UPPSALA UNIVERSITET

Statistical Machine Learning

Lecture 9 – Deep Learning and Neural Networks

UPPSALA UNIVERSITET

Niklas Wahlström
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: niklas.wahlstrom@it.uu.se,
www.it.uu.se/katalog/nikwa778

Summary of lecture 8 (I/II)

Flexible models often gives best performance.

We introduced the non-parametric probabilistic Gaussian process (GP) model.

Def. (Gaussian Process) A Gaussian process is a (potentially infinite) collection of random variables such that any finite subset of it is jointly distributed according to a multivariate Gaussian.

We assumed

$$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} m(x) \\ m(x') \end{pmatrix}, \begin{pmatrix} k(x,x) & k(x,x') \\ k(x',x) & k(x',x') \end{pmatrix}\right)$$

UPPSALA UNIVERSITET

Summary of lecture 8 (II/II)

More compact we write

$$f \sim \mathcal{GP}(m, k)$$

$$\begin{pmatrix} \mathbf{f} \\ f(x_\star) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(x_\star) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}, \mathbf{x}) & k(\mathbf{x}, x_\star) \\ k(x_\star, \mathbf{x}) & k(x_\star, x_\star) \end{pmatrix} \right),$$

GP regression: Given training data $\mathcal{T} = \{x_i, y_i\}_{i=1}^N$ and our GP prior on f, $f \sim \mathcal{GP}(m, k)$, we computed (using the theorem for conditioned Gaussians)

$$p(f_\star \mid \mathbf{y}),$$

for an arbitrary test point $\{x_\star, y_\star\}$.

Deep Learning Example: Automatic caption generation

Generate caption automatically from images

Xu, K., Lei Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R. Richard S. Zemel, R. S., and Bengio, Y. **Show, attend and tell: neural image caption generation with visual attention**. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July, 2015.

Constructing NN for regression

A **neural network (NN)** is a nonlinear function $Y = f_\theta(X)$ from an input X to a output Y parameterized by parameters θ.

Linear regression models the relationship between a continuous output Y and a continuous input X,

$$Y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j = \beta^\mathsf{T} X + \varepsilon,$$

where β is the parameters composed by the "weights" β_j and the offset ("bias"/"intercept") term β_j,

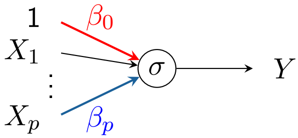
$$\beta = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 & \cdots & \beta_p \end{pmatrix}^\mathsf{T},$$

$$X = \begin{pmatrix} 1 & X_1 & X_2 & \cdots & X_p \end{pmatrix}^\mathsf{T}.$$

Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\beta^\mathsf{T} X$,

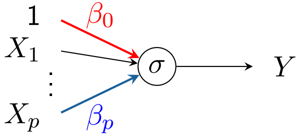
$$Y = \sigma(\beta^\mathsf{T} X) + \varepsilon.$$



Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\beta^\mathsf{T} X$,

$$Y = \sigma(\beta^\mathsf{T} X) + \varepsilon.$$

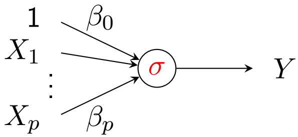


We call $\sigma(x)$ the *activation function*. Two common choices are:

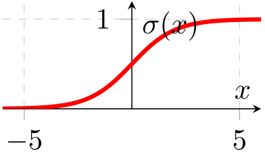
Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\beta^\mathsf{T} X$,

$$Y = \sigma(\beta^\mathsf{T} X) + \varepsilon.$$



We call $\sigma(x)$ the *activation function*. Two common choices are:

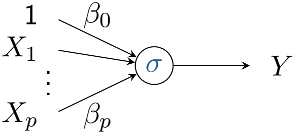


Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

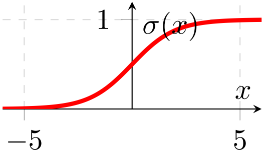
Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\beta^\mathsf{T} X$,

