Introduction to Deep Learning

Tapani Raiko

Aalto University

17 August 2015

Aalto University

About me (Tapani Raiko)

- MSc 2001 Helsinki University of Technology
  Deep learning (Hierarchical Nonlinear Factor Analysis)
  Erkki Oja, Juha Karhunen, Harri Valpola
- DSc 2006, Same group
  Variational Bayesian modelling, relational models
- ZenRobotics Ltd 2009-2014 (primary job 2013)
- Assistant prof 2014- Aalto University
- Research visits:
  - Luc De Raedt, Freiburg, 2001-2002
  - Yann Lecun, New York, 2010
  - Geoffrey Hinton, Toronto, 2012
  - Yoshua Bengio, Montreal, 2014

Aalto University

Table of Contents

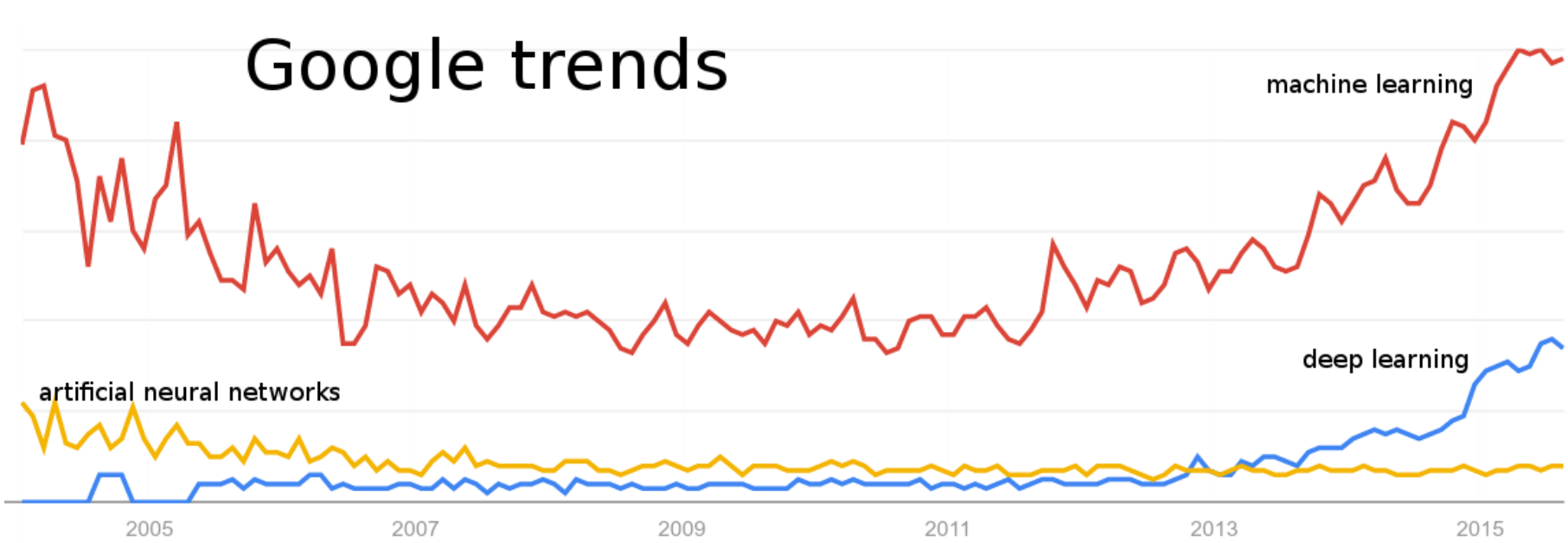
Aalto University

Deep learning is hot in academy

- "Deep learning ... dramatically improved the state-of-the-art in speech recognition, visual object recognition..." (LeCun et al., Nature, 2015)
- "...bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent..." (Mnih et al., Nature, 2015)
- "Knowing the sequence specificities of DNA- and RNA-binding proteins is essential ... deep learning outperforms other state-of-the-art methods" (Alipanahi et al., Nature Biotechnology, 2015)

Aalto University

Deep learning is hot in industry



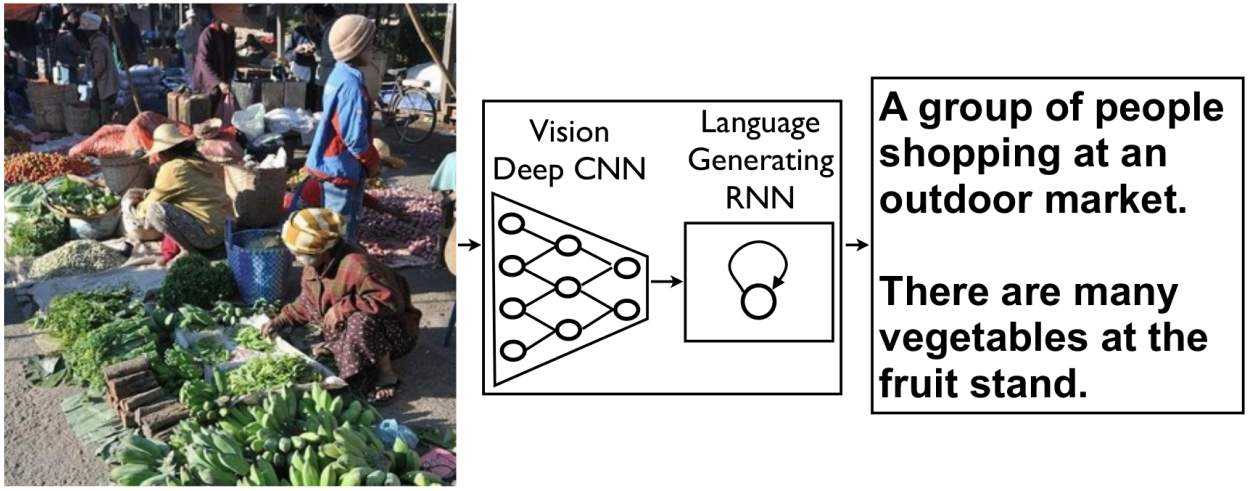
Google acquired startup DeepMind for $500M in 2014.
Also racing: Facebook, Baidu, IBM, Amazon, Samsung, Nvidia, Nokia, ...

