Hochschule Konstanz
Fakultät Informatik

Oliver Dürr

Short course on deep learning
Block 4

Southern Taiwan University of Science and Technology

This course is a short version of dl_course_2022 I created with Beate Sick.

Learning Objectives for today: looking under the hood

- Get an understanding of
  - Computational Graph
  - Backpropagation in Computational Graph
  - Maximum Likelihood principle for neural networks

Computational Graph

Looking under the hood of tf / Keras

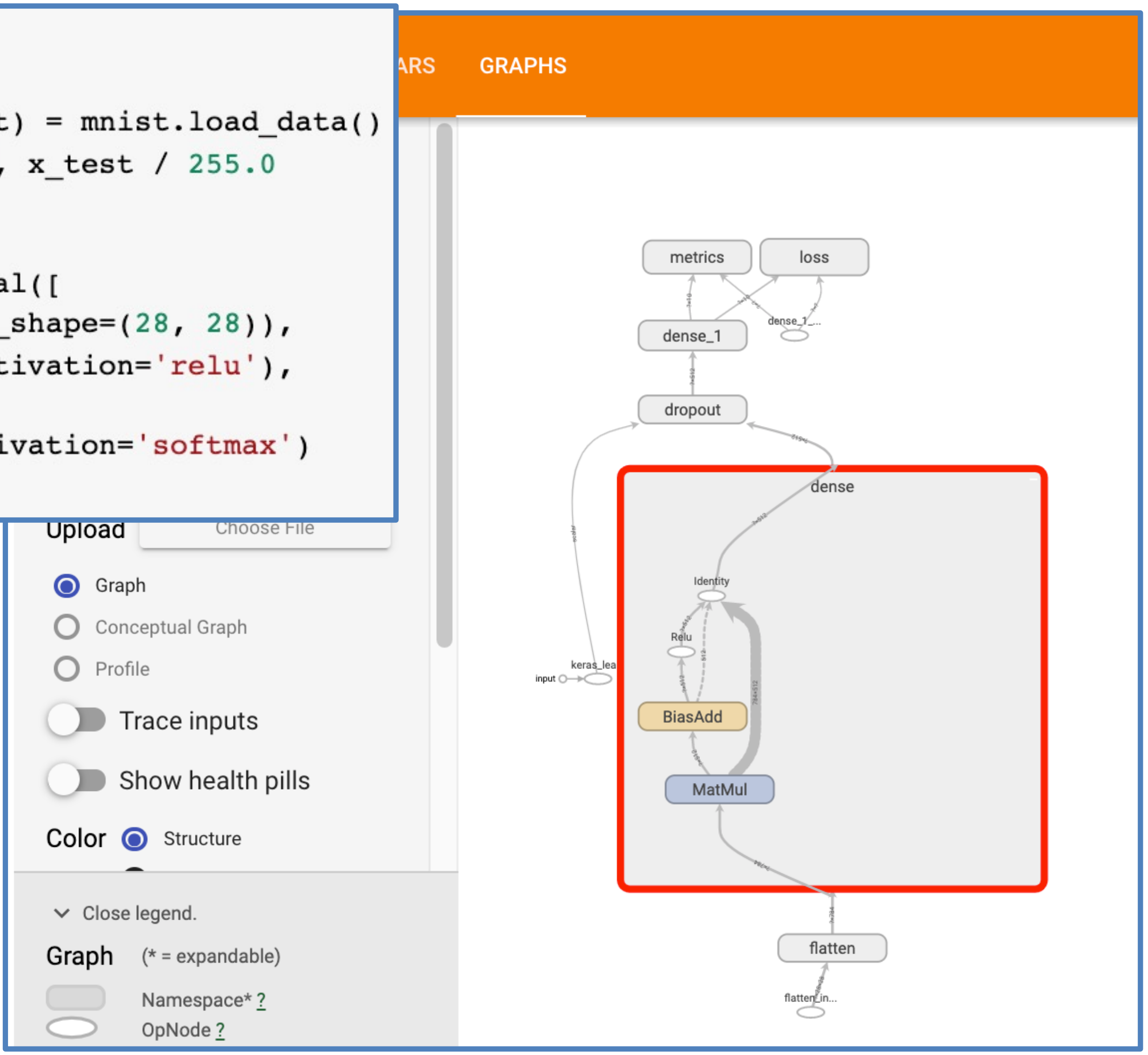
Keras

```python
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

def create_model():
  return tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
  ])
```

TensorFlow



Upload Choose File

[✓ Graph]
Conceptual Graph
Profile

Trace inputs
Show health pills

Color [✓ Structure]

Close legend.

Graph (* = expandable)

Namespace* ?
OpNode ?

Internal representation (in non-eager mode)
is a computational graph.

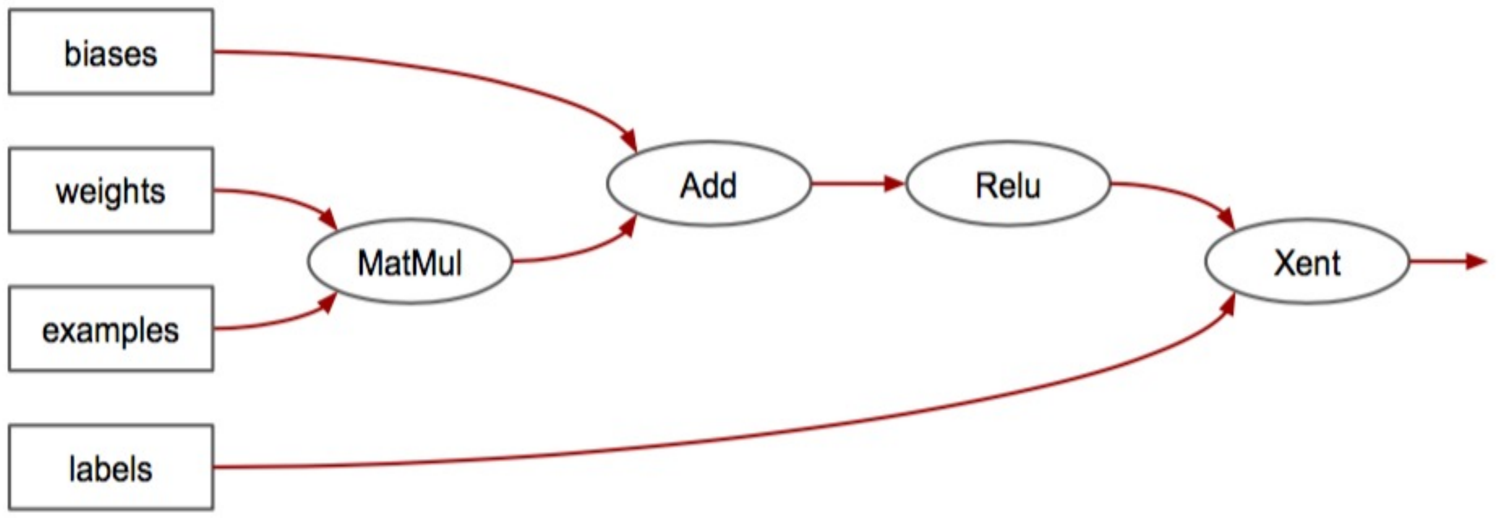
https://github.com/tensorflow/tensorboard/blob/master/docs/get_started.ipynb

Next steps

- Understand the computational graph (theoretical)
- Understand backpropagation in a graph (theoretical)

Recap

- The computation in TF is done via a computational graph

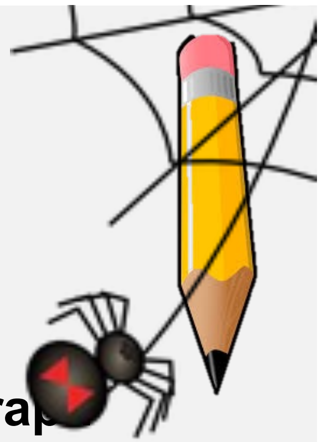


- The nodes are ops
- The edges are the flowing tensors

Recap Matrix Multiplication (scalar and with vector)

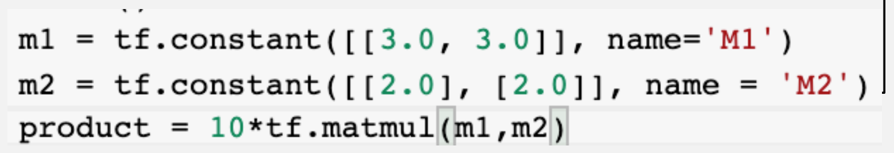
$$10\begin{pmatrix} 3 & 3 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} = 120$$

Be the spider who knits a computational graph



Translate the following TF code in a graph

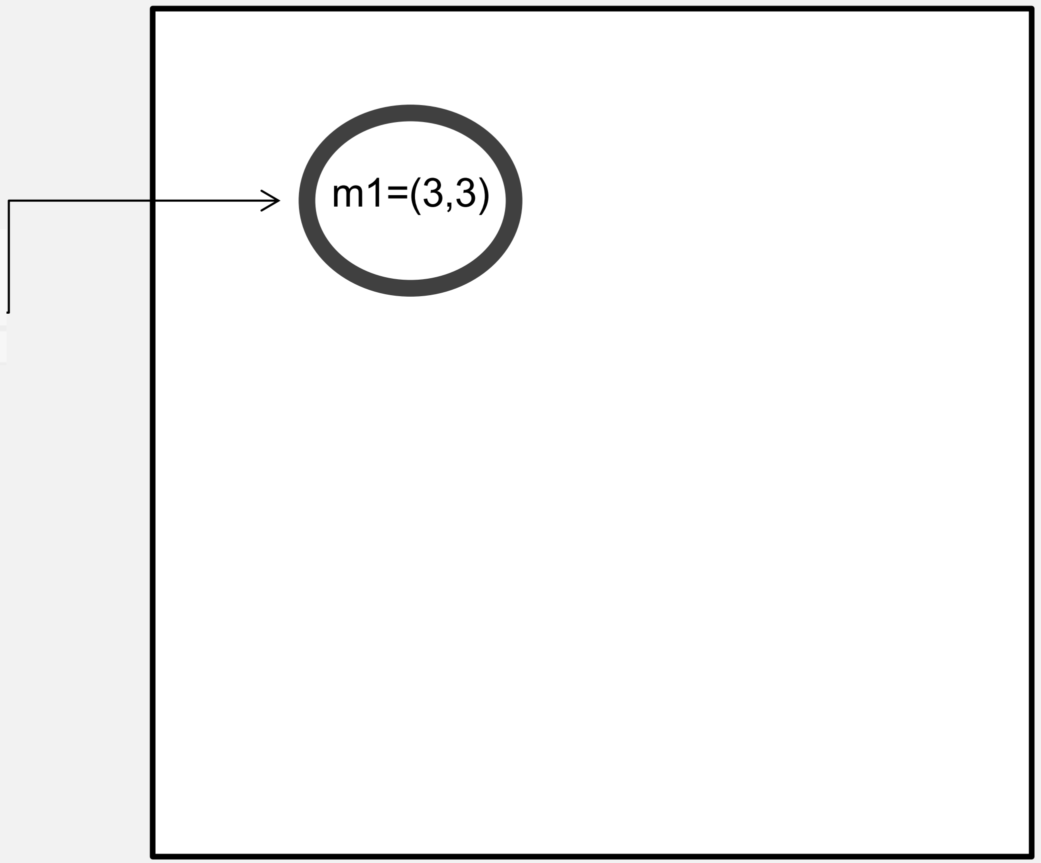
TensorFlow: Building the graph



```
m1 = tf.constant([[3.0, 3.0]], name='M1')
m2 = tf.constant([[2.0], [2.0]], name = 'M2')
product = 10*tf.matmul(m1,m2)
```

Quite much happen in here!