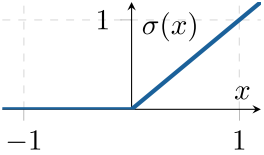
$$Y = \sigma(\beta^\mathsf{T} X) + \varepsilon.$$



We call $\sigma(x)$ the *activation function*. Two common choices are:



Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$



ReLU: $\sigma(x) = \max(0, x)$

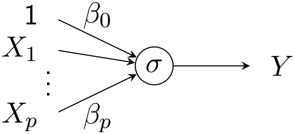
Let us consider an example of a **feed-forward NN**, indicating that the information flows from the input to the output layer.

UPPSALA UNIVERSITET

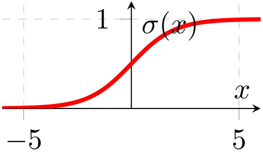
Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\beta^\mathsf{T} X$,

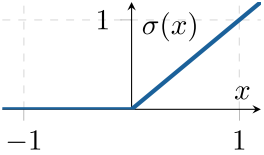
$$Y = \sigma(\beta^\mathsf{T} X) + \varepsilon.$$



We call $\sigma(x)$ the *activation function*. Two common choices are:



Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$



ReLU: $\sigma(x) = \max(0, x)$

Let us consider an example of a **feed-forward NN**, indicating that the information flows from the input to the output layer.

Neural network - construction

A NN is a sequential construction of several linear regression models.

Inputs | Hidden units | Outputs

1

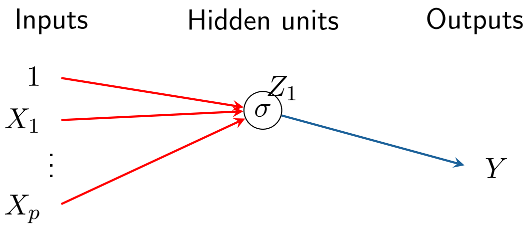
X_1

$\vdots$

X_p

Neural network - construction

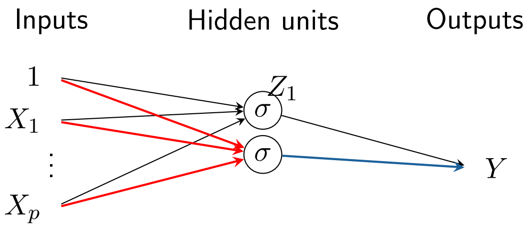
A NN is a sequential construction of several linear regression models.



$$Z_1 = \sigma\left(\beta_{01}^{(1)} + \sum_{j=1}^{p} \beta_{j1}^{(1)} X_j\right) \qquad Y = \beta_1^{(2)} Z_1$$

Neural network - construction

A NN is a sequential construction of several linear regression models.



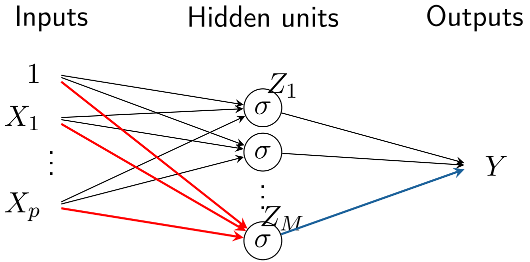
$$Z_1 = \sigma\left(\beta_{01}^{(1)} + \sum_{j=1}^{p} \beta_{j1}^{(1)} X_j\right)$$

$$Z_2 = \sigma\left(\beta_{02}^{(1)} + \sum_{j=1}^{p} \beta_{j2}^{(1)} X_j\right)$$

$$Y = \sum_{m=1}^{2} \beta_m^{(2)} Z_m$$

Neural network - construction

A NN is a sequential construction of several linear regression models.