Aalto University

Deep learning is changing the world

Aalto University

Caption generation (Vinyals et al., 2015)



A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

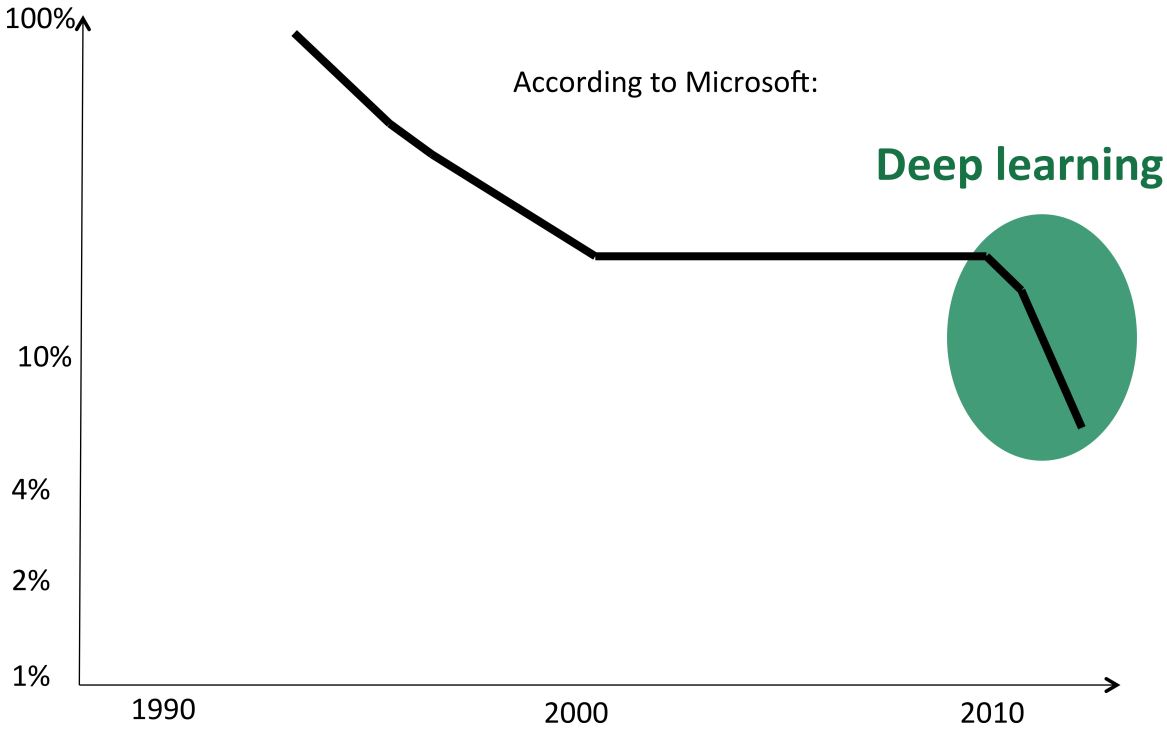
A **stop** sign is on a road with a mountain in the background

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

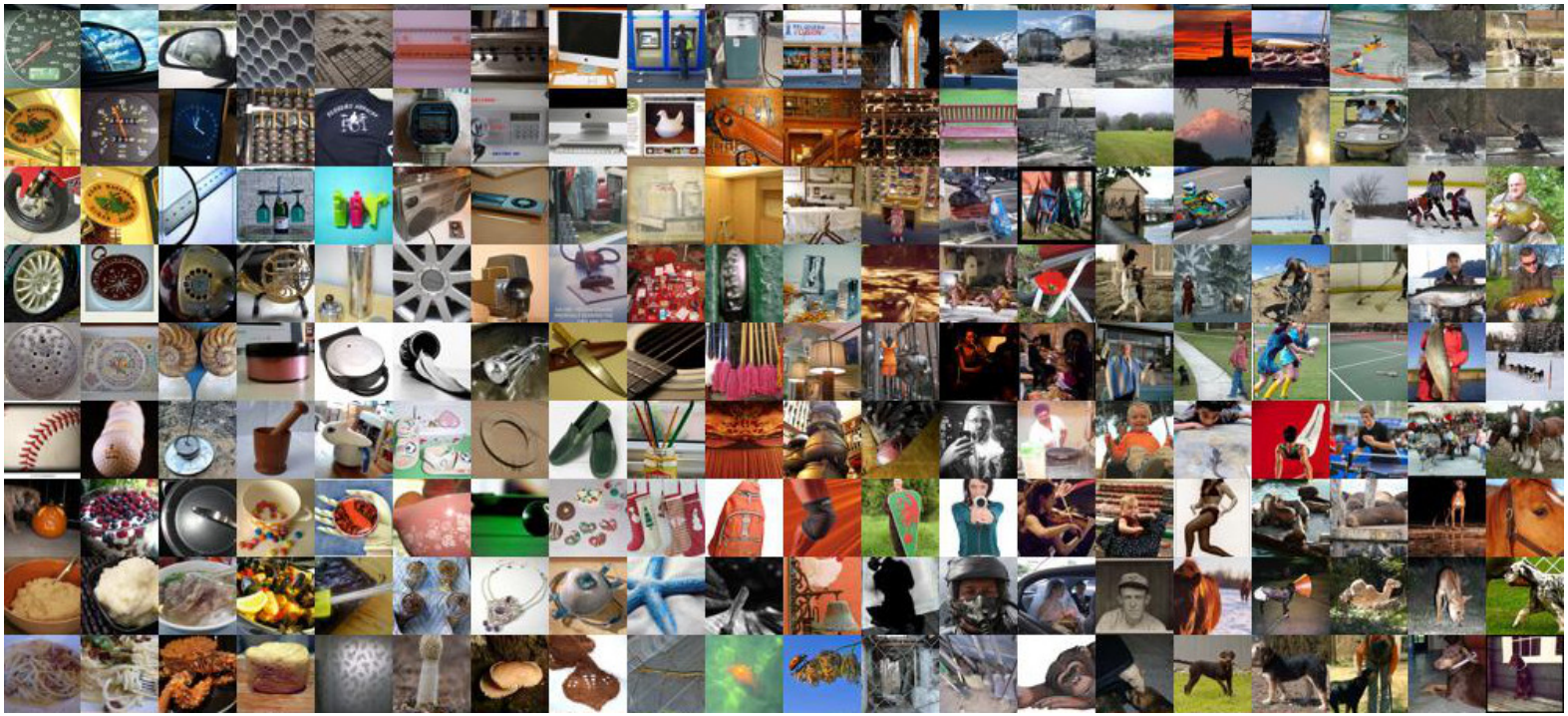
Speech recognition breakthrough



Plot from Yoshua Bengio

Aalto University

Imagenet classification challenge



Yearly competition in computer vision.
Krizhevsky et al. (2012) won with huge margin (16.4% error compared to 26.2%) by deep learning.
Soon everyone started using deep learning.

