary associated with the input (ignoring capitalisation) contains 9 characters: `(d,e,h,l,o,r,w, ,!)`

And we **hot-encode** the dictionary:

$$h = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad ! = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \cdots$$

RNNs

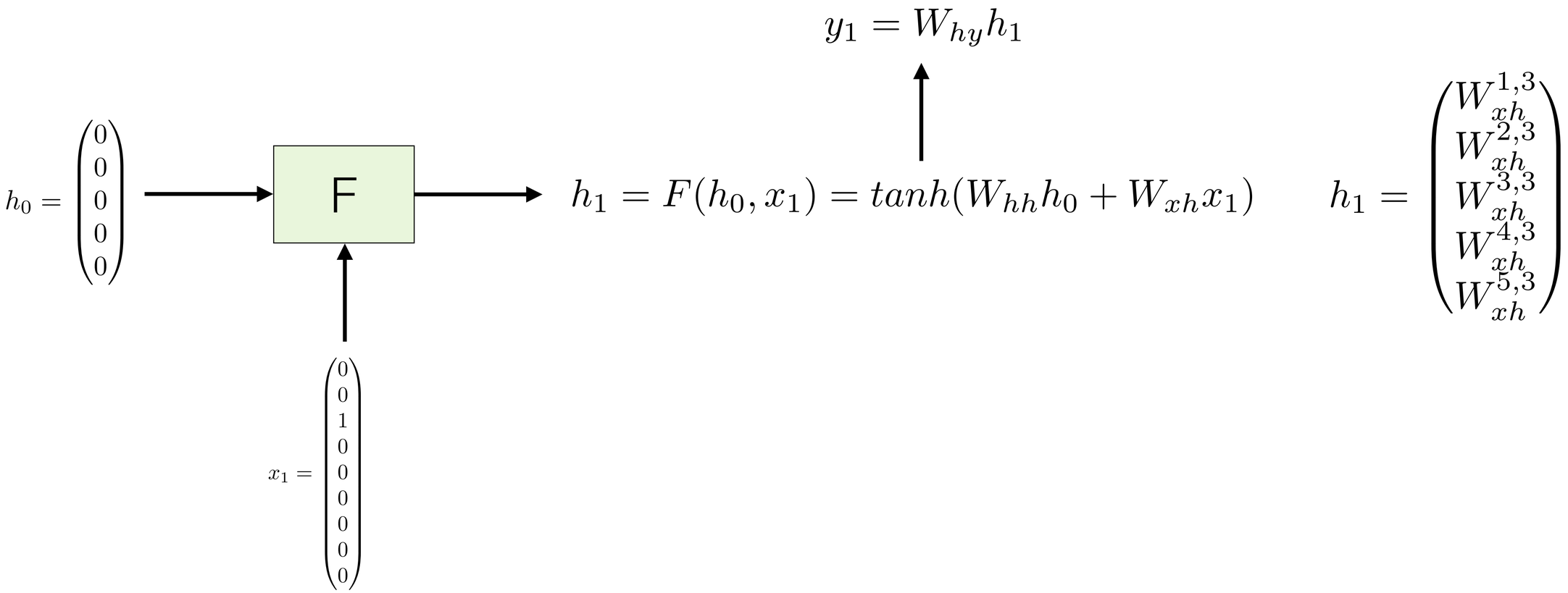
How does a basic RNN cell work?



RNNs

How does training work? Back to our simple example:

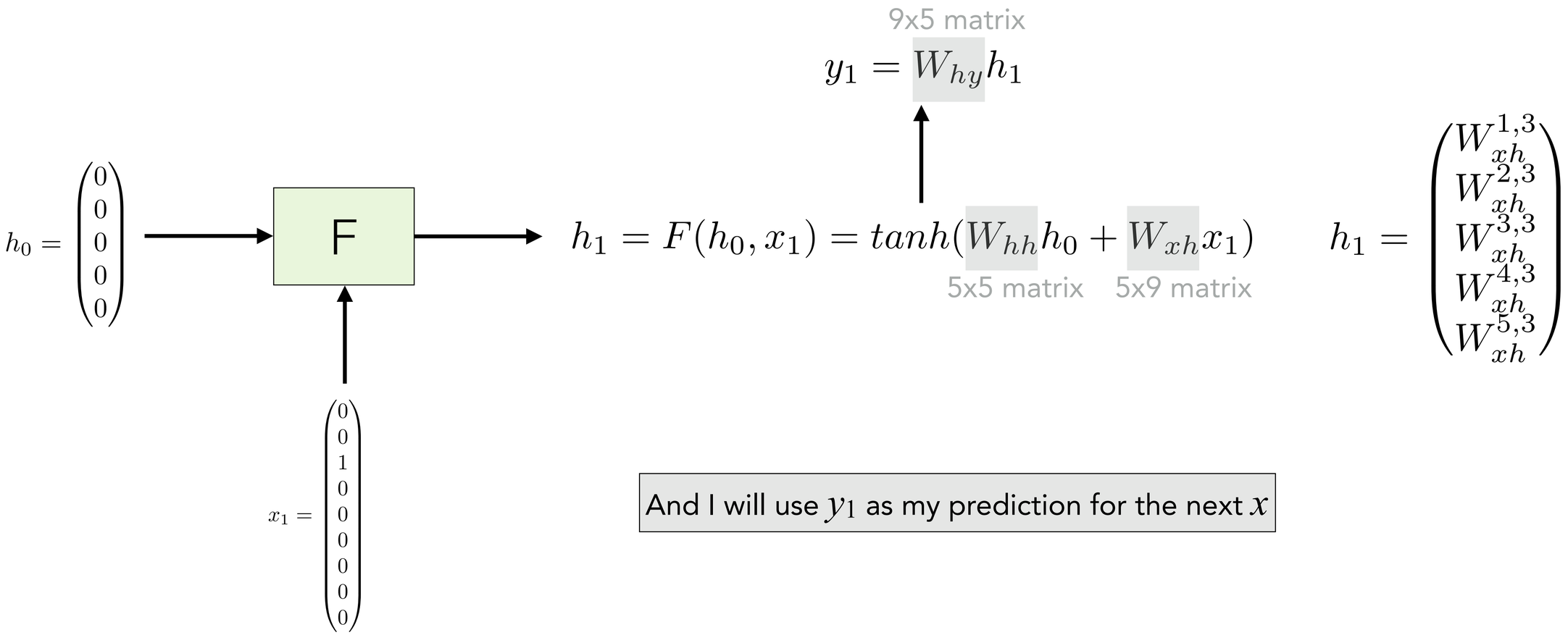
We define the size of our hidden state tensor h (hyperparameter) to be 5, and start with our first input character ('h'):



RNNs

How does training work?