$$Z_1 = \sigma\left(\beta_{01}^{(1)} + \sum_{j=1}^{p} \beta_{j1}^{(1)} X_j\right)$$

$$Z_2 = \sigma\left(\beta_{02}^{(1)} + \sum_{j=1}^{p} \beta_{j2}^{(1)} X_j\right)$$

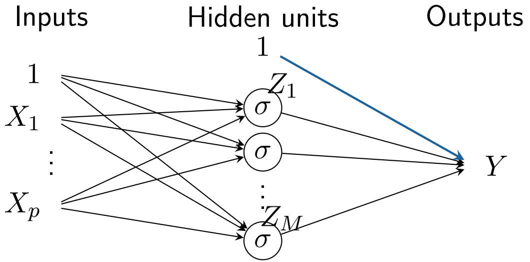
$$\vdots$$

$$Z_M = \sigma\left(\beta_{0M}^{(1)} + \sum_{j=1}^{p} \beta_{jM}^{(1)} X_j\right)$$

$$Y = \sum_{m=1}^{M} \beta_m^{(2)} Z_m$$

Neural network - construction

A NN is a sequential construction of several linear regression models.



$$Z_1 = \sigma\left(\beta_{01}^{(1)} + \sum_{j=1}^{p} \beta_{j1}^{(1)} X_j\right)$$

$$Z_2 = \sigma\left(\beta_{02}^{(1)} + \sum_{j=1}^{p} \beta_{j2}^{(1)} X_j\right)$$

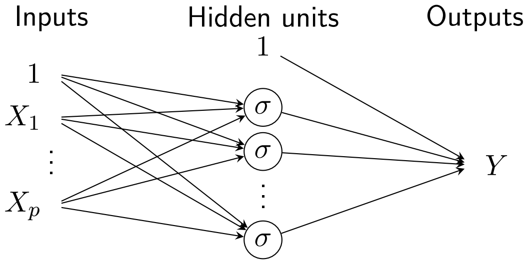
$$\vdots$$

$$Z_M = \sigma\left(\beta_{0M}^{(1)} + \sum_{j=1}^{p} \beta_{jM}^{(1)} X_j\right)$$

$$Y = \beta_0^{(2)} + \sum_{m=1}^{M} \beta_m^{(2)} Z_m$$

Neural network - construction

A NN is a sequential construction of several linear regression models.



$$Z = \sigma(W_1^\mathsf{T} X + b_1^\mathsf{T})$$

$$b_1 = \begin{bmatrix} \beta_{01}^{(1)} & \dots & \beta_{0M}^{(1)} \end{bmatrix}$$

$$W_1 = \begin{bmatrix} \beta_{01}^{(1)} & \dots & \beta_{0M}^{(1)} \\ \vdots & \dots & \vdots \\ \beta_{p1}^{(1)} & \dots & \beta_{pM}^{(1)} \end{bmatrix}$$

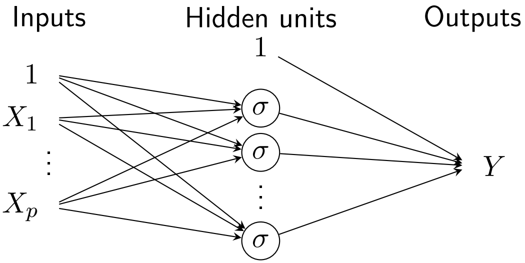
$$Y = \sigma(W_2^\mathsf{T} Z + b_2^\mathsf{T})$$

$$b_2 = \begin{bmatrix} \beta_0^{(1)} \end{bmatrix}$$

$$W_2 = \begin{bmatrix} \beta_0^{(2)} \\ \vdots \\ \beta_M^{(2)} \end{bmatrix}$$

Neural network - construction

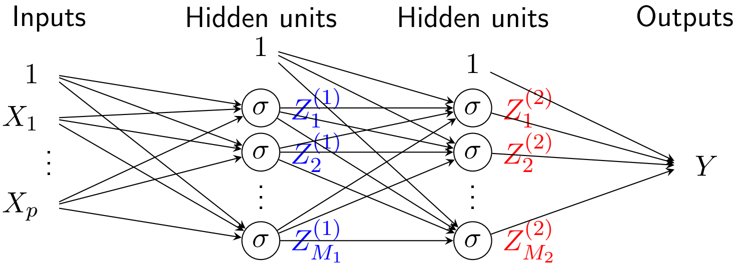
A NN is a sequential construction of several several linear regression models.