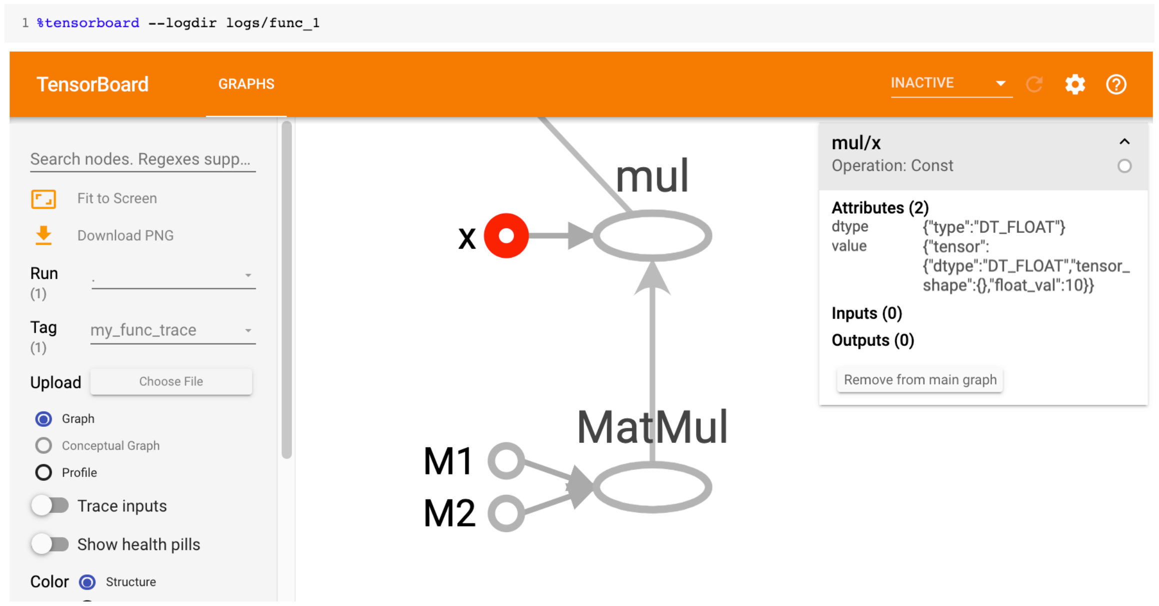
Finish the computation graph



TensorFlows internal representation

- For fast computation a graph is build
  - Technical detail in tf 2.0 you need to decorate a function with @tf.function to build a graph. Otherwise eager execution happens.

```
%tensorboard --logdir logs/func_1
```



mul/x
Operation: Const

Attributes (2)

dtype	{"type":"DT_FLOAT"}
value	{"tensor":{"dtype":"DT_FLOAT","tensor_shape":{},"float_val":10}}

Inputs (0)

Outputs (0)

Run (1): .

Tag (1): my_func_trace

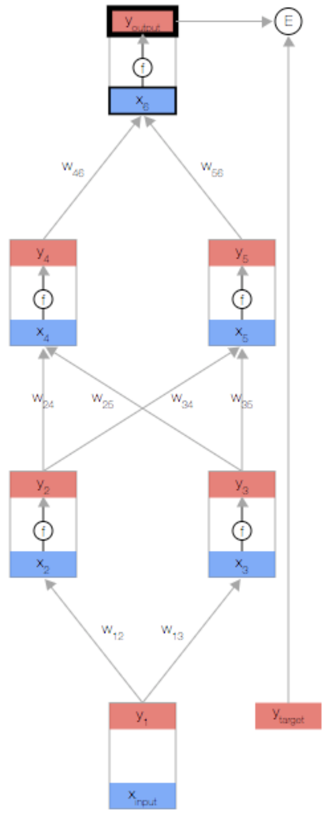
[✓ Graph] Conceptual Graph Profile

Color: [✓ Structure]

The most important benefit of computational graphs is back propagation…

Motivation: The forward and the backward pass

- https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/backprop-scroll



Scroll until the forward pass and swiftly go over the backward pass.

(The backward pass is described in more details in the next following slides).

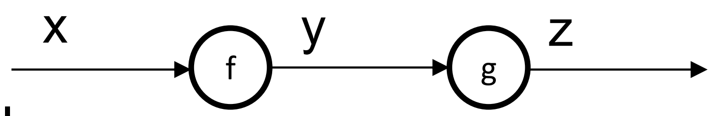
Chain rule recap

- If we have two functions f,g

  $y = f(x)\ and$

  $z = g(y)$

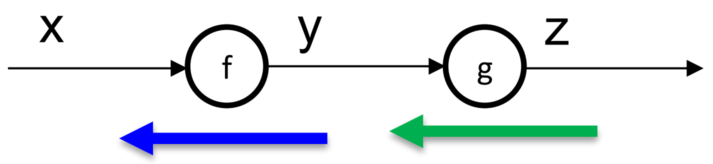
  then y and z are dependent variables.



- The chain rule:

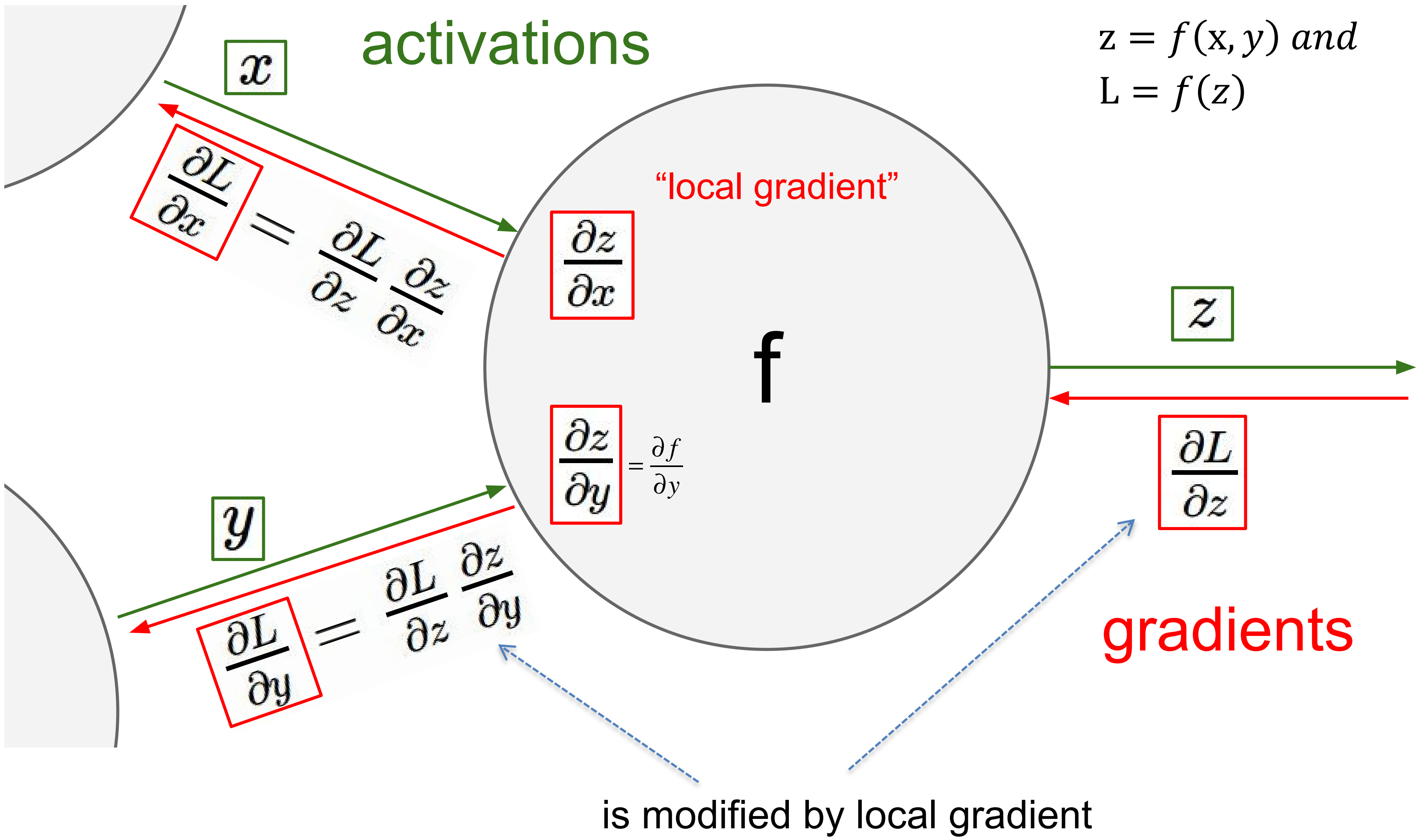
  $$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} \cdot \frac{\partial z}{\partial y}$$

- Backpropagation (flow of the gradient)



$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y}$$

Gradient flow in a computational graph: local junction

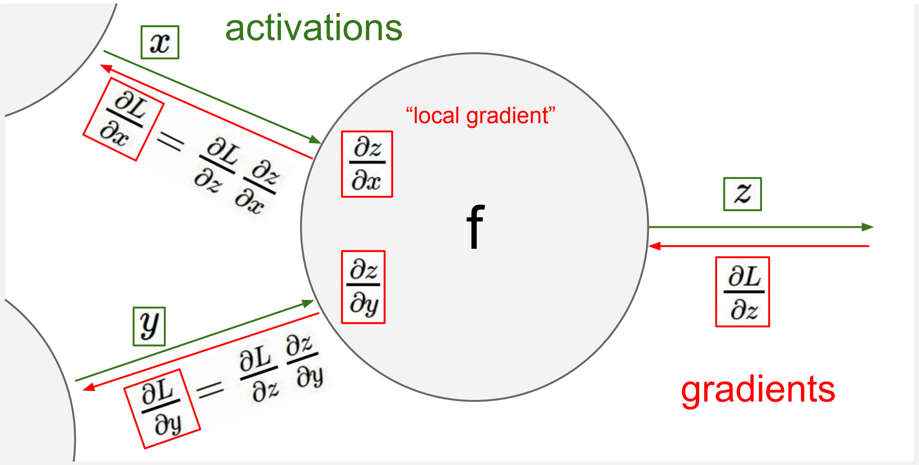


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y} = \frac{\partial f}{\partial y}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
gradients
is modified by local gradient