We define the size of our hidden state tensor h (hyperparameter) to be 5, and start with our first input character ('h'):



And I will use y_1 as my prediction for the next x

RNNs

How does training work? The loss function:

$$y_1 = W_{hy} h_1 \longrightarrow \textit{Cross-entropy}\ (\ \textit{Softmax}\ (y_1)\ ,\ \ x_2\ )$$

Loss function corresponding to the first predicted letter

remember: hot-encoded

RNNs

How does training work? The loss function:

$$y_1 = W_{hy} h_1 \longrightarrow \textit{Cross-entropy}\,(\,\textit{Softmax}\,(y_1)\,,\; x_2\,)$$

Loss function corresponding to the first predicted letter

remember: hot-encoded

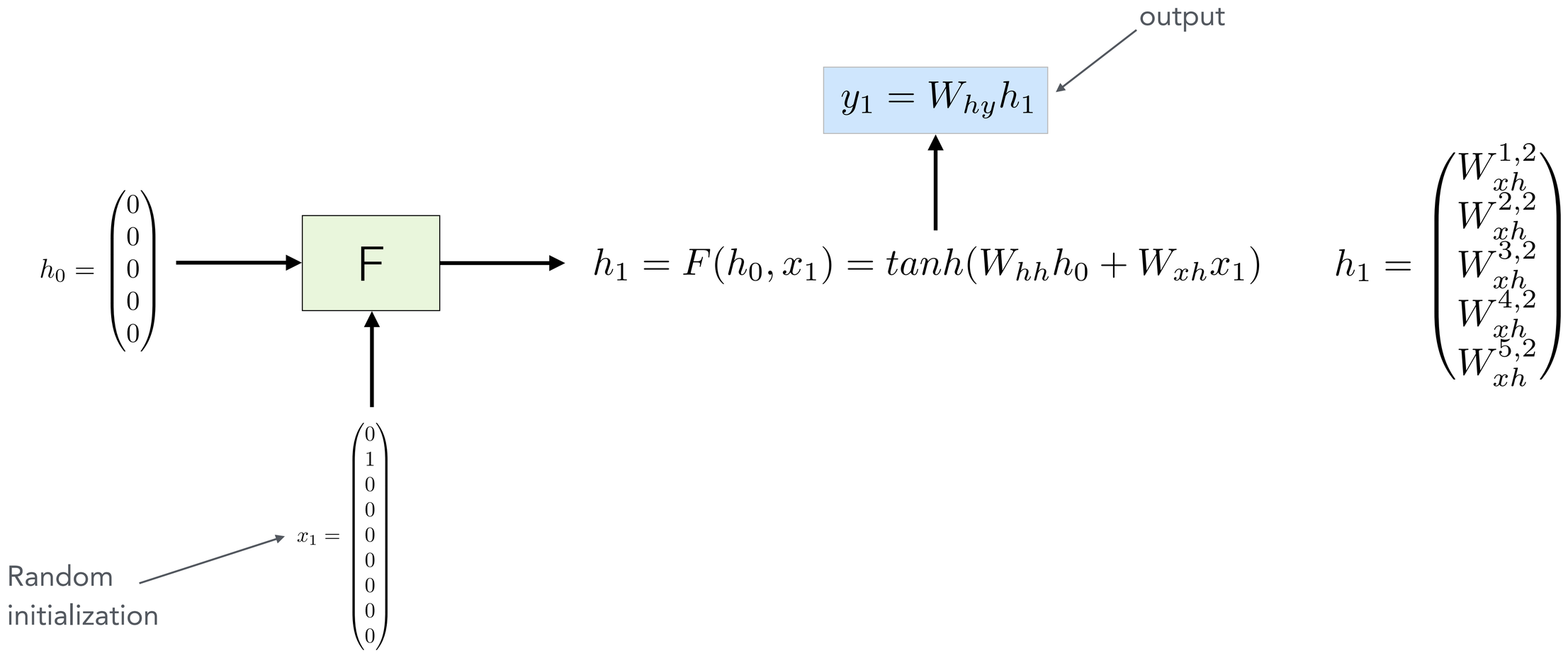
We repeat the process for all the characters in our input and add their loss contributions

And then use gradient descent to train our weights in the matrices

RNNs

Text generation using a trained network

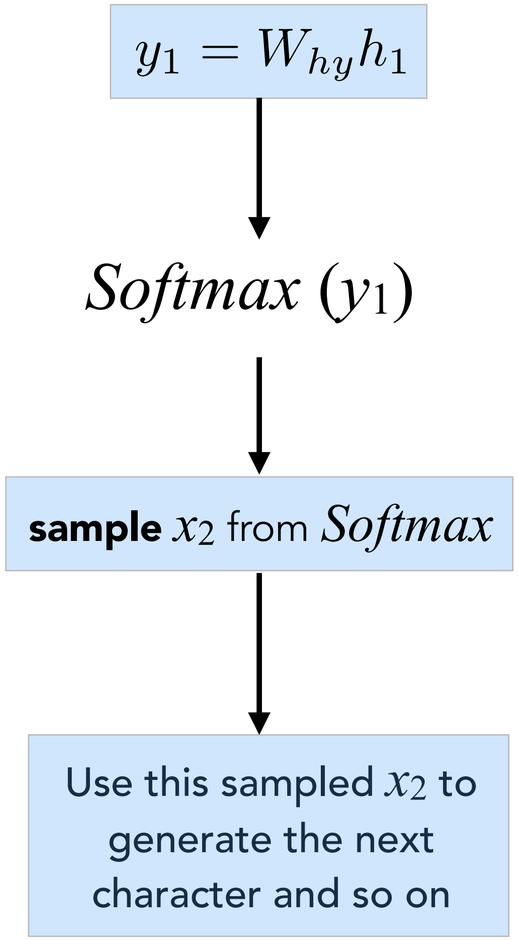
Now we are going to use our trained RNN to generate some text:



RNNs

Text generation using a trained network

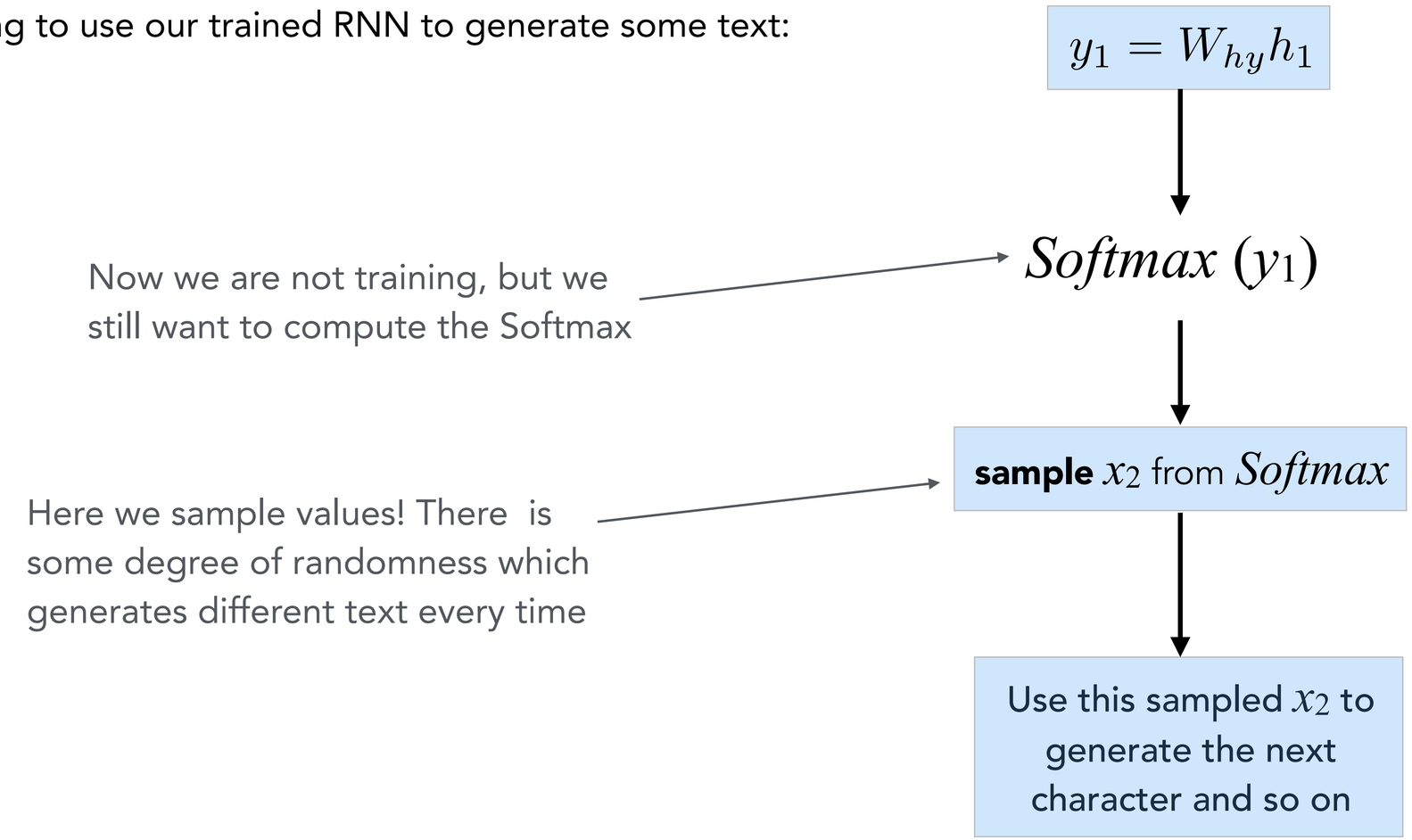
Now we are going to use our trained RNN to generate some text:



RNNs

Text generation using a trained network

Now we are going to use our trained RNN to generate some text:



RNNs

Recap of training an RNN

1. Take first character x_1, combine with current hidden state h_0 (its dimension 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 the sequence.
2. Take this second character as input to repeat the step 1. and calculate loss function again using the value of the third character of the sequence, and so on until we reach the end of the sequence.
3. Add up all the above loss functions.
4. Do back-propagation to calculate gradients according to each trainable parameter.
5. Modify parameters by gradient descent
6. Move to next sequence of p characters, until end of Training Set is reached. Size of p is a hyperparameter too!

RNNs

Recap of generating text with an RNN

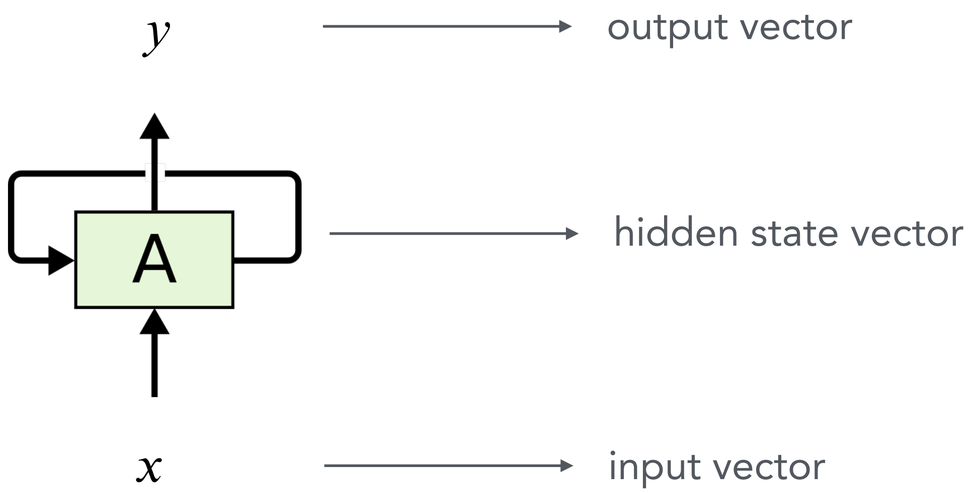
Generate a new sequence of n characters:

1. Randomly sample first character x_1
2. Combine with initial null state vector value h_0 to derive new vector h_1
3. Calculate associated outputvector y_1 of size equal to vocabulary size
4. Transform y_1 into Softmax vector
5. Sample new character x_2 based on Softmax probability
6. Continue until the required number of characters have been generated.

If we were using words instead of characters, the dictionary would be much bigger

RNNs

Synthetic notation for RNNs:



RNNs

Simple implementation in python:

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

And after a few hours of training, we start to get some decent writing:

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs

Analysis of the role of h

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

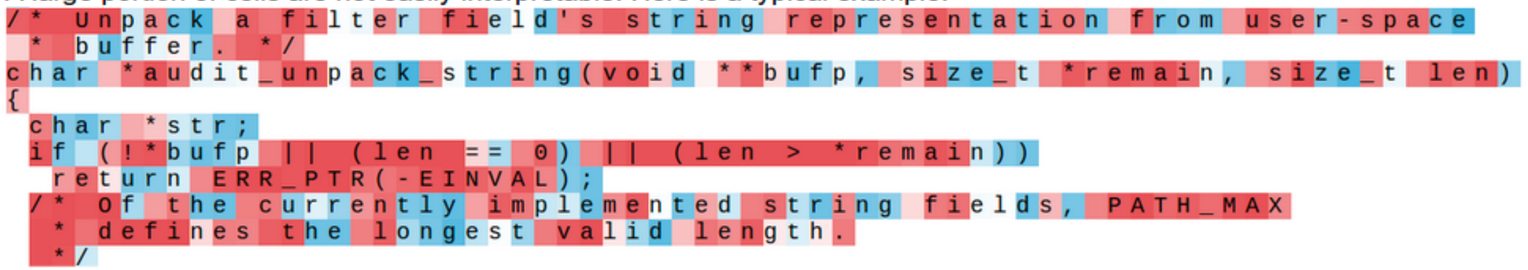
```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs

Analysis of the role of h

A large portion of cells are not easily interpretable. Here is a typical example:

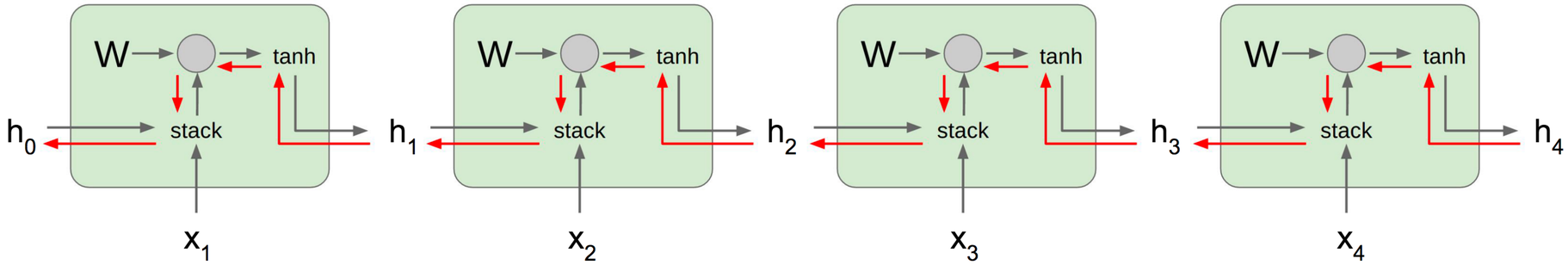


In their study, Karpathy et al found that only ~5% of the h positions were interpretable

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs

Backpropagation through RNNs



Computing gradients involves many factors of **W** and repeated **tanh** activations and that creates problems:

- **Exploding gradients**: can be solved by gradient clipping
- **Vanishing gradients**: requires changes in the RNN architecture

the derivative of tanh is in the range (0, 1) and we apply it multiple times using the chain rule in backprop

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

RNNs

Issues with RNNs

- Exploding and vanishing gradients
- Lack of long term memory

RNNs

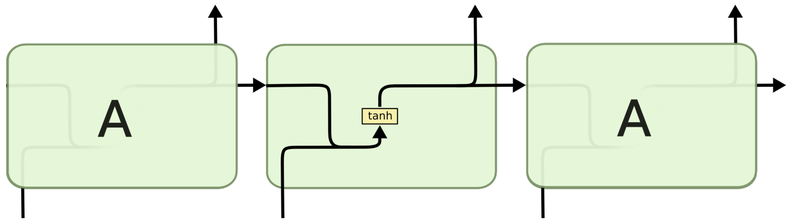
Issues with RNNs

- Exploding and vanishing gradients
- Lack of long term memory

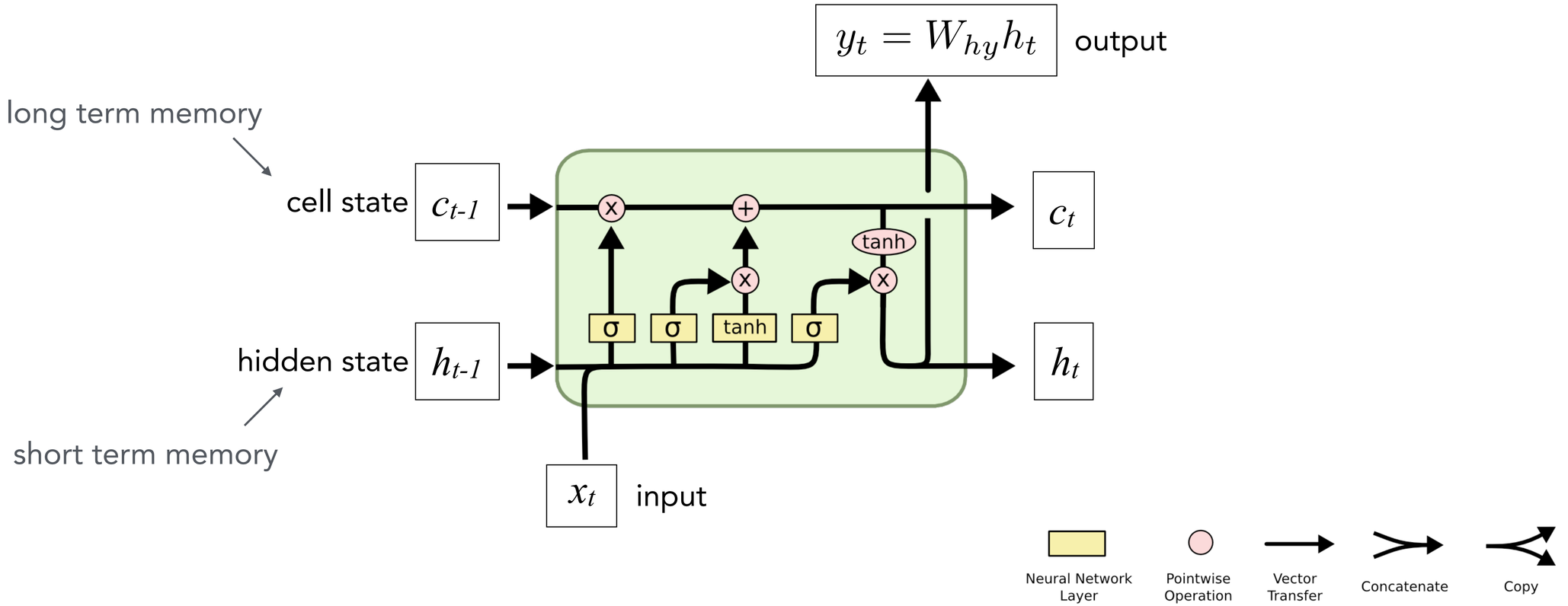
LSTMs to the rescue!

LSTMs

Basic structure of an LSTM:



RNN structure



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

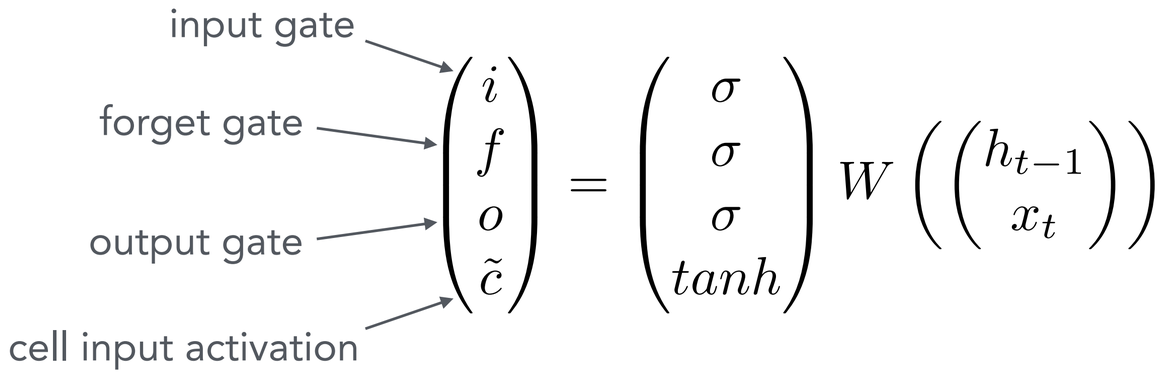
LSTMs

Basic structure of an LSTM:

vanilla RNN:

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

LSTM:



$$\begin{pmatrix}i\\ f\\ o\\ \tilde{c}\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ tanh\end{pmatrix} W\left(\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix}\right)$$

$$c_t = f \odot c_{t-1} + i \odot \tilde{c}$$

$$h_t = o \odot tanh(c_t)$$

Hadamard product (element-wise multiplication)

LSTMs

Basic structure of an LSTM:

W matrix structure:

adding bias

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

LSTM:

input gate

forget gate

output gate

cell input activation

$$
\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \left(\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
$$

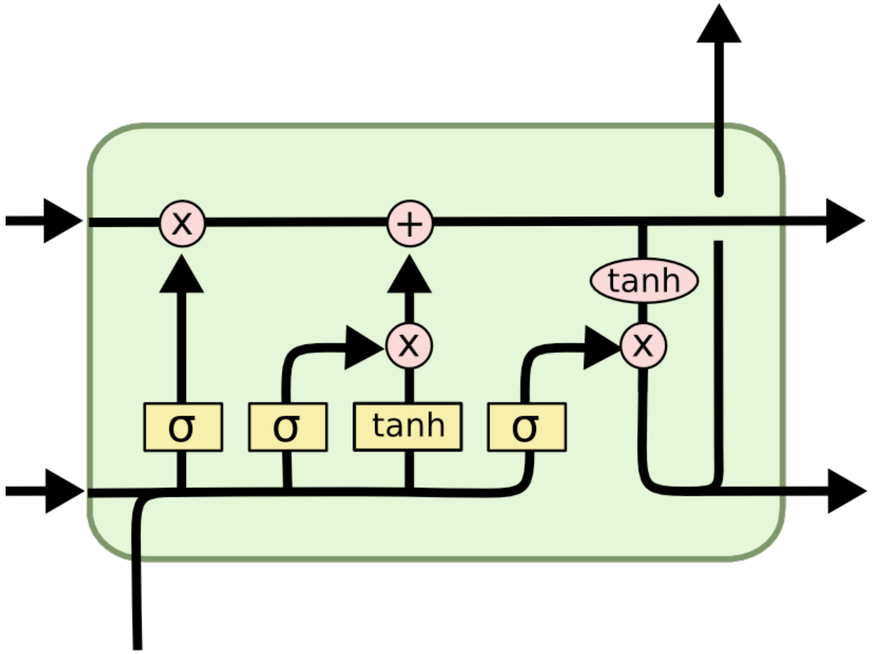
$$
c_t = f \odot c_{t-1} + i \odot \tilde{c}