$$Z = \sigma(W_1^\mathsf{T} X + b_1^\mathsf{T})$$
$$Y = W_2^\mathsf{T} Z + b_2^\mathsf{T}$$

Neural network - construction

A NN is a sequential construction of several linear regression models.



$$Z^{(1)} = \sigma(W_1^\mathsf{T} X + b_1^\mathsf{T})$$
$$Z^{(2)} = \sigma(W_2^\mathsf{T} Z^{(1)} + b_2^\mathsf{T})$$
$$Y = W_3^\mathsf{T} Z^{(2)} + b_3^\mathsf{T}$$

The model learns better using a deep network (several layers) instead of a wide and shallow network. See why after the break!

Multi-layer neural networks

We can think of the neural network as a sequential/recursive construction of several generalized linear regressions.

Each layer in a multi-layer NN is modelled as

$$Z^{(l+1)} = \sigma\left(W_{(l+1)}^{\mathsf{T}} Z^{(l)} + b_{(l+1)}^{\mathsf{T}}\right),$$

starting with the input $z^{(0)} = X$. (The non-linearity operates element-wise.)

Multi-layer neural networks

We can think of the neural network as a sequential/recursive construction of several generalized linear regressions.

Each layer in a multi-layer NN is modelled as

$$Z^{(l+1)} = \sigma\left(W_{(l+1)}^{\mathsf{T}} Z^{(l)} + b_{(l+1)}^{\mathsf{T}}\right),$$

starting with the input $z^{(0)} = X$. (The non-linearity operates element-wise.)

Key aspect: The layers are **not** designed by human engineers, they are generated from (typically lots of) data using a learning procedure and lots of computations.

UPPSALA UNIVERSITET

2-layer Neural Network in matrix notation

Now, consider N training data points $\mathcal{T} = \{x_i, y_i\}_{i=1}^{N}$. We stack each data point i in a row (as we did in linear regression)

$$\begin{bmatrix} z_1^\mathsf{T} \\ z_2^\mathsf{T} \\ \vdots \\ z_N^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \sigma(x_1^\mathsf{T} W_1 + b_1) \\ \sigma(x_2^\mathsf{T} W_1 + b_1) \\ \vdots \\ \sigma(x_N^\mathsf{T} W_1 + b_1) \end{bmatrix} \qquad \begin{bmatrix} y_1^\mathsf{T} \\ y_2^\mathsf{T} \\ \vdots \\ y_N^\mathsf{T} \end{bmatrix} = \begin{bmatrix} z_1^\mathsf{T} W_2 + b_2 \\ z_2^\mathsf{T} W_2 + b_2 \\ \vdots \\ z_N^\mathsf{T} W_2 + b_2 \end{bmatrix}$$

This is how it is written in matrix form. $+b_1$, $+b_2$ and σ applied on every row.

$$\mathbf{Z} = \sigma(\mathbf{X} W_1 + b_1)$$
$$\hat{\mathbf{y}} = \mathbf{Z} W_2 + b_2$$