$z = f(x, y)$ *and*
$L = f(z)$

Illustration: http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

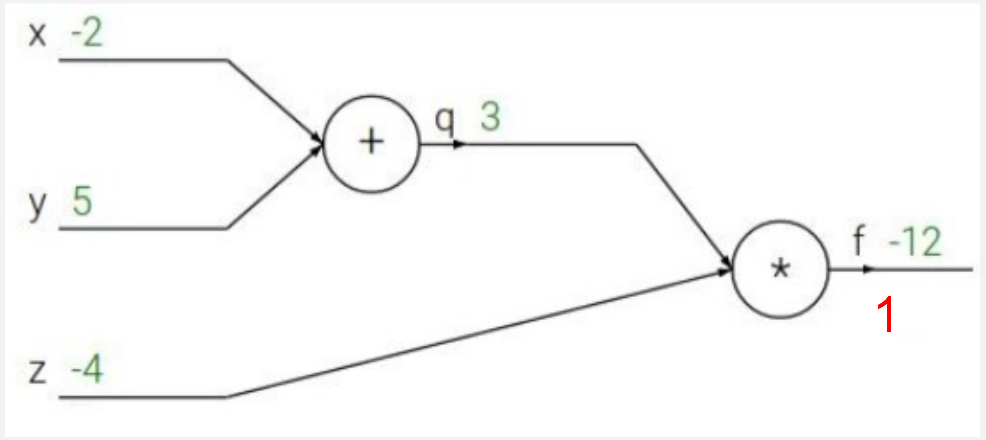
Example



x
activations
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$\frac{\partial(\alpha+\beta)}{\partial\alpha} = 1 \qquad \frac{\partial(\alpha*\beta)}{\partial\alpha} = \beta$$

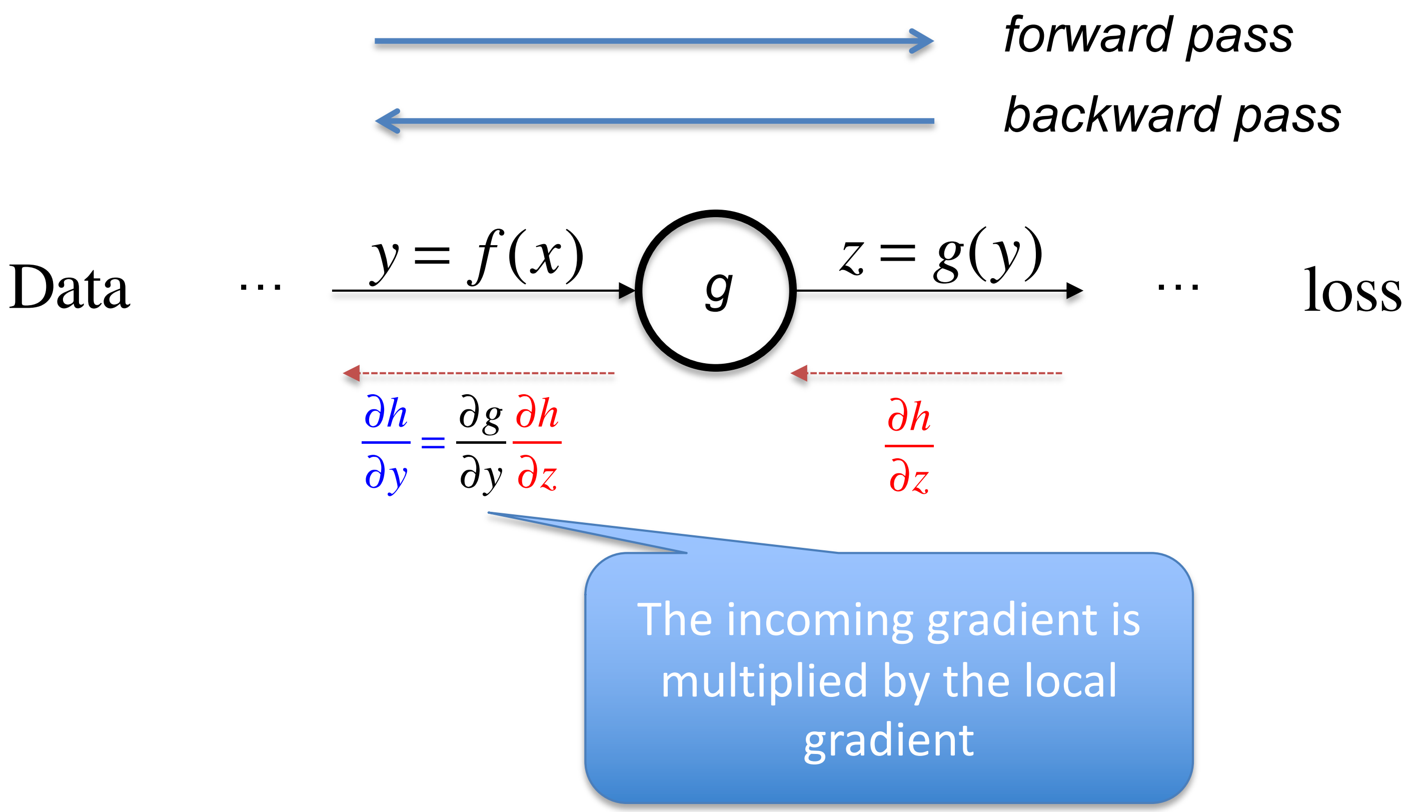
Task (10min): Calculate the derivatives. Once by hand, once with backpropagation (follow the graph)

➔Multiplication do a switch

Further References / Summary

- For a more in depth treatment have a look at
  - Lecture 4 of http://cs231n.stanford.edu/
  - Slides http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

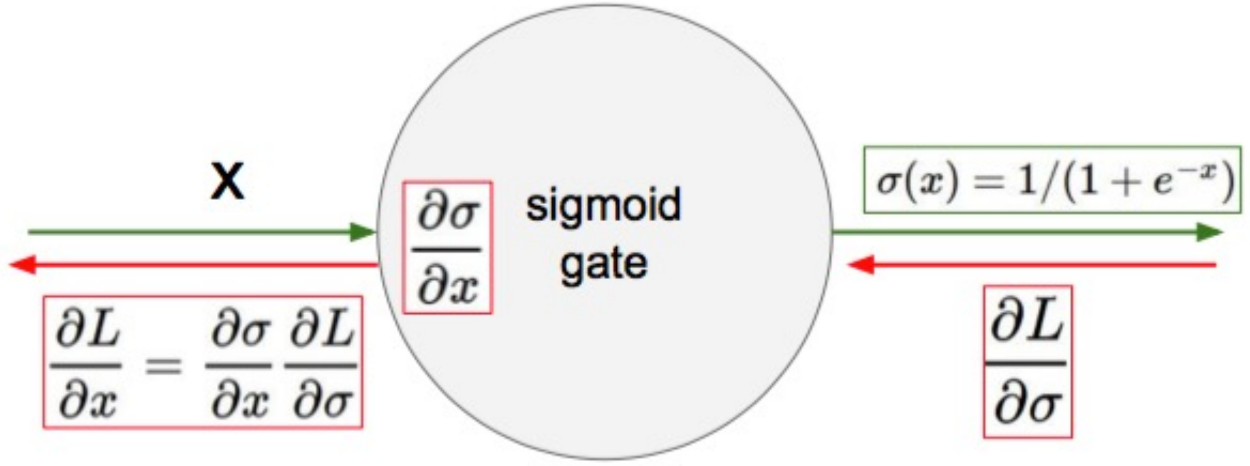
- Gradient flow is important for learning: remember!



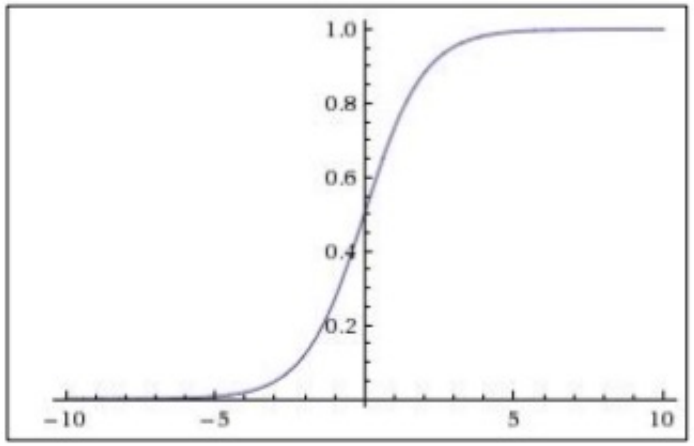
forward pass
backward pass
Data
$\cdots$
$y = f(x)$
g
$z = g(y)$
$\cdots$
loss
$\frac{\partial h}{\partial y} = \frac{\partial g}{\partial y}\frac{\partial h}{\partial z}$
$\frac{\partial h}{\partial z}$
The incoming gradient is multiplied by the local gradient

Consequences of Backprop

Backpropagation through sigmoid



x
sigmoid gate
$\frac{\partial \sigma}{\partial x}$
$\sigma(x) = 1/(1+e^{-x})$
$\frac{\partial L}{\partial \sigma}$
$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x}\frac{\partial L}{\partial \sigma}$



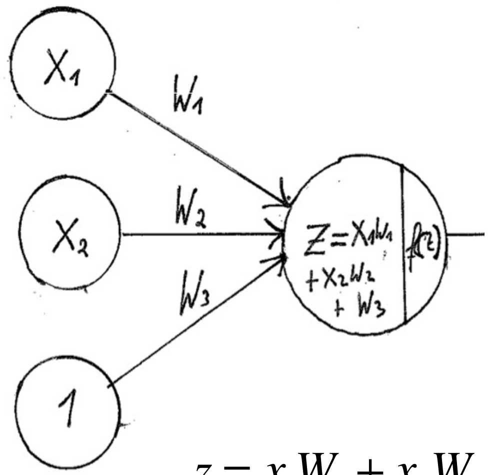
What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

Gradients are killed, when not in active region! Slow learning!