$$

$$
h_t = o \odot tanh(c_t)
$$

Hadamard product (element-wise multiplication)

LSTMs

Basic structure of an LSTM:



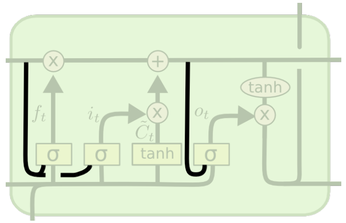
$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \left(\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

$$c_t = f \odot c_{t-1} + i \odot \tilde{c}$$

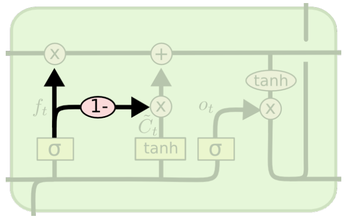
$$h_t = o \odot tanh(c_t)$$

LSTMs

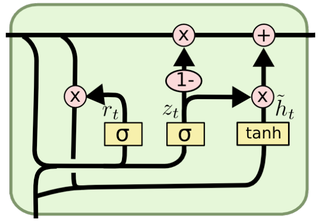
Variations on LSTMs:



peep-hole connections

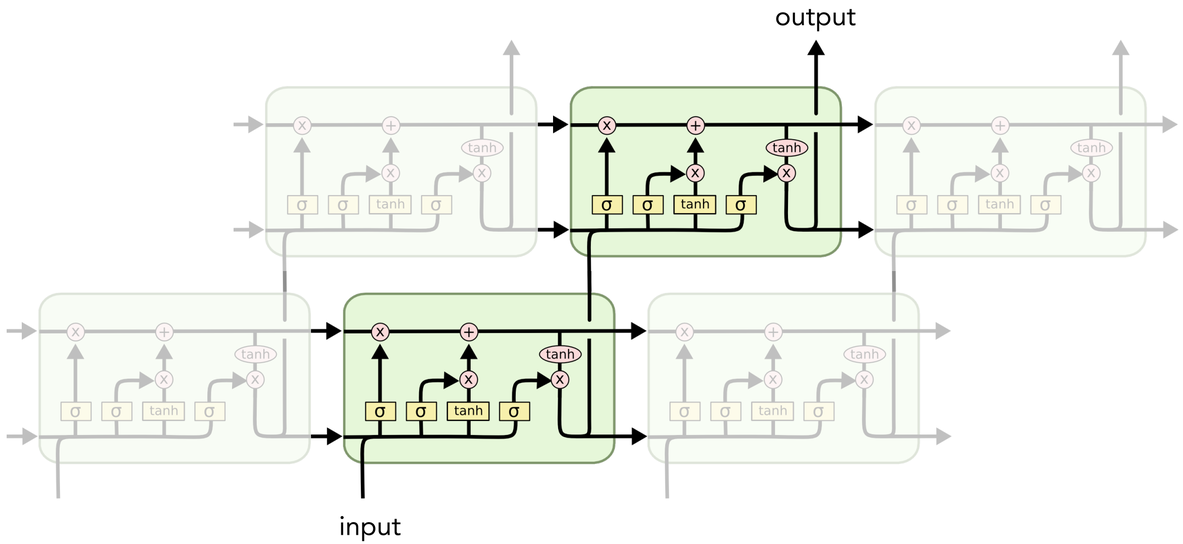


coupled forget and input gates



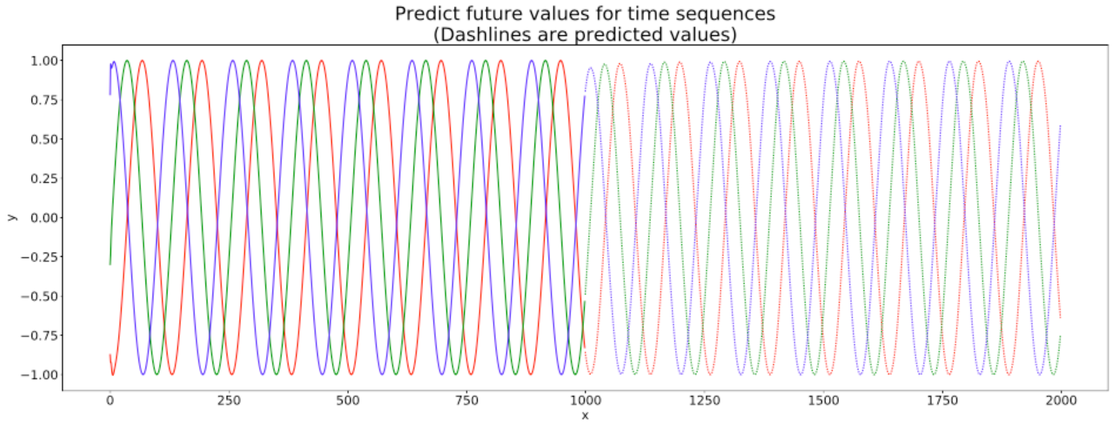
GRU (Gated Recurrent Unit), combines forget and input gates into a single update gate

stacked LSTMs



LSTMs

Examples of application:



https://github.com/pytorch/examples/tree/master/time_sequence_prediction

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Conclusions on RNNs and LSTMs

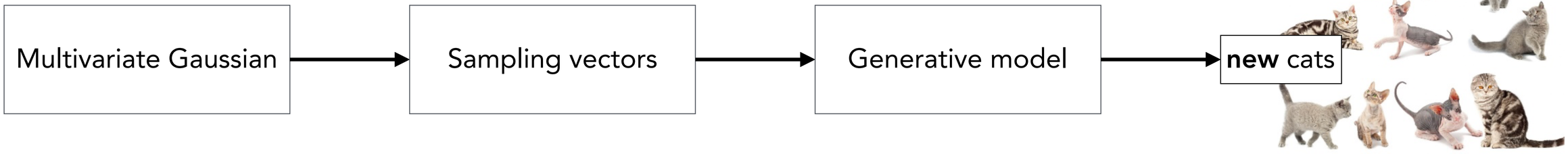
- Recurrent Neural Networks (RNNs) are the basic approach for sequential data.
- RNNs suffer from exploding and vanishing gradients and only preserve short-term memory.
- Long-Term Short-Term Memory (LSTM) networks address these two issues.

Probabilities for Deep Learning

Objective

Introduce basic probability concepts required to understand Generative Models

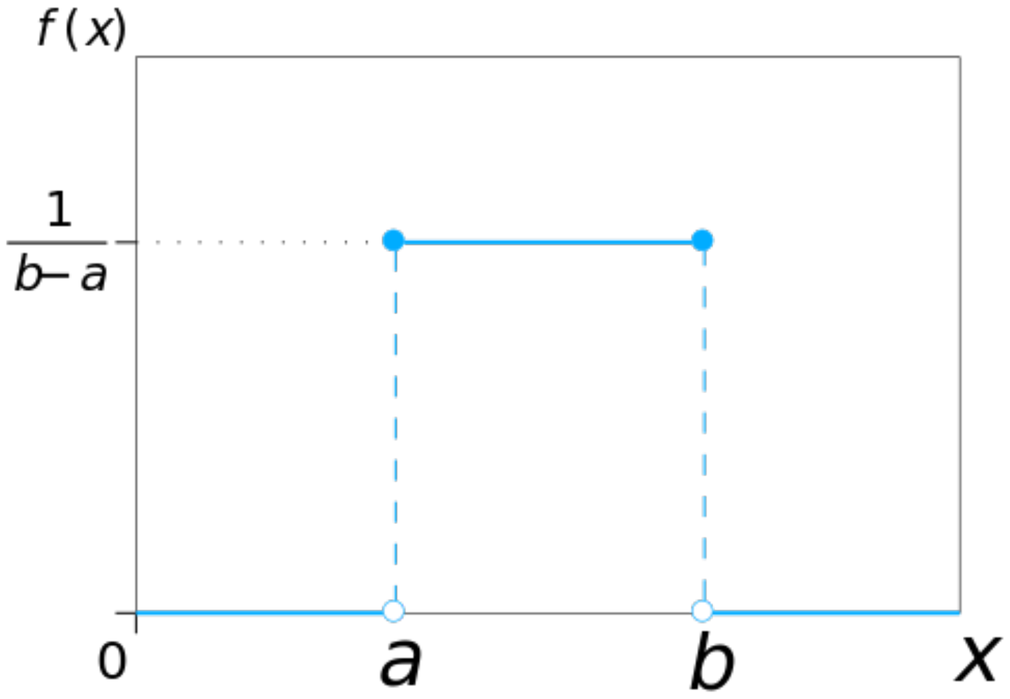
Given a training set with lots of cats, how can I generate **new** cats?



Probabilities for Deep Learning

1. **Uniform, Gaussian, and Bernouilli distributions**
2. Maximum likelihood
3. Comparing probability density functions

Uniform distribution (continuous variables)



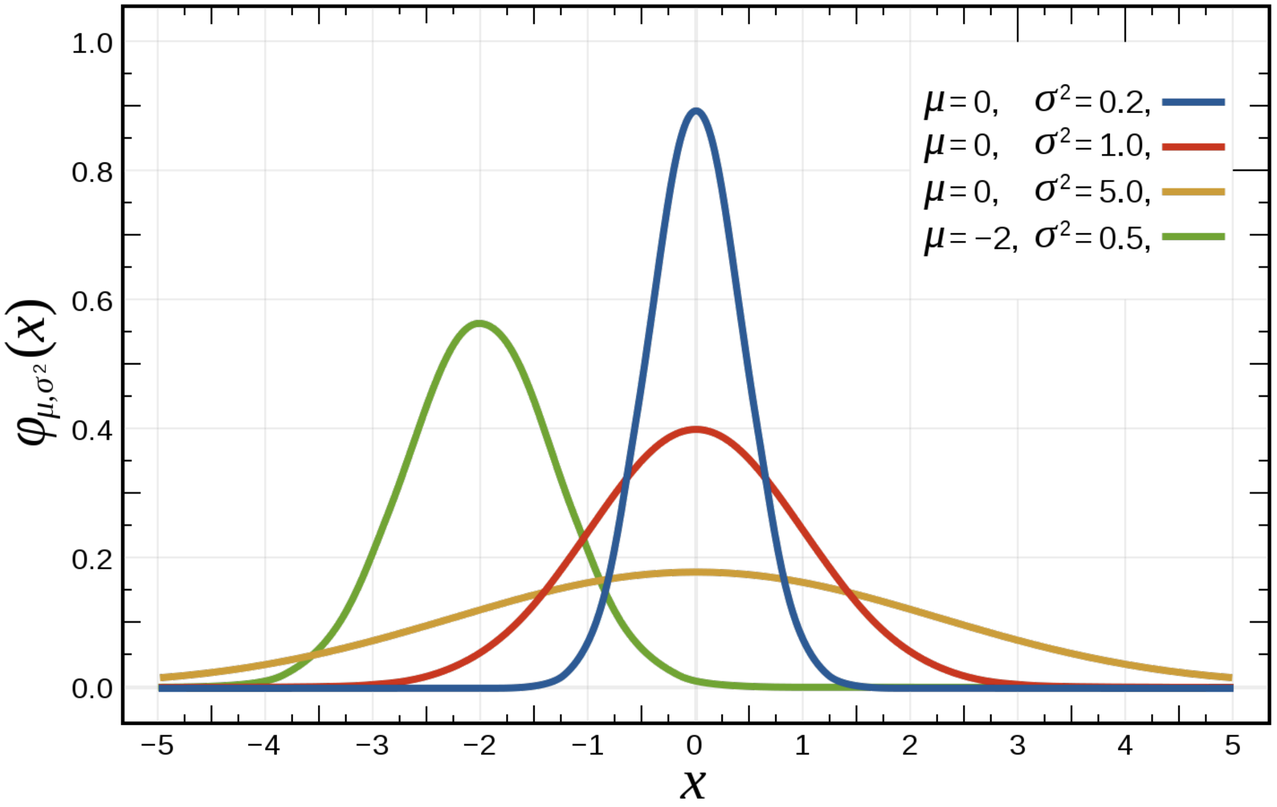
so that the area under the curve is 1

$$f(x) = \frac{1}{b-a} \qquad \text{if} \qquad x \in [a, b]$$

$$f(x) = 0 \qquad \text{if} \qquad x \notin [a, b]$$

$f(x)$ is also written $U(x; a, b)$

Gaussian or normal distribution (continuous variables)



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

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

Probability density function (pdf)

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

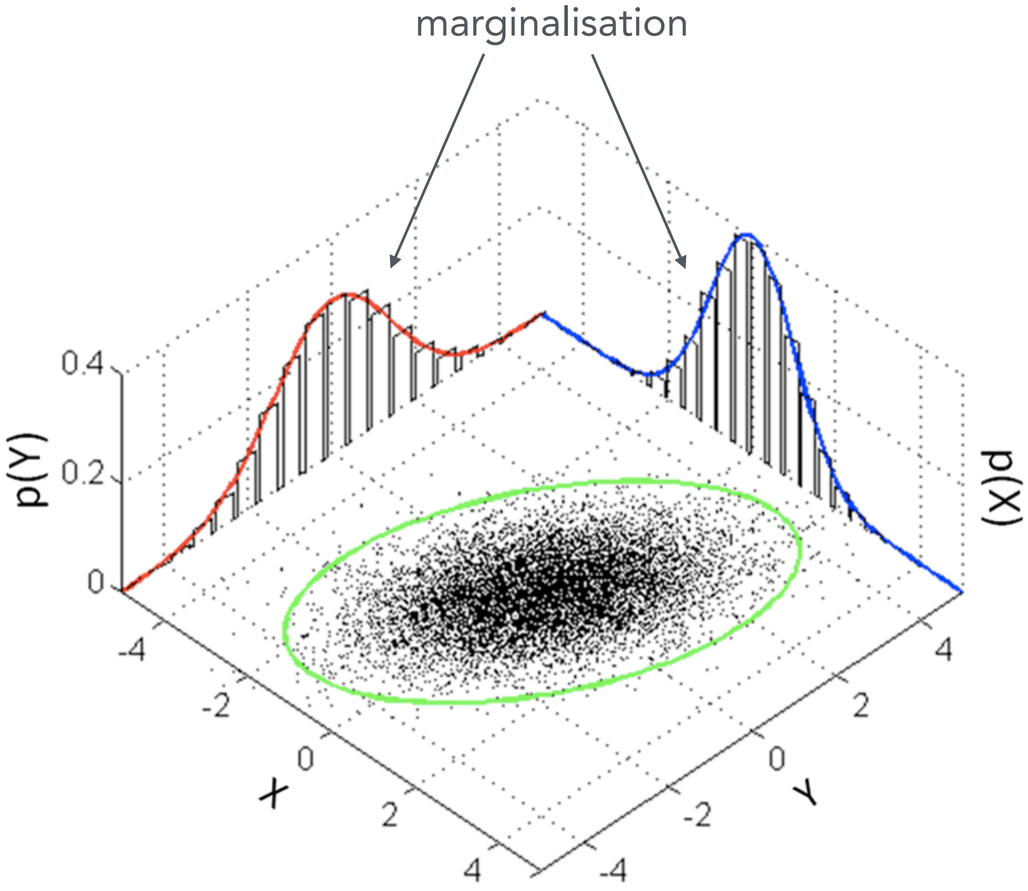
Expectation (or mean)