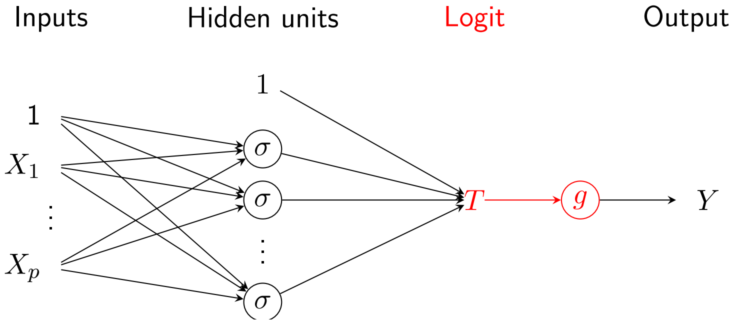
... and in Tensorflow (software package used in the lab)

```
# The model
Z <- tf$sigmoid(tf$matmul(X, W1) + b1)
Yhat = tf$matmul(Z, W2) + b2
```

NN for classification ($K = 2$ classes)

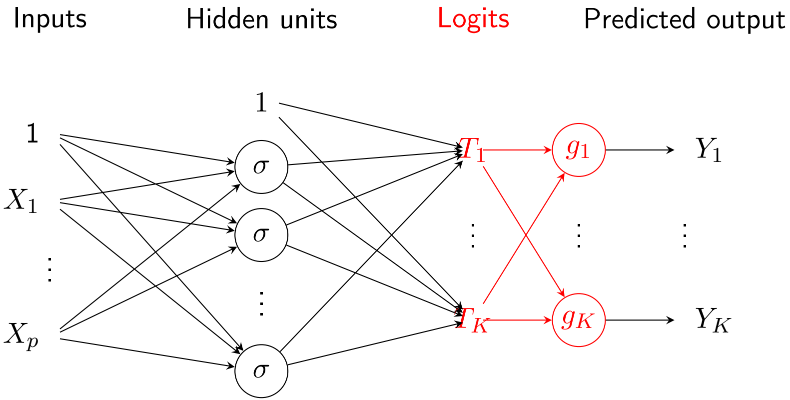
We can also use neural networks for classification. We use the logistic function as we did in logistic regression to map $T \in \mathbb{R}$ onto $Y \in [0, 1]$

$$\Pr(Y = 1|X) = f(T), \qquad f(T) = \frac{e^T}{1 + e^T}$$



NN for classification ($K > 2$ classes)

In the lab we will consider a classification problem with $K = 10$ classes.



We will not go into detail. Look at the preparatory exercises in the lab-pm!

Skin cancer – background

One recent result on the use of deep learning in medicine - Detecting skin cancer (February 2017)

Andre Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M. and Thrun, S. **Dermatologist-level classification of skin cancer with deep neural networks**. *Nature*, 542, 115–118, February, 2017.

Skin cancer – background

One recent result on the use of deep learning in medicine - Detecting skin cancer (February 2017)

Andre Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M. and Thrun, S. **Dermatologist-level classification of skin cancer with deep neural networks**. *Nature*, 542, 115–118, February, 2017.

Some background figures (from the US) on skin cancer:

- Melanomas represents less than 5% of all skin cancers, **but** accounts for 75% of all skin-cancer-related deaths.
- Early detection absolutely critical. Estimated 5-year survival rate for melanoma: Over 99% if detected in its earlier stages and 14% is detected in its later stages.

Skin cancer – task

Image copyright Nature (doi:10.1038/nature21056)

Skin cancer – taxonomy used

Skin cancer – solution (ultrabrief)

Start from a neural network trained on 1.28 million images (**transfer learning**).

Make minor modifications to this model, specializing to present situation.

Skin cancer – solution (ultrabrief)

Start from a neural network trained on 1.28 million images (**transfer learning**).

Make minor modifications to this model, specializing to present situation.

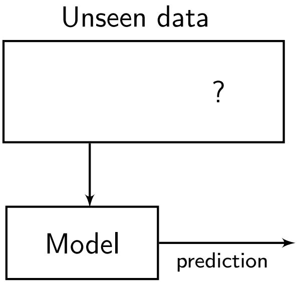
Learn new model parameters using $129\,450$ clinical images (~ 100 times more images than any previous study).