Slide from: CS231

Different activations in inner layers

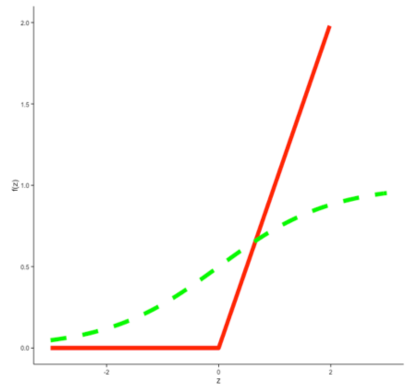
N-D log regression



$$f(z) = \begin{cases} \dfrac{\exp(z)}{1+\exp(z)} \\ \max(0,z) \end{cases}$$

$$z = x_1W_1 + x_2W_2 + b = Wx + b$$

Activation function a.k.a. Nonlinearity f(z)



Motivation:
Green:
logistic regression.
Red:
ReLU faster convergence

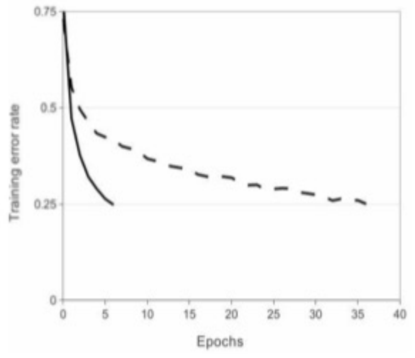


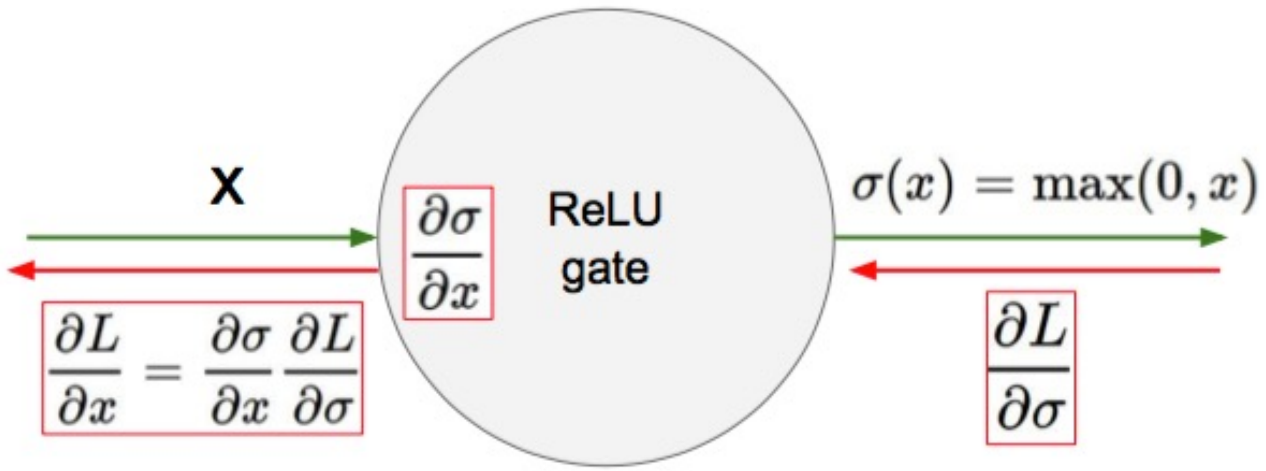
Figure 1: A four-layer convolutional neural network with ReLUs (**solid line**) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons

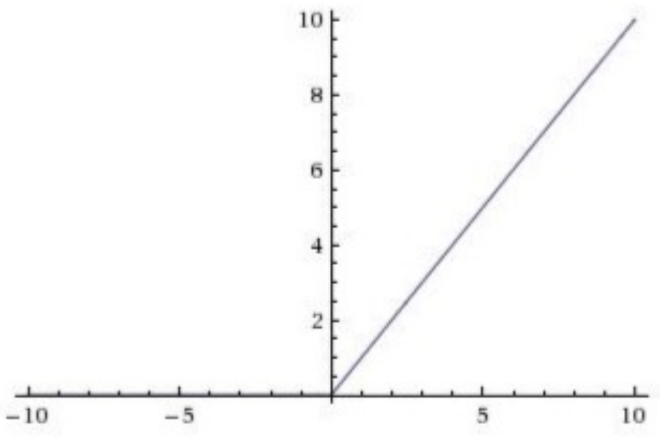
Source:
Alexnet
Krizhevsky et al 2012

There are other alternatives besides sigmoid and ReLU.

Currently ReLU is standard

Backpropagation through ReLU


x
ReLU gate
$\frac{\partial \sigma}{\partial x}$
$\sigma(x) = \max(0, x)$
$\frac{\partial L}{\partial \sigma}$
$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x} \frac{\partial L}{\partial \sigma}$



What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

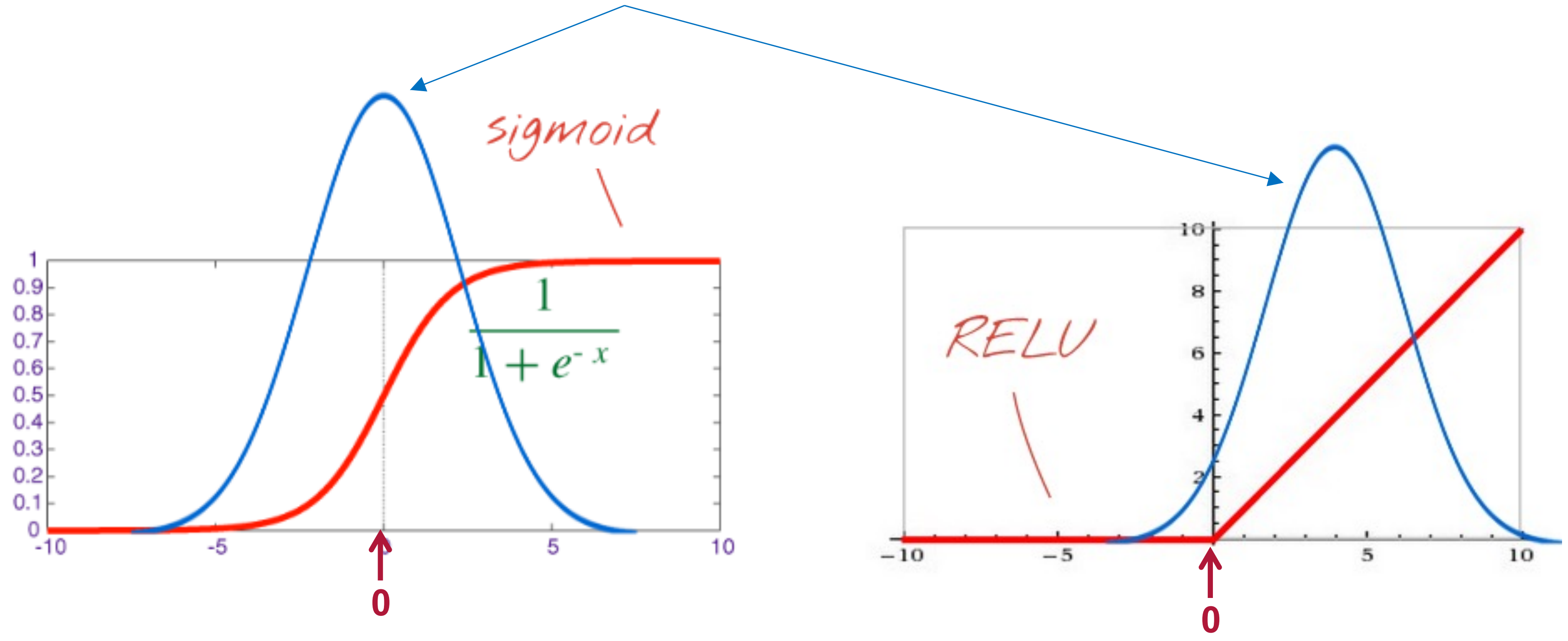
Gradients are killed, only when x < 0

Slide from: CS231

Recap: Batch Normalization is beneficial in many NN

After BN the input to the activation function is in the sweet spot

Observed distributions of signal after BN before going into the activation layer.



When using BN consider the following:

- Using a higher learning rate might work better
- Use less regularization, e.g. reduce dropout probability
- In the linear transformation the biases can be dropped (step 2 takes care of the shift)
- In case of ReLu only the shift β in steps 2 need to be learned (α can be dropped)

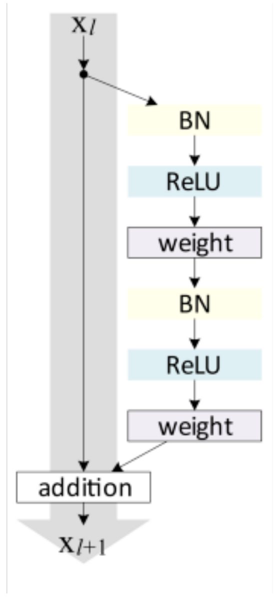
Image credits: Martin Gorner: https://docs.google.com/presentation/d/e/2PACX-1vRouwj_3cYsmLrNNI3Uq5gv5-hYp_QFdeoan2GlxKgIZRSejozruAbVV0IMXBoPsINB7Jw92vJo2EAM/pub?slide=id.g187d73109b_1_2921

"ResNet" from Microsoft 2015 winner of imageNet

152
layers

ResNet basic design (VGG-style)

- add shortcut connections every two
- all 3x3 conv (almost)



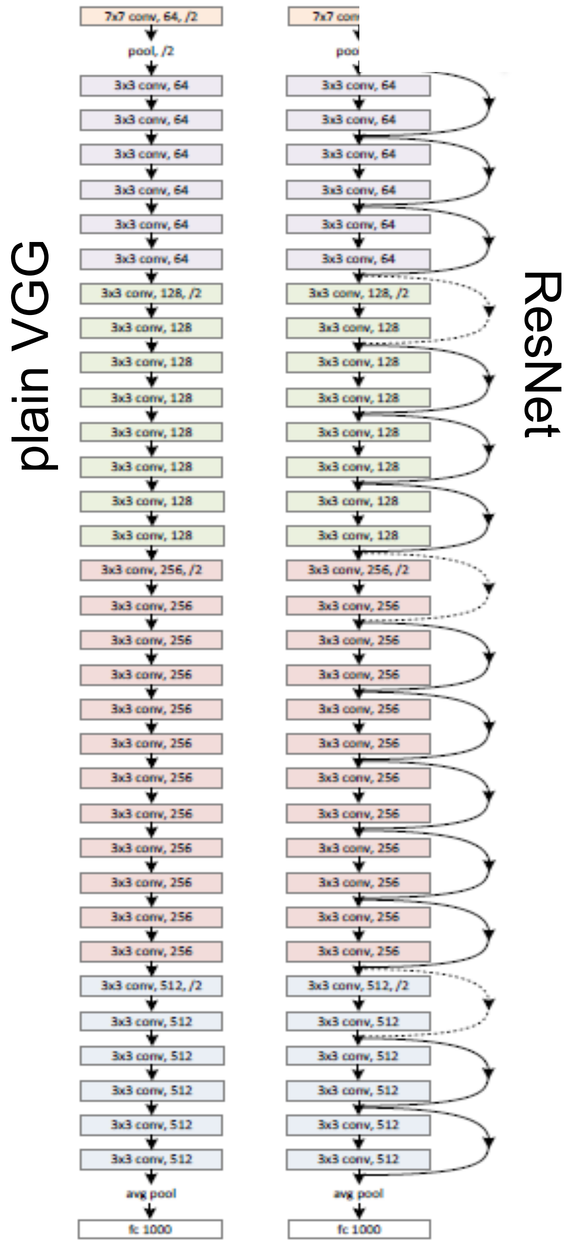
$F(x_l)$

$H(x_l)=x_{l+1} = x_l + F(x_l)$

152 layers:
Why does this train at all?