Aalto University

Representation learning

Traditional way:
Data $\rightarrow$ Feature engineering $\rightarrow$ Machine learning

- Feature selection
- Feature extraction (e.g. PCA)
- Feature construction (e.g. SIFT)

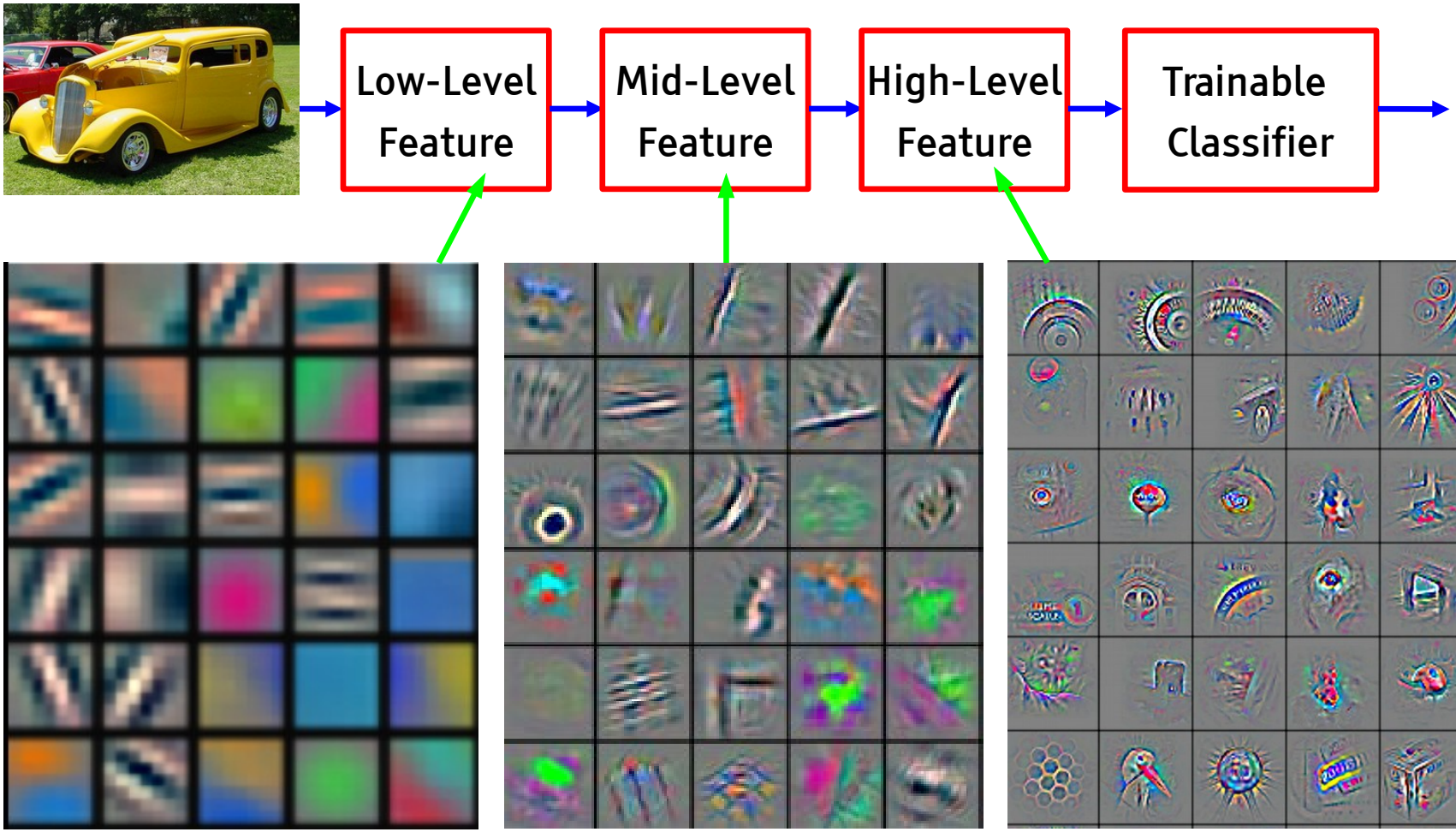
Deep learning way:
Data $\rightarrow$ End-to-end learning

Aalto University

Deep Learning = Learning Hierarchical Representations

Y LeCun

- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Review article, May 2015:

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

nature

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Book, draft available online:

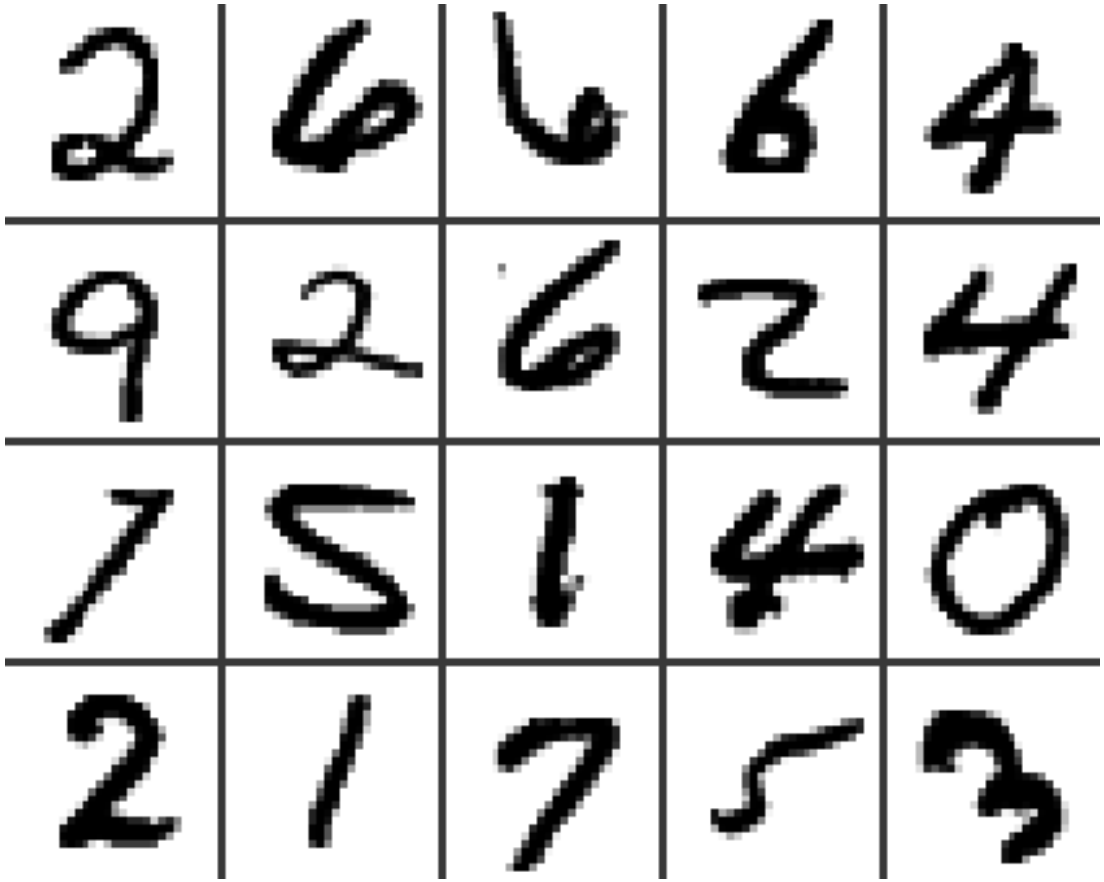
Portal: `deeplearning.net`

Aalto University

Table of Contents

Aalto University

Example: MNIST handwritten digits



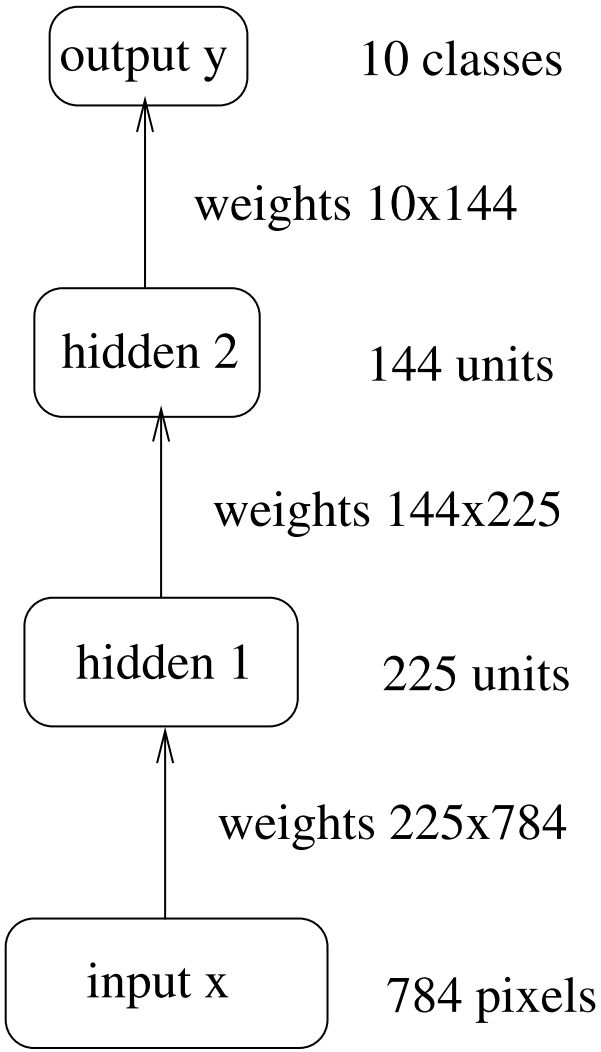
Train a network to classify 28×28 images.

Data: 60000 input images $\mathbf{x}(n)$ and labels $y(n)$.

Example model gives around 1.2% test error.

Aalto University

Example Network



$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$

$$\mathbf{h}^{(2)} = \text{relu}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$