Skin cancer – solution (ultrabrief)

Start from a neural network trained on 1.28 million images (**transfer learning**).

Make minor modifications to this model, specializing to present situation.

Learn new model parameters using $129\,450$ clinical images (~ 100 times more images than any previous study).



Skin cancer – indication of the results

$$\text{sensitivity} = \frac{\text{true positive}}{\text{positive}} \qquad \text{specificity} = \frac{\text{true negative}}{\text{negative}}$$

UPPSALA UNIVERSITET

Skin cancer – indication of the results

$$\text{sensitivity} = \frac{\text{true positive}}{\text{positive}} \qquad \text{specificity} = \frac{\text{true negative}}{\text{negative}}$$

UPPSALA UNIVERSITET

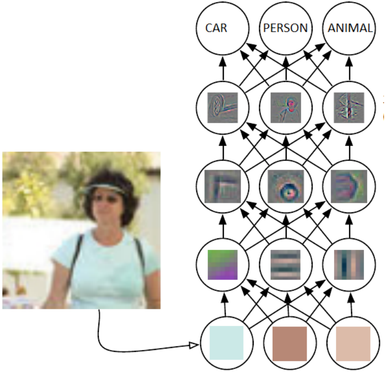
Why do deep neural networks work so well?

Example: Image classification

Input: pixels of an **image**

Output: **object identity**

- 1 megapixel (black/white) $\Rightarrow$ $2^{1'000'000}$ possible images!
- A **deep neural network** can solve this with a few million parameters!

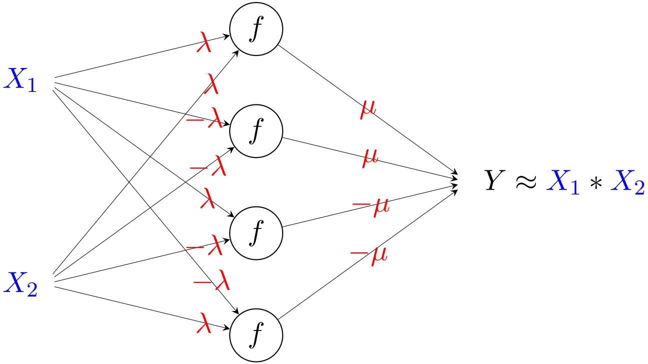


How can deep neural networks work so well?

Why neural networks?

Continuous multiplication gate

A neural network with only four hidden units can model multiplication of two numbers arbitrarily well.



If we choose $\mu = \frac{1}{4\lambda^2 f''(0)}$ then $Y \to X_1 * X_2$ when $\lambda \to 0$.

Henry W. Lin and Max Tegmark. (2016) **Why does deep and cheap learning work so well?**, *arXiv*

A regression example

Input: $X \in \mathbb{R}^{1000}$
Output: $Y \in \mathbb{R}$
Task: Model a quadratic relationship between Y and X

UPPSALA UNIVERSITET

A regression example

Input: $X \in \mathbb{R}^{1000}$
Output: $Y \in \mathbb{R}$
Task: Model a quadratic relationship between Y and X

Linear regression

$$Y = X_1 X_1 \beta_{1,1} + X_1 X_2 \beta_{1,2} + \cdots + X_{1000} X_{1000} \beta_{1000,1000} = \bar{X}^\mathsf{T} \beta$$

where

$$\bar{X} = \begin{bmatrix} X_1 X_1 & X_1 X_2 & \ldots & X_{1000} X_{1000} \end{bmatrix}^\mathsf{T}$$
$$\beta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \ldots & \beta_{1000,1000} \end{bmatrix}^\mathsf{T}$$