$$E(x) = \mu = \int_{-\infty}^{\infty} x f(x)\,dx$$

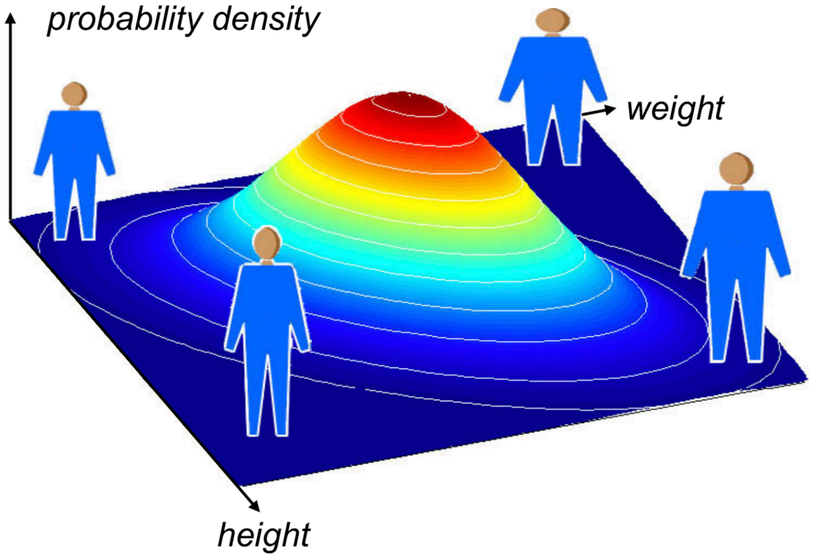
Variance

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

Multivariate Gaussian distribution



for example:



Covariance and Correlation

Assume we have two random variables:

X_1 has mean μ_1 and standard deviation σ_1

X_2 has mean μ_2 and standard deviation σ_2

The covariance is defined as:

$$Cov(X_1, X_2) = E[(X_1 - \mu_1)(X_2 - \mu_2)]$$

The correlation coefficient is defined as:

$$\rho = \frac{Cov(X_1, X_2)}{\sigma_1, \sigma_2}$$

properties of ρ: $-1 \leq \rho \leq 1$

$Cov(X_1, X_2) = 0$ or $\rho = 0$ ⟶ X_1 and X_2 are uncorrelated

$\rho = -1$ (or +1) means perfect linear relation between them

Examples of bivariate normal distributions

https://distill.pub/2019/visual-exploration-gaussian-processes/

Bivariate normal density

If μ is the 2x1 mean vector and Σ is the 2x2 "Variance-Covariance" Matrix of the Bivariate Gaussian : (X_1, X_2)

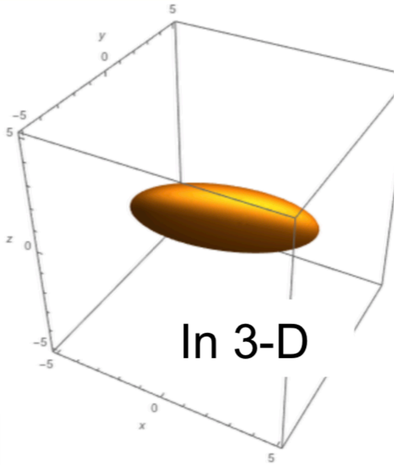
$$\Sigma = \begin{pmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_1, X_2) & Var(X_2) \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

$$f(x_1, x_2) = \frac{1}{2\pi} \frac{1}{\sqrt{\det(\Sigma)}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right]$$

Multivariate normal pdf

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

$\boldsymbol{x}$ is the $n \times 1$ vector $\quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$



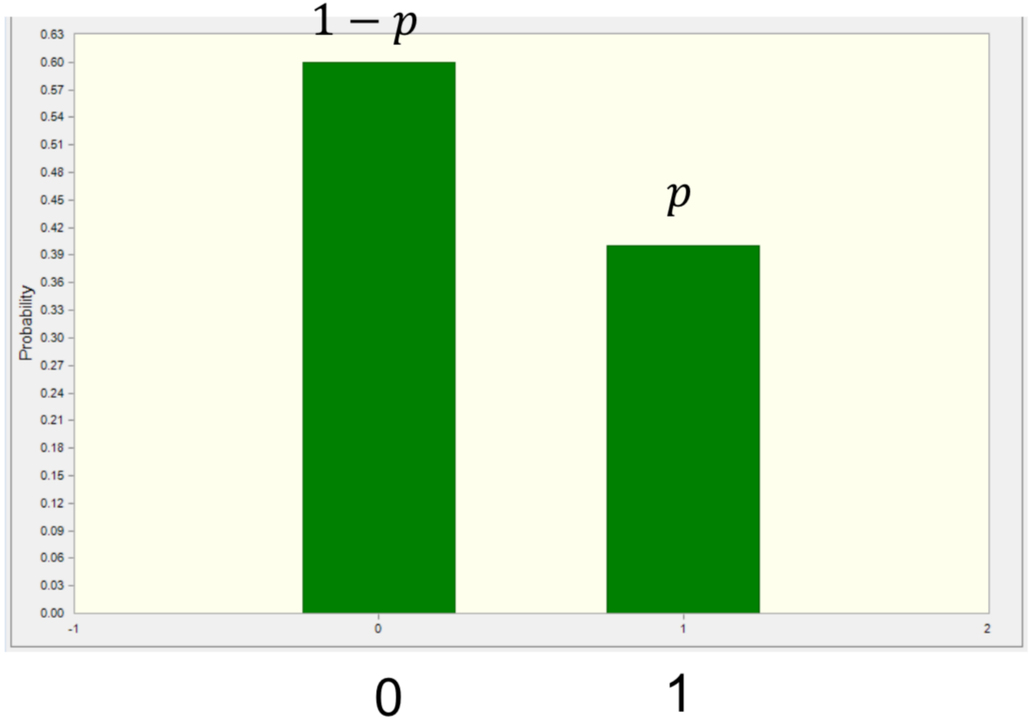
In 3-D

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

Probability distribution of a Bernoulli Random Variable (discrete)

For discrete randome variables, we use the term pmf (probability mass function)



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:

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

https://seeing-theory.brown.edu/

IID: Independent and Identically Distributed

*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.

Are augmented datasets IID?

Probabilities for Deep Learning

1. Uniform, Gaussian, and Bernouilli distributions
2. **Maximum likelihood**
3. Comparing probability density functions

Problem addressed by Maximum Likelihood Estimation

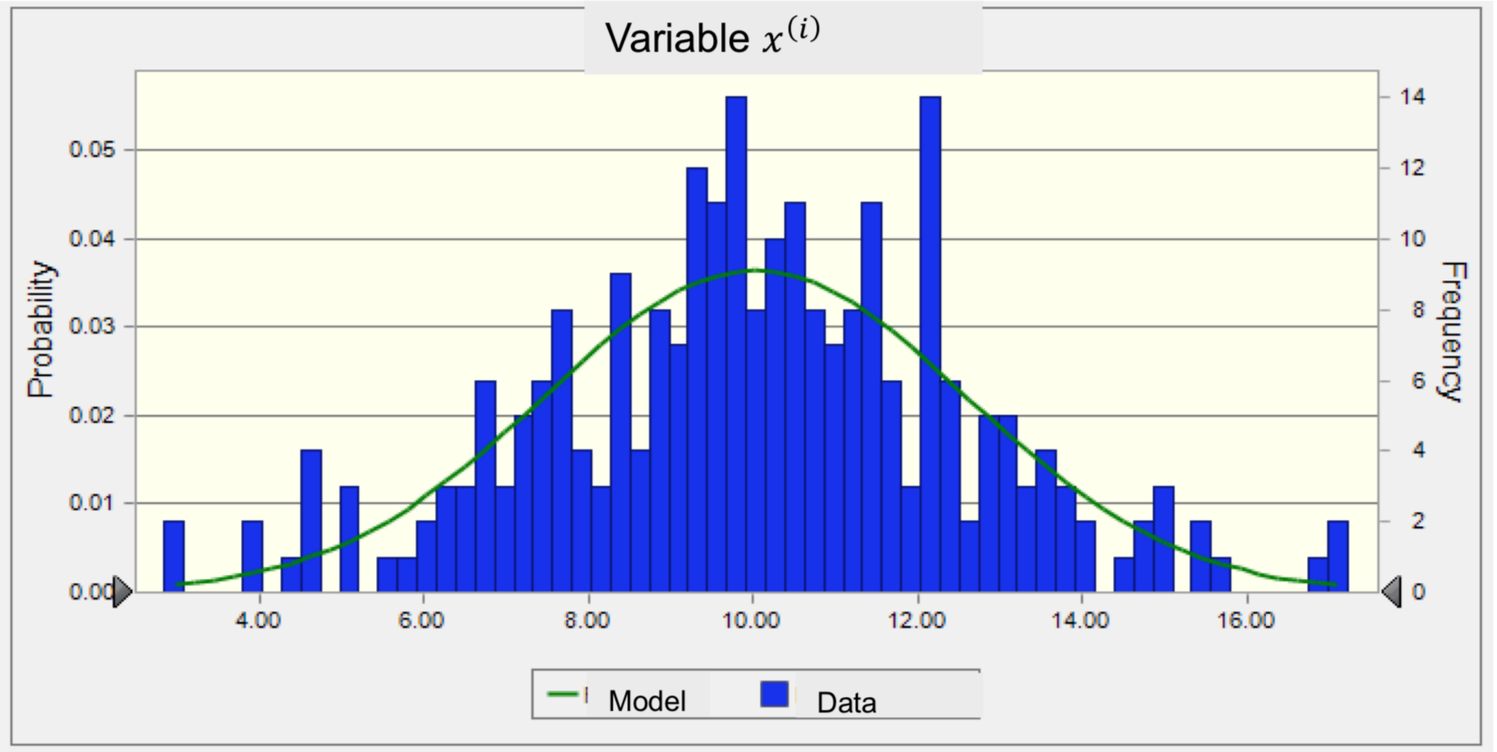