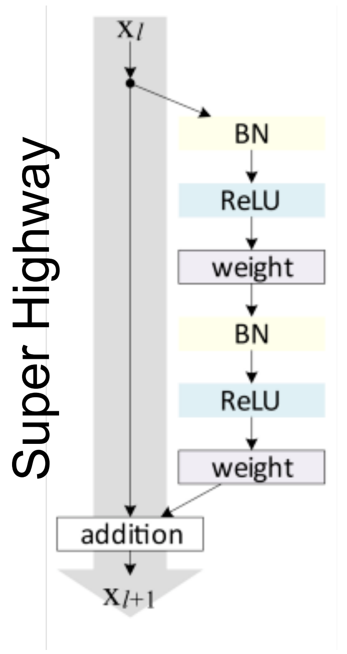
This deep architecture could still be trained, since the gradients can skip layers which diminish the gradient!

F(x) is called "residual" since it only learns the "delta" which is needed to add to x to get H(x)



credits: https://chatbotslife.com/resnets-highwaynets-and-densenets-oh-my-9bb15918ee32

Closer Look



$$\frac{\partial(\alpha+\beta)}{\partial\alpha}=1$$

➔'Gradient Super Highways'

What comes in (on the right) does go out (on the left)

Similar to LTMS (just in case you know)

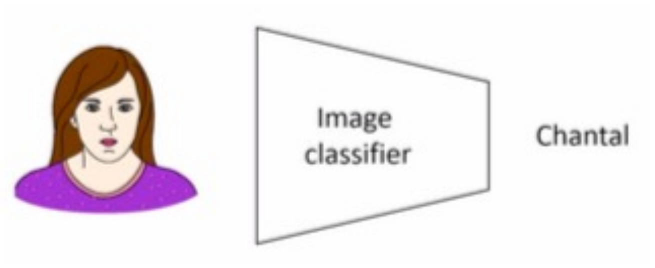
Probabilistic Models

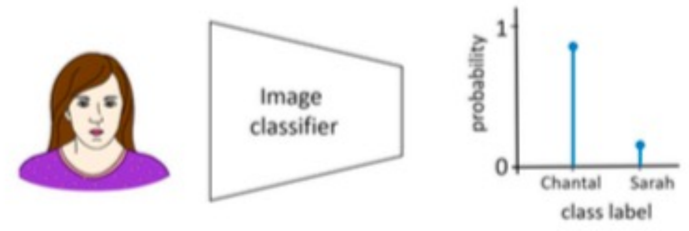
Probabilistic vs deterministic models

Deterministic

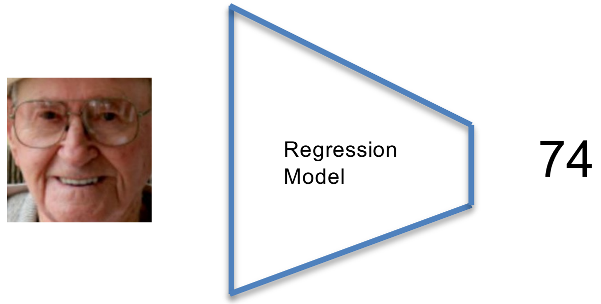
Probabilistic

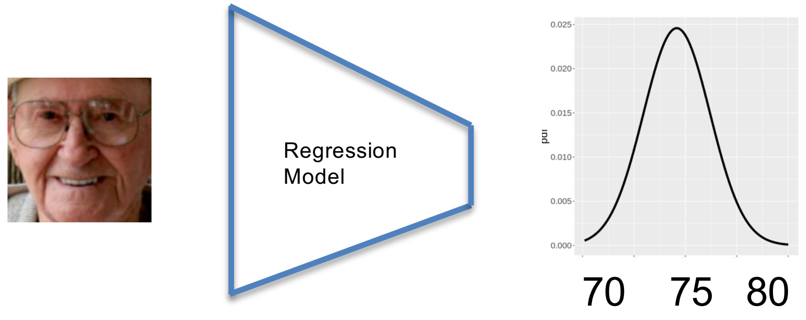
“Classification”





“Regression”



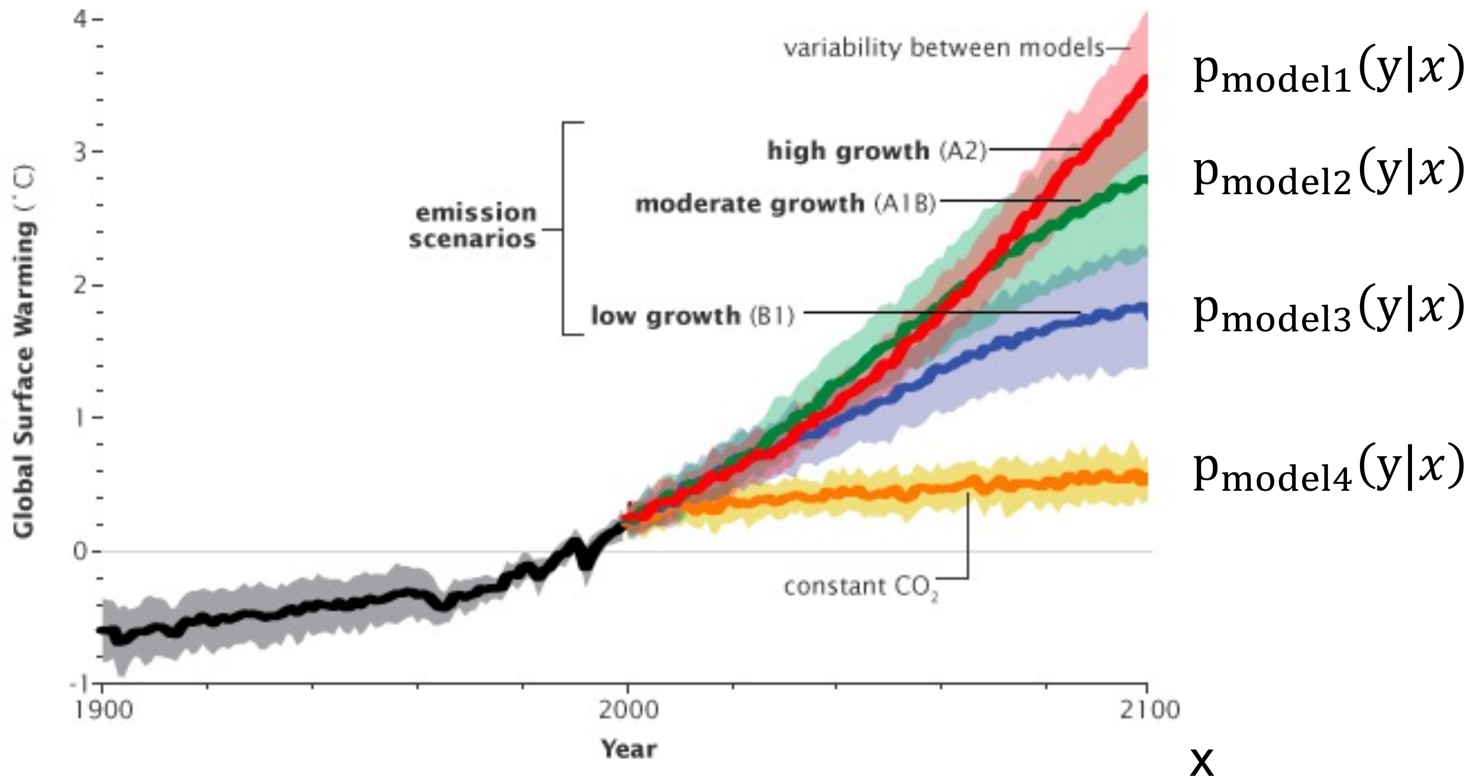


Conditional probability distribution (CPD)

$p(y|x)$

Examples of probabilistic models: global warming forecast

For each time-point x a probability distributions is reported (for 4 models)



$p(y|x\ )$

- y is a 1D continuous variable (of the increase of the global surface temperature)
- x is 1D continuous time

Figure from: https://earthobservatory.nasa.gov/features/GlobalWarming

Benefits of probabilistic classification / regression

Probabilistic models quantify uncertainty

- "It is scientific to say what is more likely and what is less likely..."
  - Richard Feynman

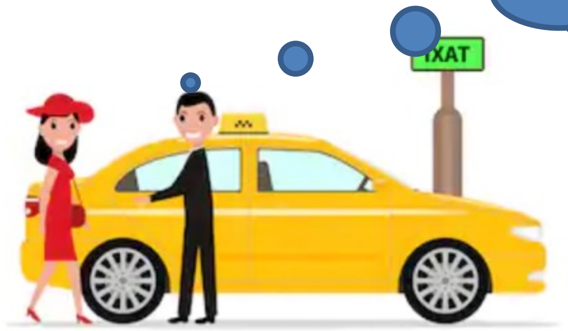
- Example: Classification of cancer from tissue
  - Probabilistic model
    - Patient A probability of cancer is 49%
    - Patient B of cancer is 0.0001%
    - Doctor will do further checks on patient A
  - Non-Probabilistic model
    - Patient A $\rightarrow$ no cancer
    - Patient B $\rightarrow$ no cancer
    - Doctor has no clue that patient A is on the edge