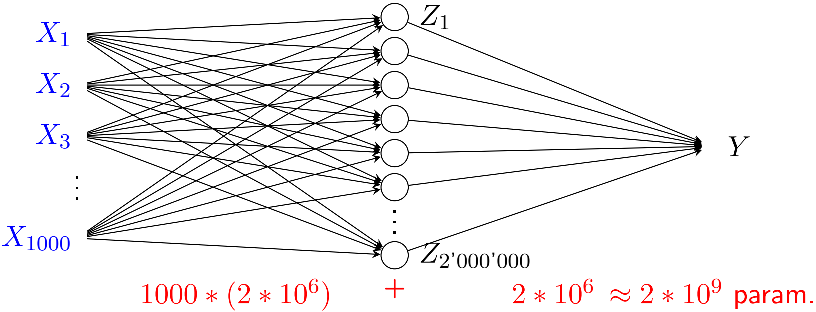
Requires $\approx \frac{1'000 * 1'000}{2} = 500'000$ parameters!

A regression example

Input: $X \in \mathbb{R}^{1000}$
Output: $Y \in \mathbb{R}$
Task: Model a quadratic relationship between Y and X

Neural network

To model all products with a neural network we would need $4 * 500'000 = 2 * 10^6$ hidden units and hence 2 billion parameters...



A regression example (cont.)

Input: $X \in \mathbb{R}^{1000}$
Output: $Y \in \mathbb{R}$
Task: Model a quadratic relationship between X and Y
Assume that only 10 of the regressors $X_i X_j$ are of importance

Linear regression

$$Y = X_1 X_1 \beta_{1,1} + X_1 X_2 \beta_{1,2} + \cdots + X_{1000} X_{1000} \beta_{1000,1000} = \bar{X}^\mathsf{T} \beta$$

where

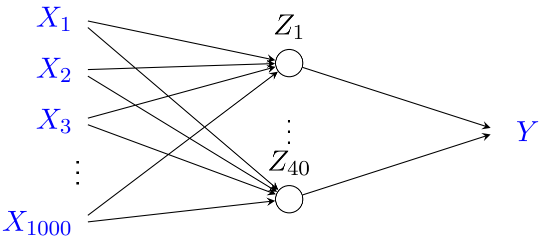
$$\bar{X} = \begin{bmatrix} X_1 X_1 & X_1 X_2 & \dots & X_{1000} X_{1000} \end{bmatrix}^\mathsf{T}$$
$$\beta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \dots & \beta_{1000,1000} \end{bmatrix}^\mathsf{T}$$

You probably want to regularize, but 500’000 parameters are still required!

A regression example (cont.)

Input: $X \in \mathbb{R}^{1000}$
Output: $Y \in \mathbb{R}$
Task: Model a quadratic relationship between X and Y
Assume that only 10 of the regressors $X_i X_j$ are of importance

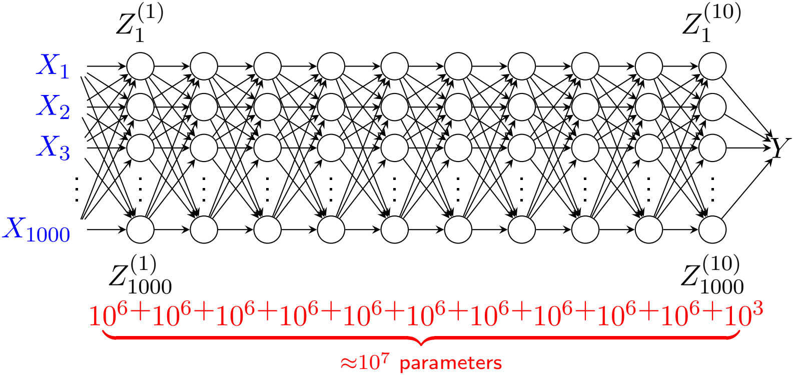
Neural network
To model 10 products with a neural network we would need 4*10 hidden units, i.e. leading to only $\approx$ 40'000 parameters!