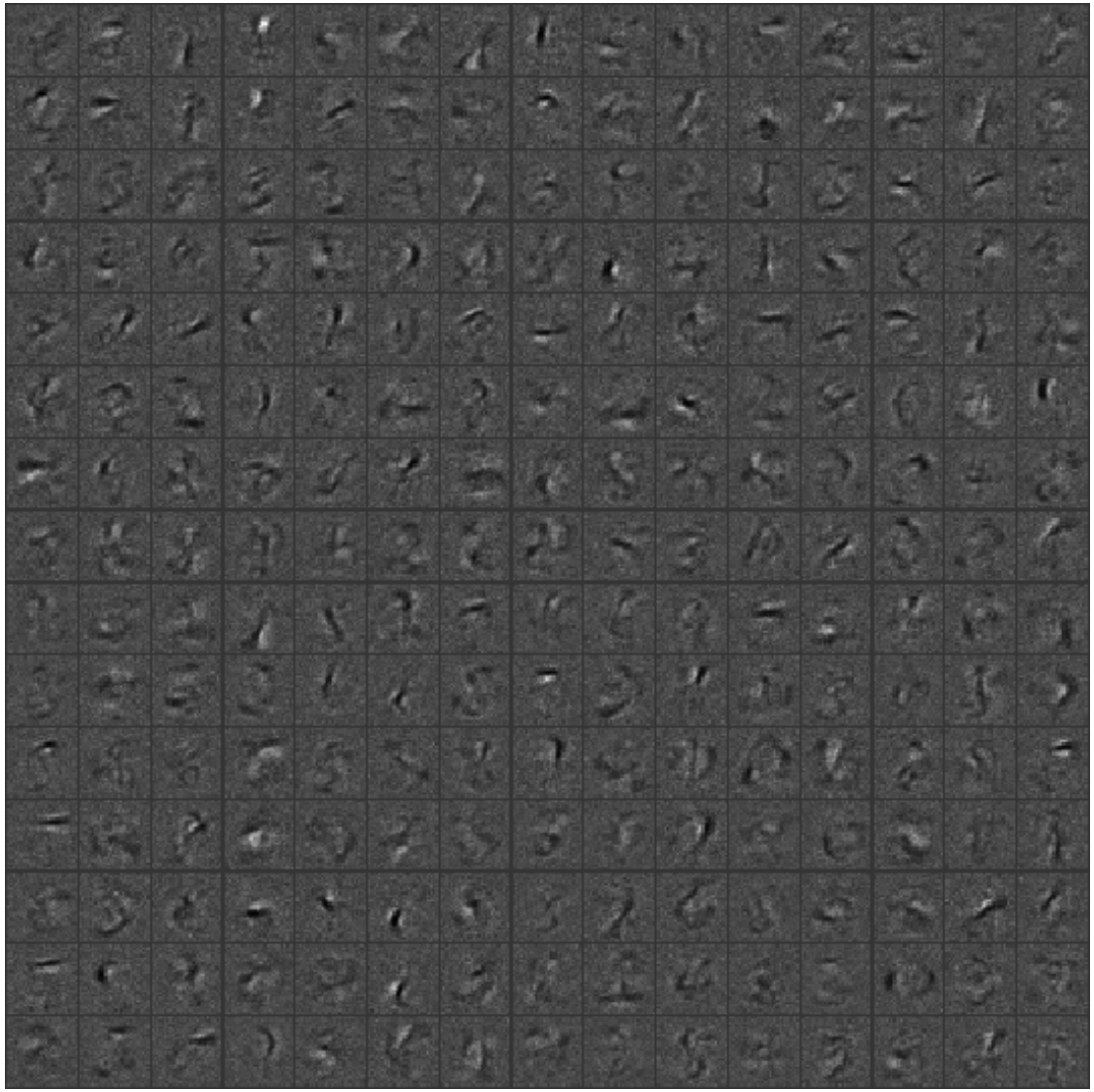
$$\mathbf{h}^{(1)} = \text{relu}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$\text{relu}(z) = max(0, z)$$