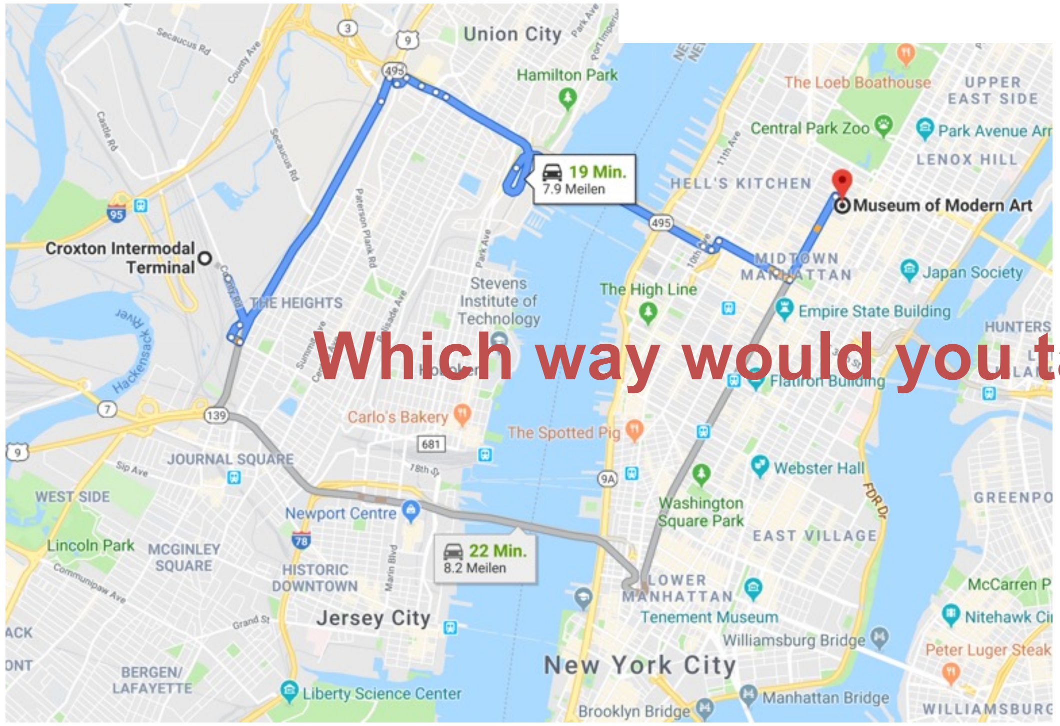
- Practical reason, minimize expected costs need probabilities.

Probabilistic travel time prediction (cost)

You'll get 500$ tip if I arrive at MOMA within 25 minutes!

Let's use my probabilistic travel time gadget!



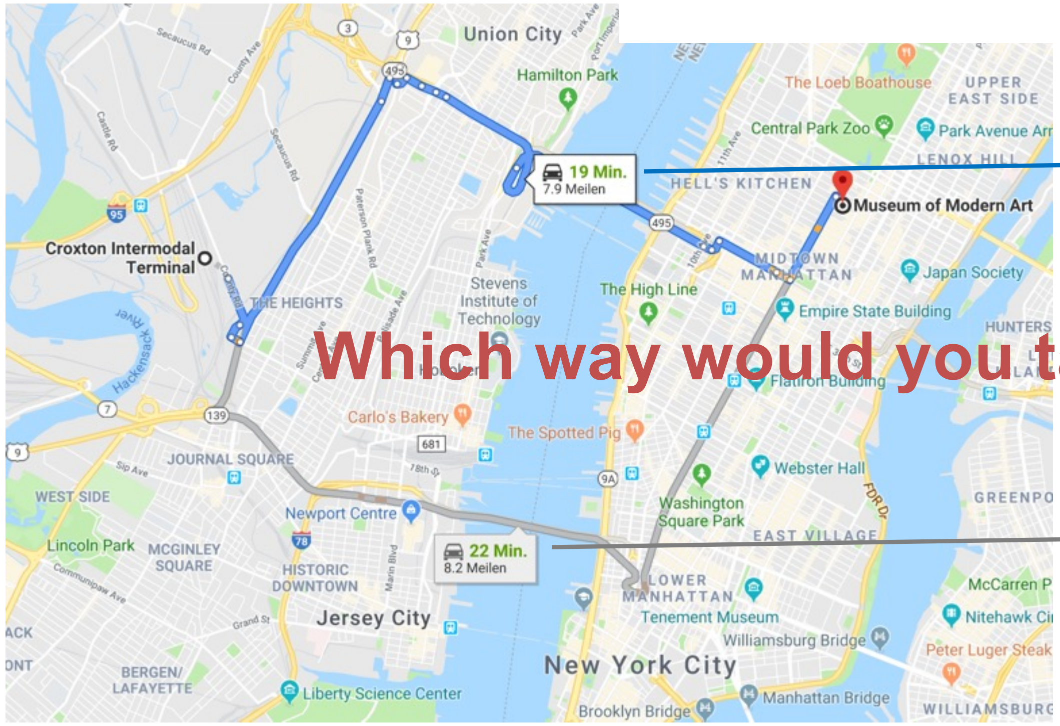


Which way would you take?

Probabilistic travel time prediction (cost)

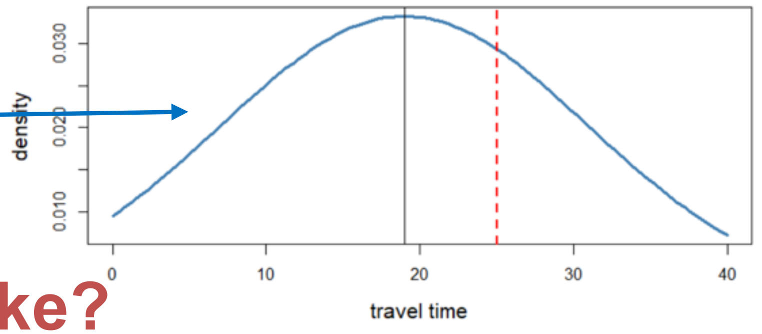
You'll get 500$ tip if I arrive at MOMA within 25 minutes!

Let's use my probabilistic travel time gadget!

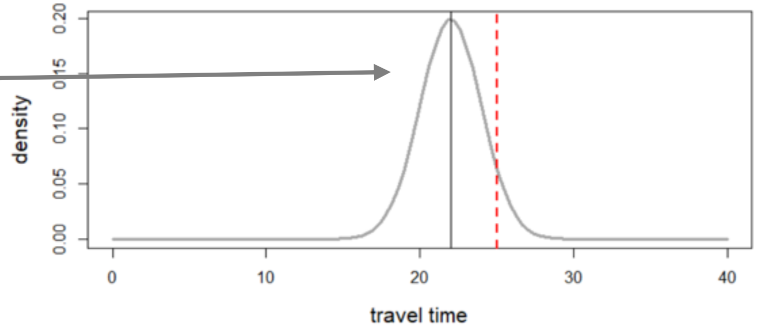


Which way would you take?

Chance to get tip: 69%



Chance to get tip: 93%



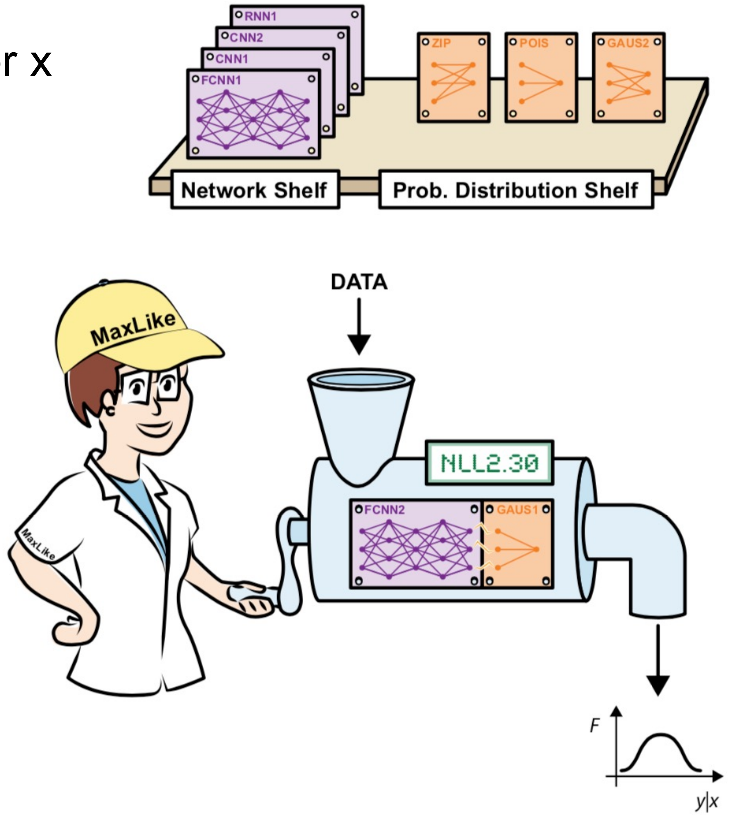
Probabilistic Model

How to build probabilistic deep learning models?

- Recipe
  1. Build model that outputs the probability $p(y|x)$ of y given the input x.
  2. Train model to maximize the probability of the training data (Maximum Likelihood Principle)

Special networks for x
- Vector FCNN
- Image CNN
- Text CNN/RNN

Special heads for y
- Classes
- Regression



Recap Classification: Softmax Activation

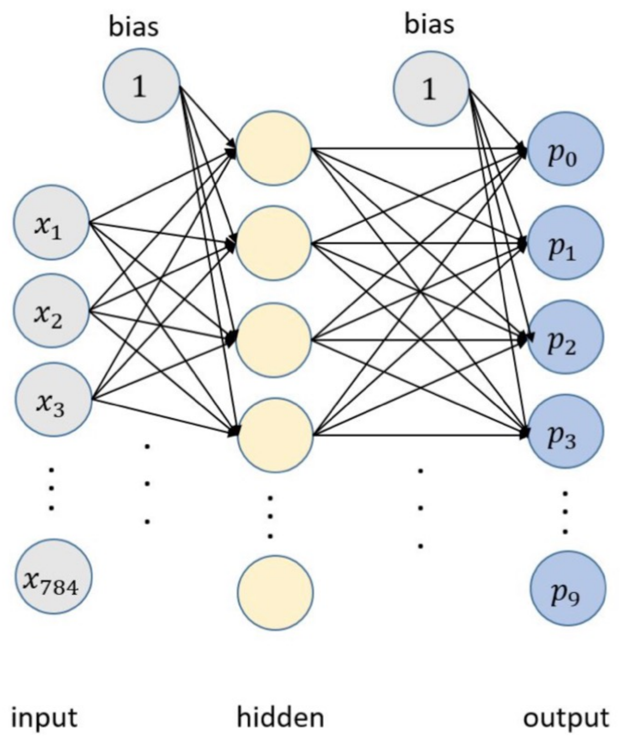


Figure 2.12: A fully connected NN with 2 hidden layers. For the MNIST example, the input layer has 784 values for the 28 x 28 pixels and the output layer out of 10 nodes for the 10 classes.

$p_o, p_1 \dots p_9$ are probabilities for the classes 0 to 9.