Number of IID Samples

$m = 250$

Number of Features

$n = 1$

How to calculate the normal distribution that best fits these data?



Problem addressed by Maximum Likelihood Estimation

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.

Example of likelihood calculation for a normal pdf

Assume we have 4 *independent (IID)* data points $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$ with:

$$x^{(1)} = -1,\ x^{(2)} = 0,\ x^{(3)} = 1,\ x^{(4)} = 2$$

Here each data point $x^{(i)}$ is composed of just $n = 1$ feature. We want to fit a Normal distribution $N(x; \mu, \sigma^2)$ to these four data points.

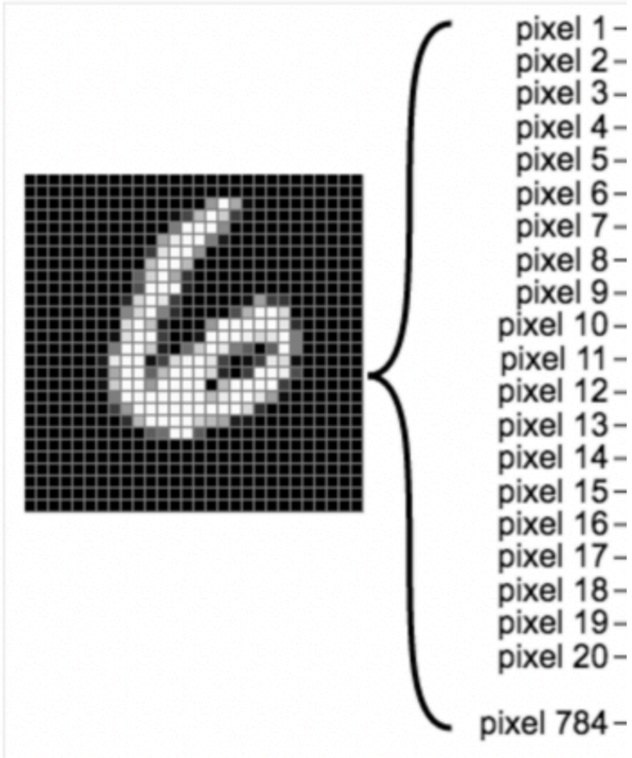
The likelihood of $x^{(1)}$ is: the value of $N(x; \mu, \sigma^2)$ for $x = x^{(1)} = -1$:

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

And the likelihood of the IID sequence $(x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)})$ is:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-1-\mu}{\sigma}\right)^2} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{0-\mu}{\sigma}\right)^2} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{1-\mu}{\sigma}\right)^2} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{2-\mu}{\sigma}\right)^2}$$

Samples can be images (like we saw in MNIST)



Multivariate Gaussian distribution with 784 dimensions

Likelihood and Maximum Likelihood

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

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

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

Maximum Likelihood in the univariate Gaussian case

Goal : fit a Gaussian pdf to a dataset

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

Here the θ parameters are μ and σ

We have $p_\theta(x) = N(x;\mu,\sigma^2)$

Hence the log-Likelihood $\sum_{i=1}^{m}\left(\log p_\theta\left(x^{(i)}\right)\right) = -\frac{1}{2}\sum_{i=1}^{m}\left(\frac{x^{(i)}-\mu}{\sigma}\right)^2 - m\log\sigma - \frac{m}{2}\log 2\pi$

The Maximum Likelihood Estimate for a Gaussian leads to the L2 norm!

Exercise

We have m IID values of real numbers $(x_i)_{i=1...m}$

We want to calculate the parameters of a normal distribution $N(x; \mu, \sigma^2)$ that best fits these *m* values.