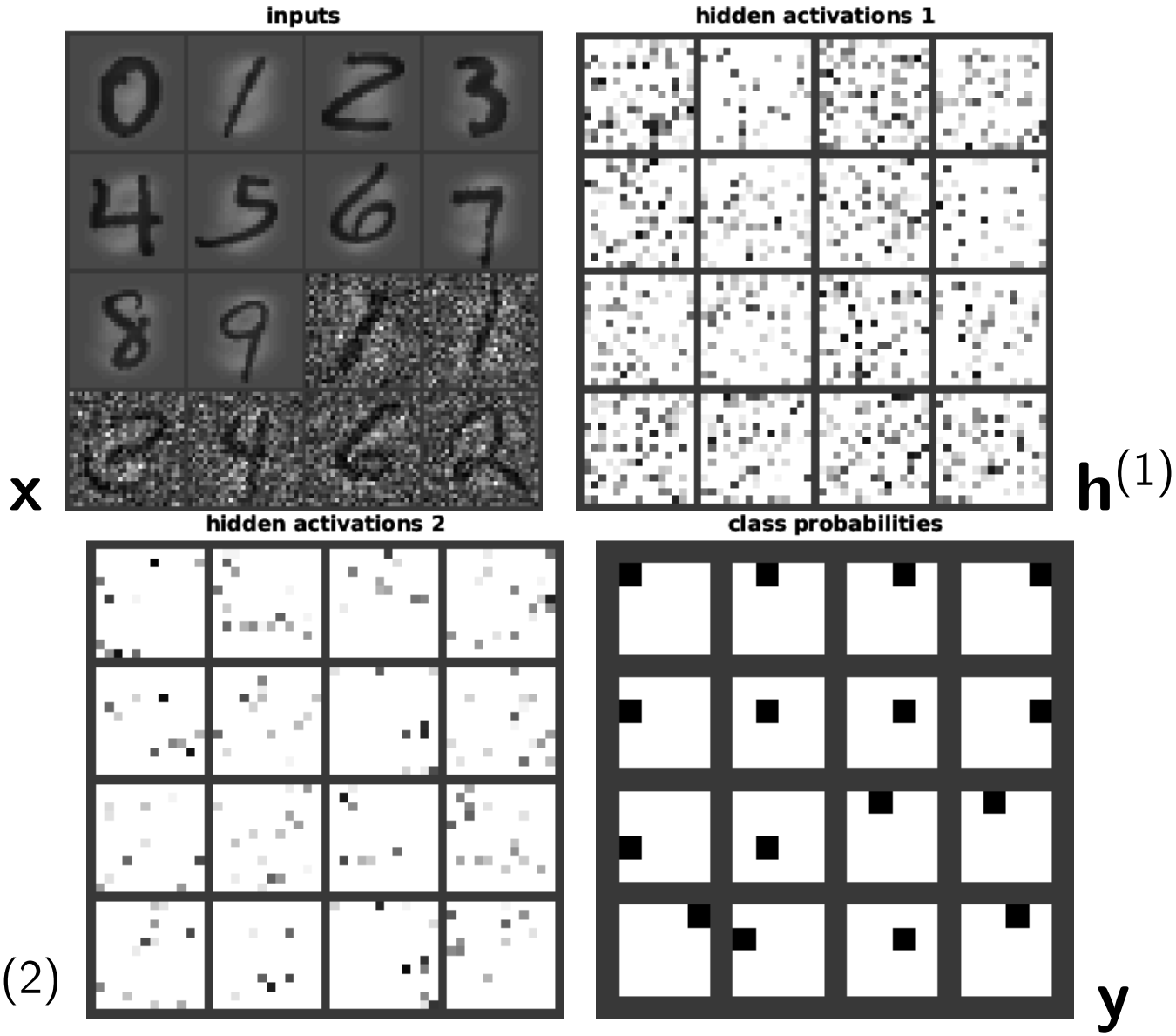
Aalto University

Weight matrix $\mathbf{W}^{(1)}$ size 225×784



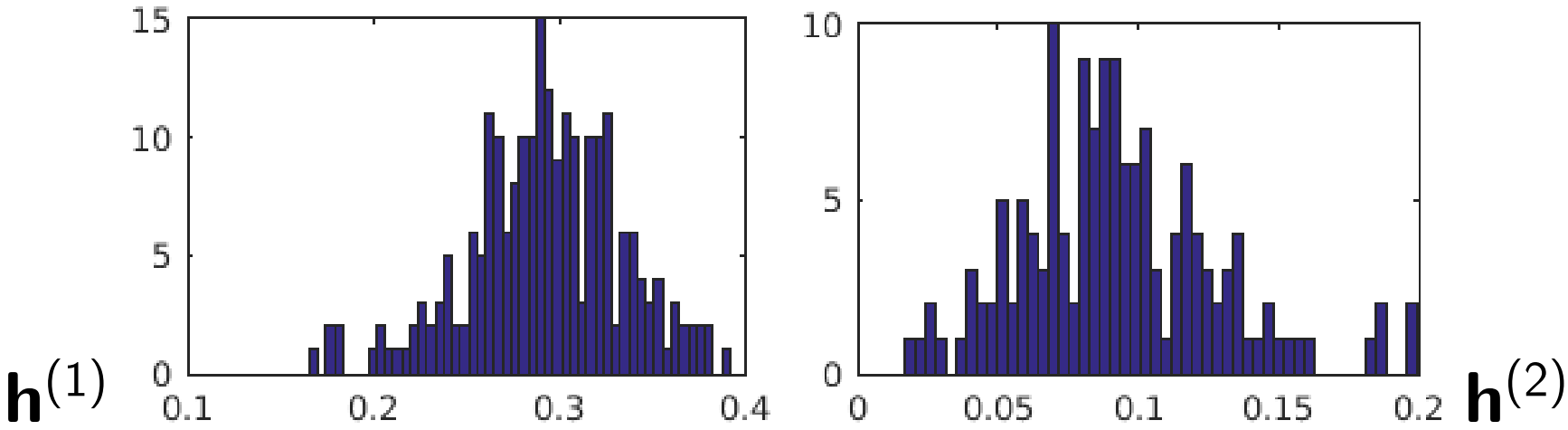
Aalto University

Signals $\mathbf{x} \rightarrow \mathbf{h}^{(1)} \rightarrow \mathbf{h}^{(2)} \rightarrow \mathbf{y}$



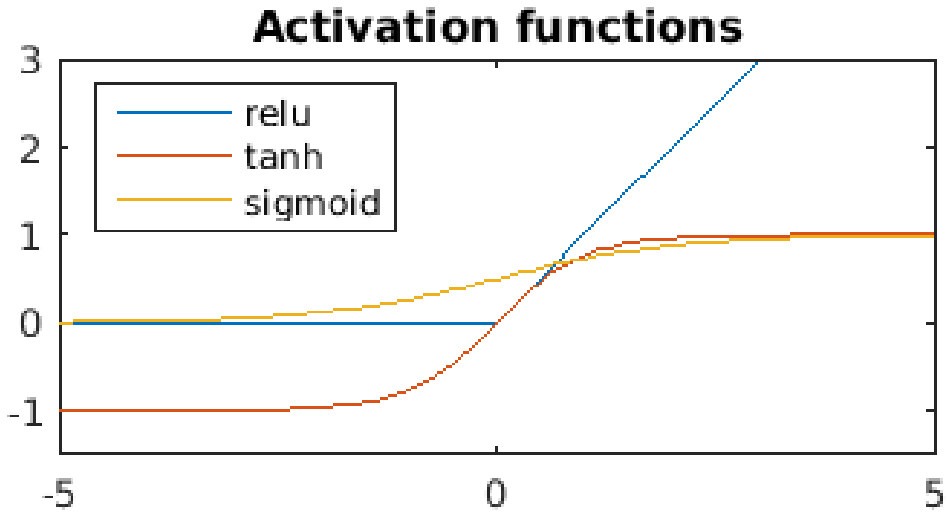
Aalto University

On sparsity



How often $h_i > 0$? Histogram over units i.
(Sometimes units become completely dead.)

On activation functions



$\text{relu}(z) = \max(0, z)$ is replacing old sigmoid and tanh.
Note that identity function would lead into:

$$\begin{aligned}
\mathbf{h}^{(2)} &= \mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)} \\
&= \mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} \\
&= (\mathbf{W}^{(2)}\mathbf{W}^{(1)})\mathbf{x} + (\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}) \\
&= \mathbf{W}'\mathbf{x} + \mathbf{b}'
\end{aligned}$$

Aalto University

Training criterion

Find parameters

$$\boldsymbol{\theta} = \{\mathbf{W}^{(L)}, \mathbf{b}^{(L)}\}$$

that minimize expected negative log-likelihood:

$$C = \mathbb{E}_{\text{data}} \left[-\log P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) \right].$$

Learning becomes optimization.