Activation of last layer z_i incomming

$$p_i = \frac{e^{z_i}}{\sum_{j=0}^{9} e^{z_j}}$$

Makes outcome positive

Ensures that pi's sum up to one

This activation is called softmax

Neural networks are probabilistic models

- The output of a neural network, can be understood as a probability*
  - Classification
    - Probability of class 1…,K
  - Regression
    - Probability Distribution
- Output of a neural network for training example i

$$p(y_i|x_i, w)$$

Output
- Probability for class y_i
- Distribution

Input

weights

*More on probabilistic interpretation next lecture

Maximum Likelihood

Tune the parameters weights of the network, so that observed data (training data) is most likely.

Practically: Minimize Negative Log-Likelihood of the CPD

$$\hat{w} = argmin \sum_{i=i}^{N} -\log(p(y_i|x_i, w))$$

Maximum Likelihood
(one of the most beautiful ideas in statistics)

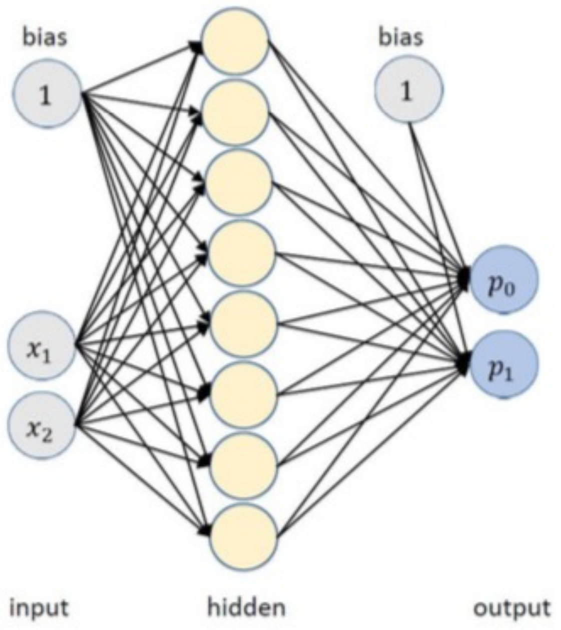
Ronald Fisher in 1913
Also used before by
Gauss, Laplace

$$M(\theta) \xrightarrow{\text{Likelihood / “probability“ (often known)}} \text{Data}$$

Tune the parameter(s) θ of the model M so that (observed) data is most likely

What‘s the likelihood of the data for lin. regression...

Maximum Likelihood principle for binary classification



x_i, y_i Training data $i = 1, \dots N$

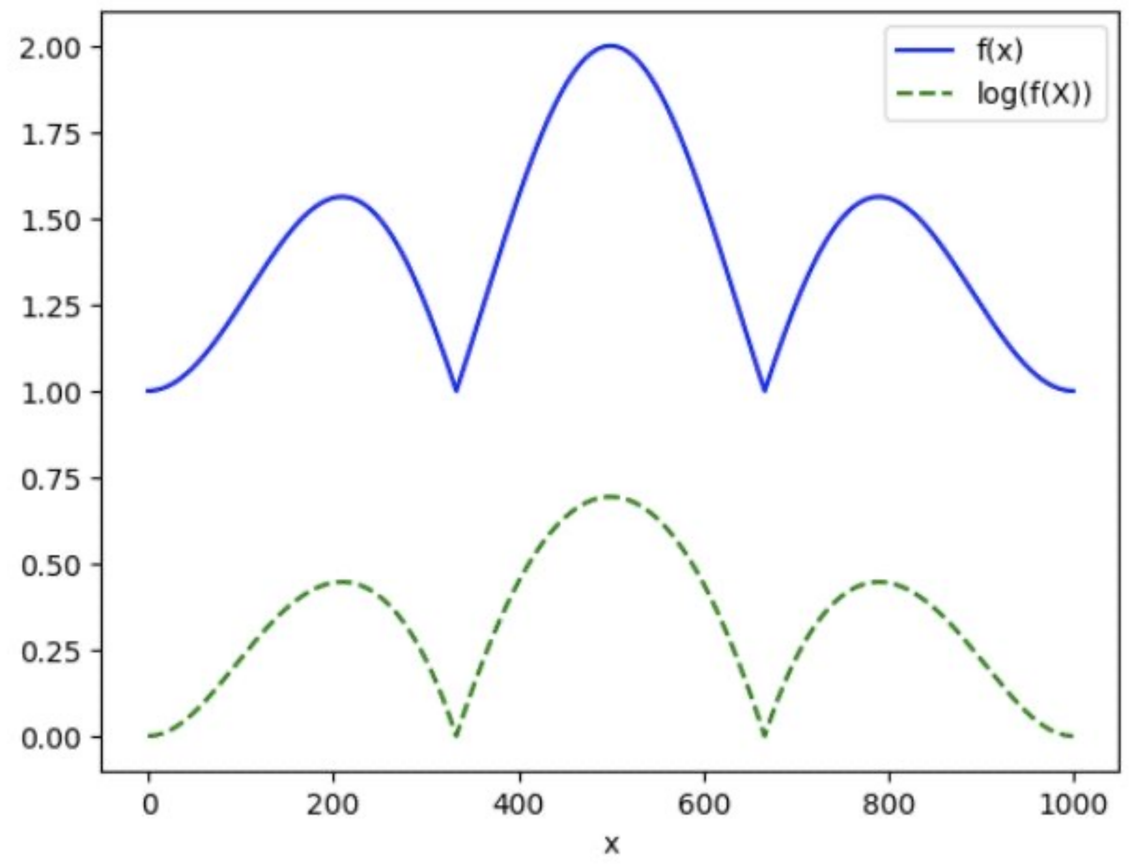
$p_0(x_i)$ is probability for $y_i = 0$

$p_0(x_i)$ is probability for $y_i = 1$

Answer:
What is probability for the training set of say 5 examples? The first 3 are of class 0, last two 2 of class 1?

$$Pr(Training) = p_0(x_1) \cdot p_0(x_2) \cdot p_0(x_3) \cdot p_1(x_4) \cdot p_1(x_5) = \prod_{j=1}^{3} p_0(x_j) \cdot \prod_{j=4}^{5} p_1(x_j)$$

Taking the log



To determine the maximal value, taking log is also ok.

Negative Log-Likelihood (NLL)

- Likelihood of training data

$$Pr(Training) = \prod_{j\ for\ with\ y=0} p_0(x_j) \cdot \prod_{j\ for\ with\ y=1} p_1(x_j)$$

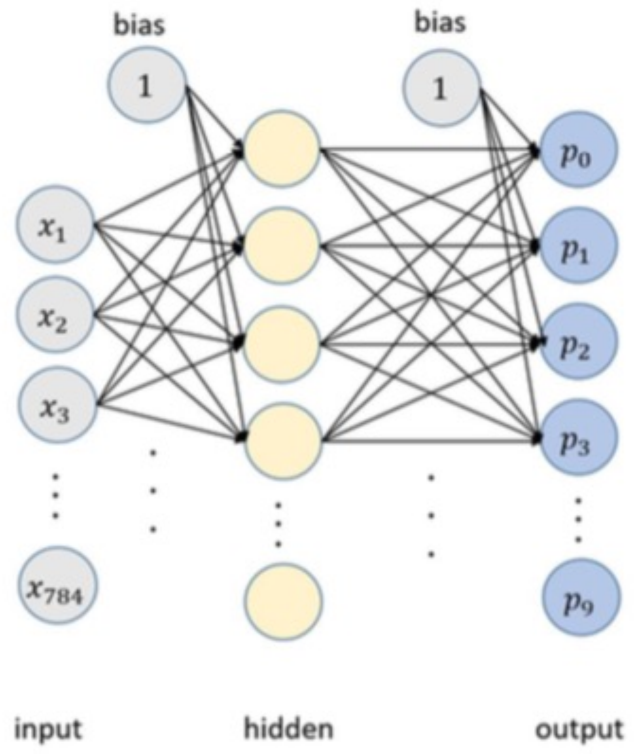
- LogLike

$$\log(Pr(Training)) = \sum_{j\ for\ y=0} \log(p_0(x_j)) + \sum_{j\ for\ y=1} \log(p_1(x_j))$$

- Crossentropy / NNL negative log likelihood (per example divided by n)

$$crossentropy = -\tfrac{1}{n}\left(\sum_{j\ for\ y=0} \log(p_0(x_j)) + \sum_{j\ for\ y=1} \log(p_1(x_j))\right)$$

More than 2 classes



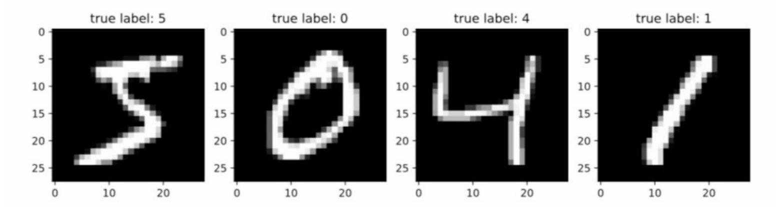


Figure 2.11 The first four digits of the MNIST data set—the standard data set used for benchmarking NN for images classification

$$crossentropy = -\frac{1}{n}\left(\sum_{j\ for\ y=0} \log(p_0(x_j)) + \sum_{j\ for\ y=1} \log(p_1(x_j)) + \ldots + \sum_{j\ for\ y=K-1} \log(p_{k-1}(x_j))\right)$$