What is the maximum likelihood estimate μ_{ML} of μ?

What is the maximum likelihood estimate σ^2_{ML} of σ^2?

Exercise

We have m IID values of real numbers $(x_i)_{i=1...m}$

We want to calculate the parameters of a normal distribution $N(x; \mu, \sigma^2)$ that best fits these *m* values.

What is the maximum likelihood estimate μ_{ML} of μ?

What is the maximum likelihood estimate σ^2_{ML} of σ^2?

$$\mu_{ML} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma^2_{ML} = \frac{1}{m}\sum_{i=1}^{m} \left(x^{(i)} - \mu\right)^2$$

Maximum Likelihood for a Bernoulli 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: $\log b(x_1) = x_1 \log p + (1-x_1)\log(1-p)$

Maximum Likelihood for a Bernoulli 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 : $\boldsymbol{p_{ML} = \frac{1}{m}\sum_{i=1\ldots m} x_i}$.

(p_{ML} is the proportion of samples equal to 1)

Probabilities for Deep Learning

1. Uniform, Gaussian, and Bernouilli distributions
2. Maximum likelihood
3. **Comparing probability density functions**

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: $D_{KL}(p\|q) \neq D_{KL}(q\|p)$

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

KL Divergence between $f = N(x; \mu_1, \sigma_1^2)$ and $g = N(x; \mu_2, \sigma_2^2)$

$$KL(f, g) = -\frac{1}{2} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} + \log \frac{\sigma_2}{\sigma_1}$$

(see Exercise 3 for calculation of KL Divergence for Gaussians and Multi-Gaussians)

KL divergence and Maximum Likelihood

If the Training Set consists of m *IID* data points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$, 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!

Conclusions on Probabilities

- The basic pdfs used in Deep Learning are Bernouilli for discrete variables and (Multivariate) Gaussian or Uniform for continuous variables.
- Maximum Likelihood is a key approach in Deep Learning and applying it to Bernouilli and (Multivariate) Gaussian random variables leads respectively to the minimization of the cross-entropy for classification or the L2 norm for regression.
- The Kullback-Leibner (KL) Divergence is used to calculate the dissimilarity between two pdfs. Minimizing it allows one pdf to be adjusted to another.