Say we have a true distribution $P(\mathbf{y} \mid \mathbf{x})$ and we would like to find a model $Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ that matches P. Let us study how maximizing expected negative log-likelihood $C = \mathbb{E}_P [-\log Q]$ works as a learning criterion.

$$\boldsymbol{\theta}^* = \arg\min_{\theta} C(\theta) = \arg\min_{\theta} \mathbb{E}_{P(\mathbf{y}|\mathbf{x})} [-\log Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})] .$$

Let us assume that there is a $\boldsymbol{\theta}^*$ for which $Q(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}^*) = P(\mathbf{y}|\mathbf{x})$. We can note that the gradient at $\boldsymbol{\theta}^*$

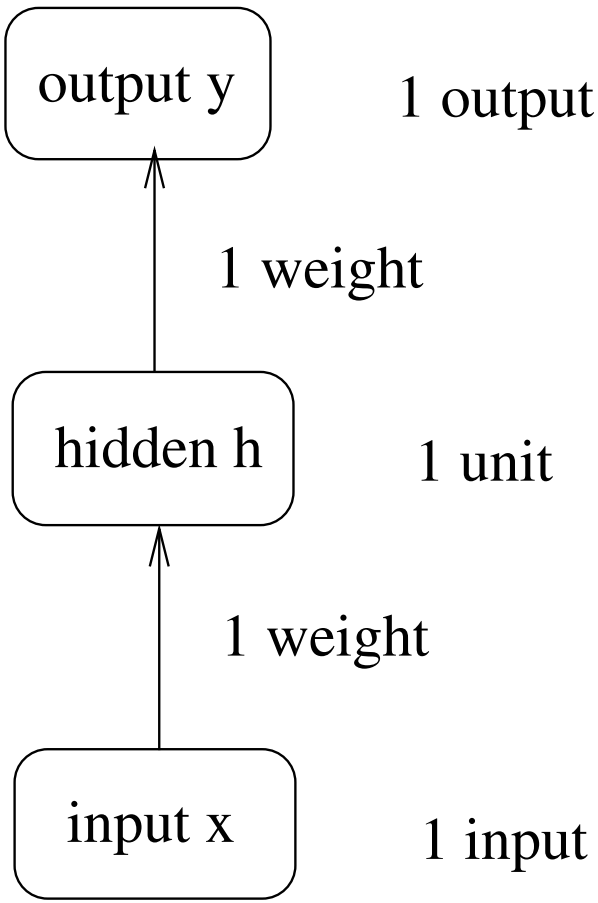
$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_{P(\mathbf{y}|\mathbf{x})} \left[\log Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*) \right] \\
&= \mathbb{E}_{P(\mathbf{y}|\mathbf{x})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*) \right] \\
&= \int P(\mathbf{y} \mid \mathbf{x}) \frac{\frac{\partial}{\partial \boldsymbol{\theta}} Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*)}{Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*)} \mathrm{d}\mathbf{y} \\
&= \int \frac{\partial}{\partial \boldsymbol{\theta}} Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*) \mathrm{d}\mathbf{y} \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \int Q(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*) \mathrm{d}\mathbf{y} = \frac{\partial}{\partial \theta} 1 = 0
\end{aligned}$$

becomes zero, that is, the learning converges when $Q = P$. Therefore the expected log-likelihood is a reasonable training criterion.