1000*40 + 40 = 40'040

Why deep? - A regression example

- Consider the same example. Now we want a model with complexity corresponding to polynomials of degree 1'000.
- Keep 250 products in each layer ⇒ $250*4=1'000$ hidden units.



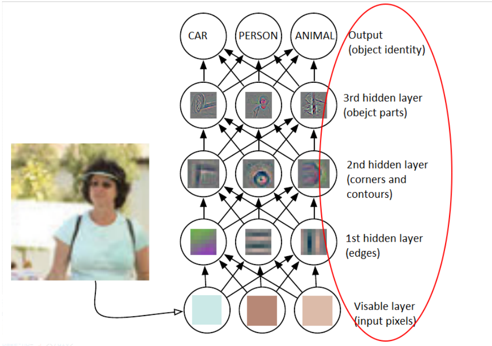
Linear regression would require $\approx \frac{1000^{1000}}{1000!}$ parameters to model such a relationship...

Why deep? - Image classification

Example: Image classification

Input: pixels of an image
Output: object identity
Each hidden layer extracts increasingly abstract features.

Human interpretation



Zeiler, M. D. and Fergus, R. **Visualizing and understanding convolutional networks** *Computer Vision - ECCV* (2014).

Some comments - Why now?

Neural networks have been around for more than fifty years. Why have they become so popular now (again)?

To solve really interesting problems you need:

1. Efficient learning **algorithms**
2. Efficient computational **hardware**
3. A lot of labeled **data**!

These three factors have not been fulfilled to a satisfactory level until the last 5-10 years.

UPPSALA UNIVERSITET

Some pointers

A book has recently been published

I. Goodfellow, Y. Bengio and A. Courville **Deep learning**

http://www.deeplearningbook.org/

UPPSALA UNIVERSITET

Some pointers

A book has recently been published

I. Goodfellow, Y. Bengio and A. Courville **Deep learning**

http://www.deeplearningbook.org/

A well written and timely introduction:

LeCun, Y., Bengio, Y., and Hinton, G. (2015) **Deep learning**, *Nature*, 521(7553), 436–444.

UPPSALA UNIVERSITET

Some pointers

A book has recently been published

I. Goodfellow, Y. Bengio and A. Courville **Deep learning**

http://www.deeplearningbook.org/

A well written and timely introduction:

LeCun, Y., Bengio, Y., and Hinton, G. (2015) **Deep learning**, *Nature*, 521(7553), 436–444.

You will also find more material than you can possibly want here

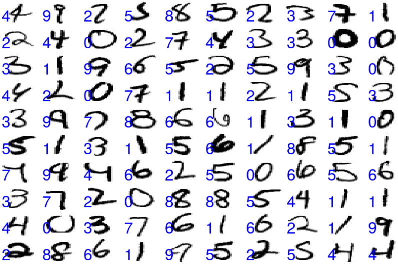
http://deeplearning.net/

UPPSALA UNIVERSITET

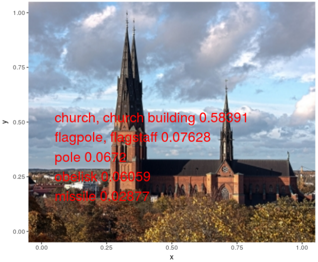
The lab

Topic: Image classification with neural networks

Task 1
Classification of hand-written digits



Task 2
Real world image classification



- The lab-pm is available from the course homepage.
- Read Section 2 and do the preparatory exercises in Section 3 *before* the lab

Summary

A **neural network (NN)** is a nonlinear function $Y = f_\theta(X)$ from an input X to a predicted output Y parameterized by parameters θ.

We can think of an NN as a sequential/recursive construction of several generalized linear regressions.

Deep learning refers to learning NNs with several hidden layers. Allows for data-driven models that automatically learns rep. of data (features) with multiple layers of abstraction.

A deep NN is very **parameter efficient** when modelling high-dimensional, complex data.