$$crossentropy = -\frac{1}{n}\sum_{i=1}^{n} {}^{true}p_i \cdot log(p_i)$$

Continous Data

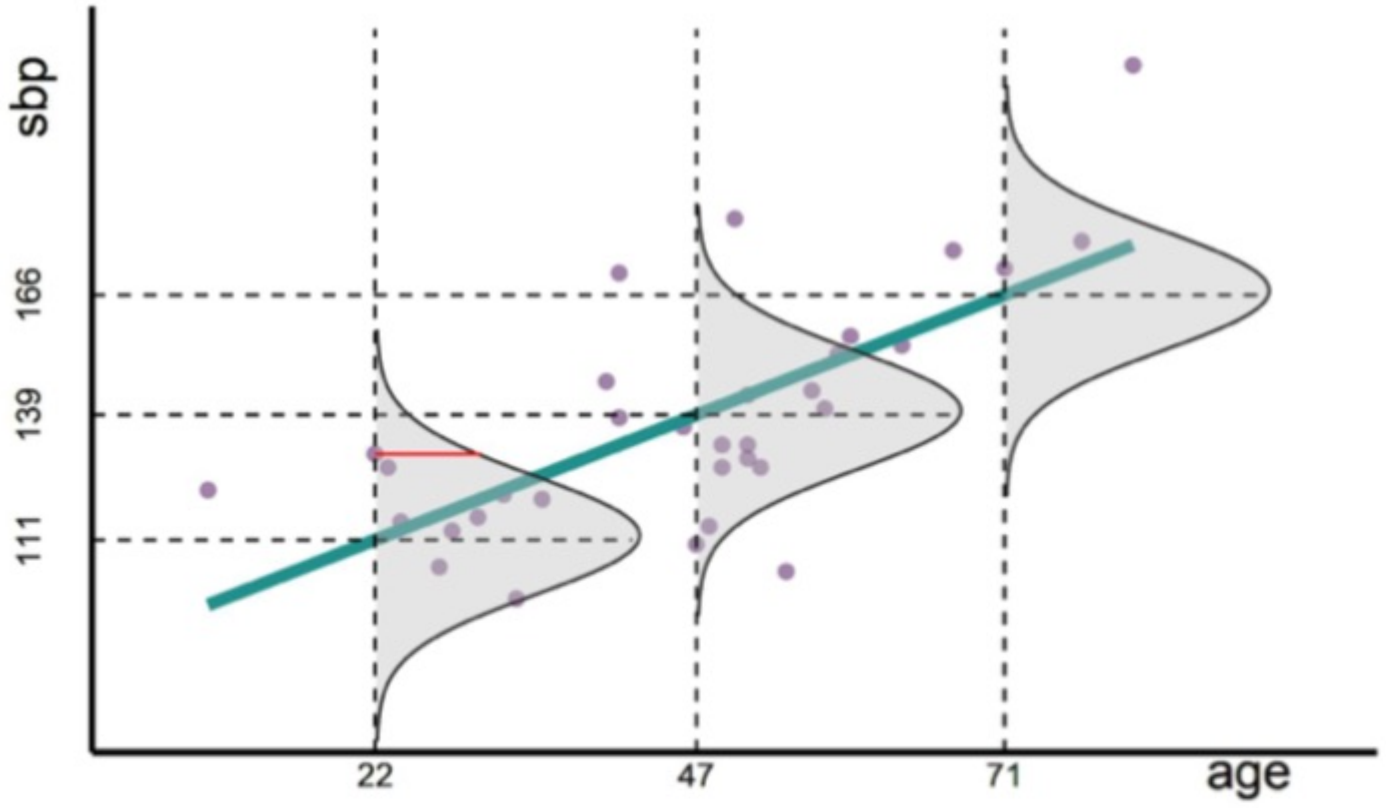
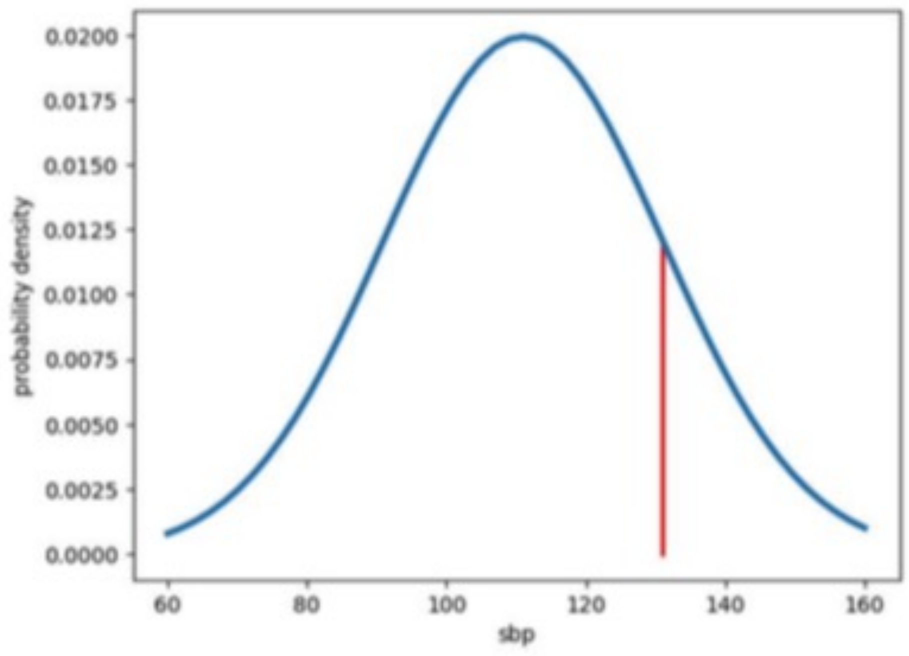


Figure 4.12 Scatterplot and regression model for the blood pressure example: The dots are the measured data points, the straight line is the linear model, the bell-shaped curves are the conditional probability distributions of the outcome Y conditioned on the observed value x. The length of the red bar indicates likelihood for the observed sbp of 131 for a 22 year old woman.

Zoom for one example



$$f(y = 131, \mu = 111, \sigma = 20) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi \cdot 400}} e^{-\frac{(131-111)^2}{2400}} \approx 0.01209$$

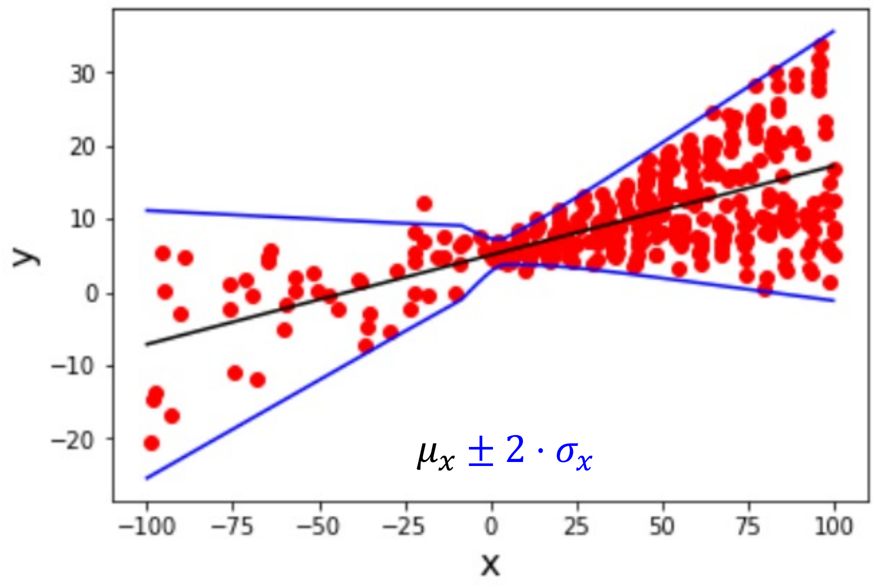
Figure 4.13 The conditional normal density function f. The height of the red bar indicates the likelihood of the specific value under this model.

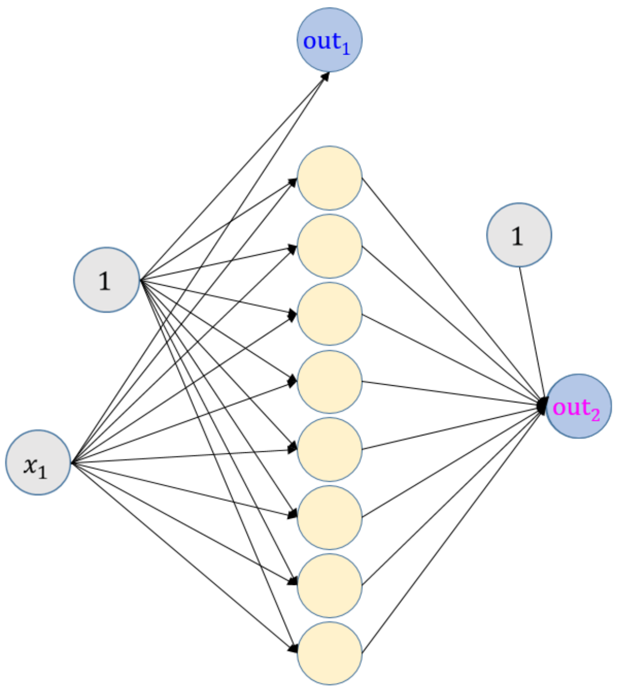
For all training Examples:

$$L(a, b) = f(y_1, \mu_{x_2,} \sigma) \cdot f(y_2, \mu_{x_2}, \sigma) \ldots = \prod_{i=1}^{n} f(y_i, \mu_{i,} \sigma) = \prod_{i=1}^{n} f(y_{i,} a \cdot x_i + b)$$

Fit a probabilistic regression with flexible non-constant variance

$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma_x^2)$$





$$\mu_{x_i} = \text{out}_{1_i}$$

$$\sigma_{x_i} = e^{\text{out}_{2_i}}$$

Minimize the mean negative log-likelihood (NLL) on train data:

$$NLL(w) = \frac{1}{n} \sum_{train-data} -log(p(y_i|x_i, w))$$

gradient descent with NLL loss

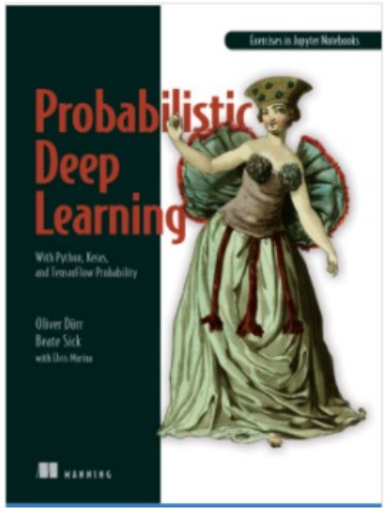
$$\hat{\mathbf{w}}_{\text{ML}} = \underset{w}{\text{argmin}} \sum_{i=1}^{n} -\log\left(\frac{1}{\sqrt{2\pi\sigma_{x_i}^2}}\right) + \frac{\left(y_i - \mu_{x_i}\right)^2}{2\sigma_{x_i}^2}$$

Note: we do not need to know the "ground truth for σ" – the likelihood does the job!

Final Note on Probabilistic Models

- In many problems it’s pretentious to do exact forecasts
  - You can’t predict the value of an asset tomorrow
  - You can’t forecast tomorrows’ weather with no uncertainity
  - You can’t tell if a person has cancer given a tiny image
  - …

- It’s sounder to provide a probabilistic forecast
- Deep Learning can be made probabilistic with practically no afford.
- More on probabilistic deep learning
  - https://tensorchiefs.github.io/dl_rcourse_2022/ (in R)
  - https://tensorchiefs.github.io/dl_course_2022/ (in pythion)
  - Book on probabilistic deep learning



Exercises in Jupyter Notebooks
Probabilistic Deep Learning
With Python, Keras, and TensorFlow Probability
Oliver Dürr
Beate Sick
with Elvis Murina
MANNING

Final Projects (deliverables)

- For the final projects, due Thursday 17th November 12:50
  - ~5 Minutes spotlight talk
    - Send pdf of talk by latest by Thurday by course line
    - First Slide must contain names and student ids
  - Poster presentation
    - Introduce project to your fellow students and myself
    - Can be just some printed slides