Table of Contents

Aalto University

Tiny Example

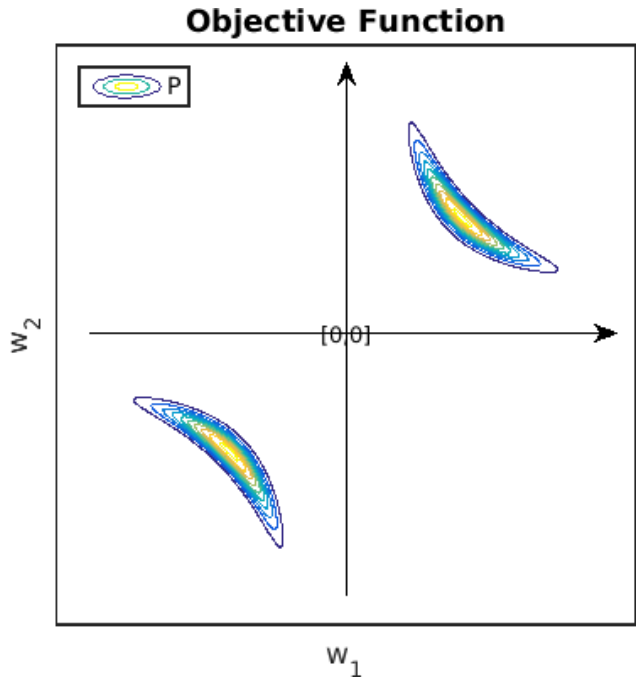


$$y \sim \mathcal{N}(w_2 h, 1)$$
$$h = w_1 x$$

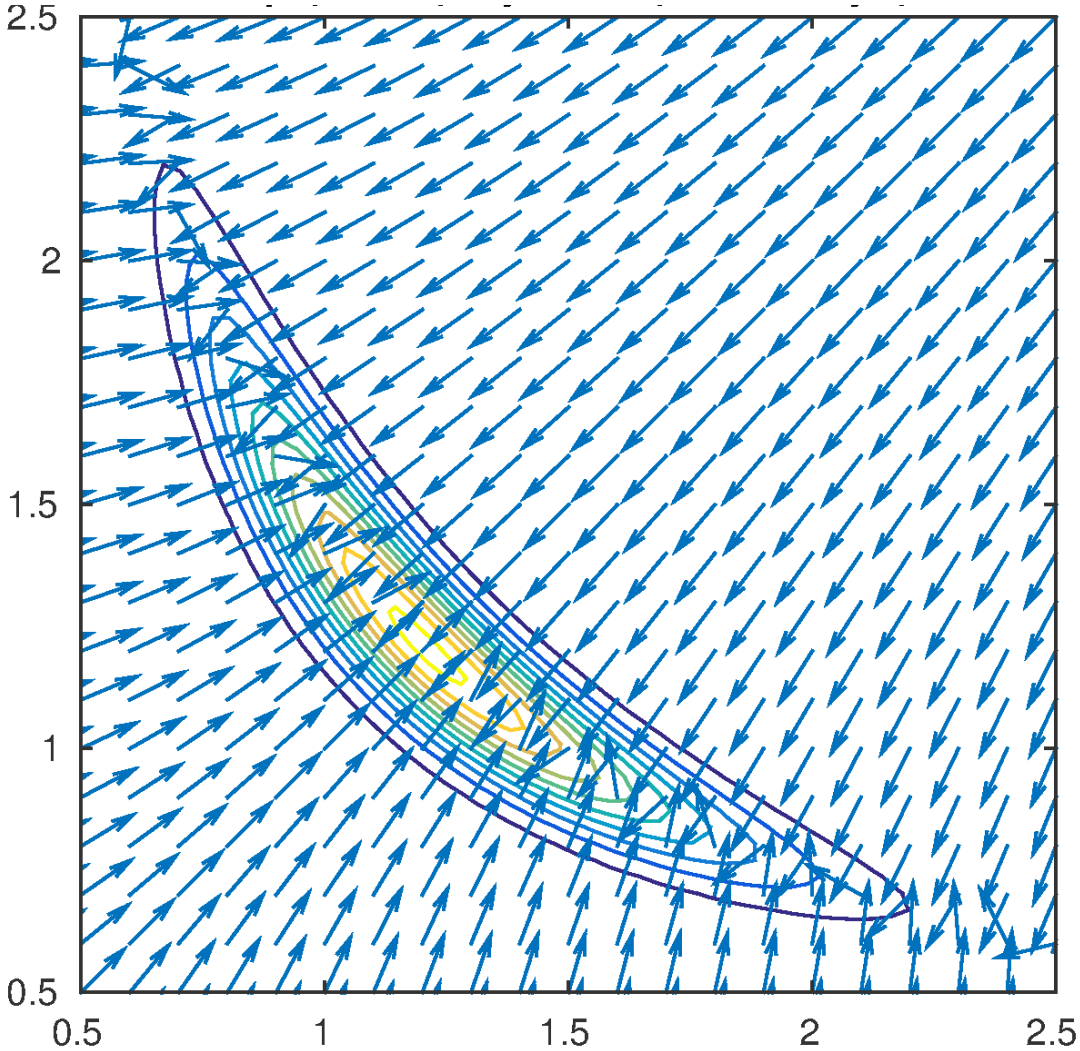
Data "set": $\{x = 1, y = 1.5\}$

Some weight decay.

$$C = (w_1 w_2 - 1.5)^2 + 0.04(w_1^2 + w_2^2)$$



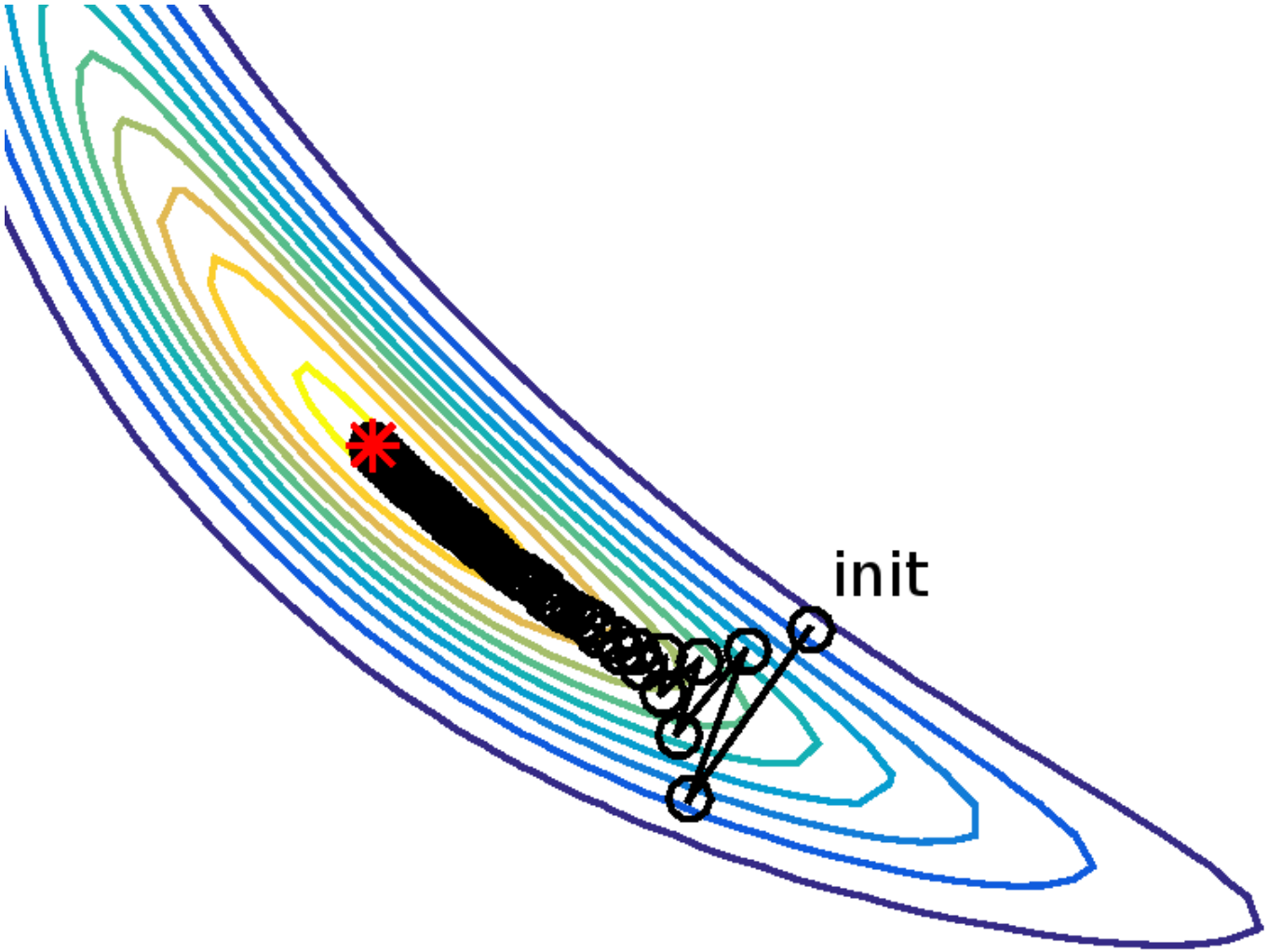
Gradient $\mathbf{g} = \nabla_\theta C(\theta) = \begin{pmatrix} \frac{\partial C}{\partial \theta_1} \\ \vdots \\ \frac{\partial C}{\partial \theta_n} \end{pmatrix}$



Aalto University

Gradient descent, $\eta_k = 0.25$ ($\rightarrow$ too slow)

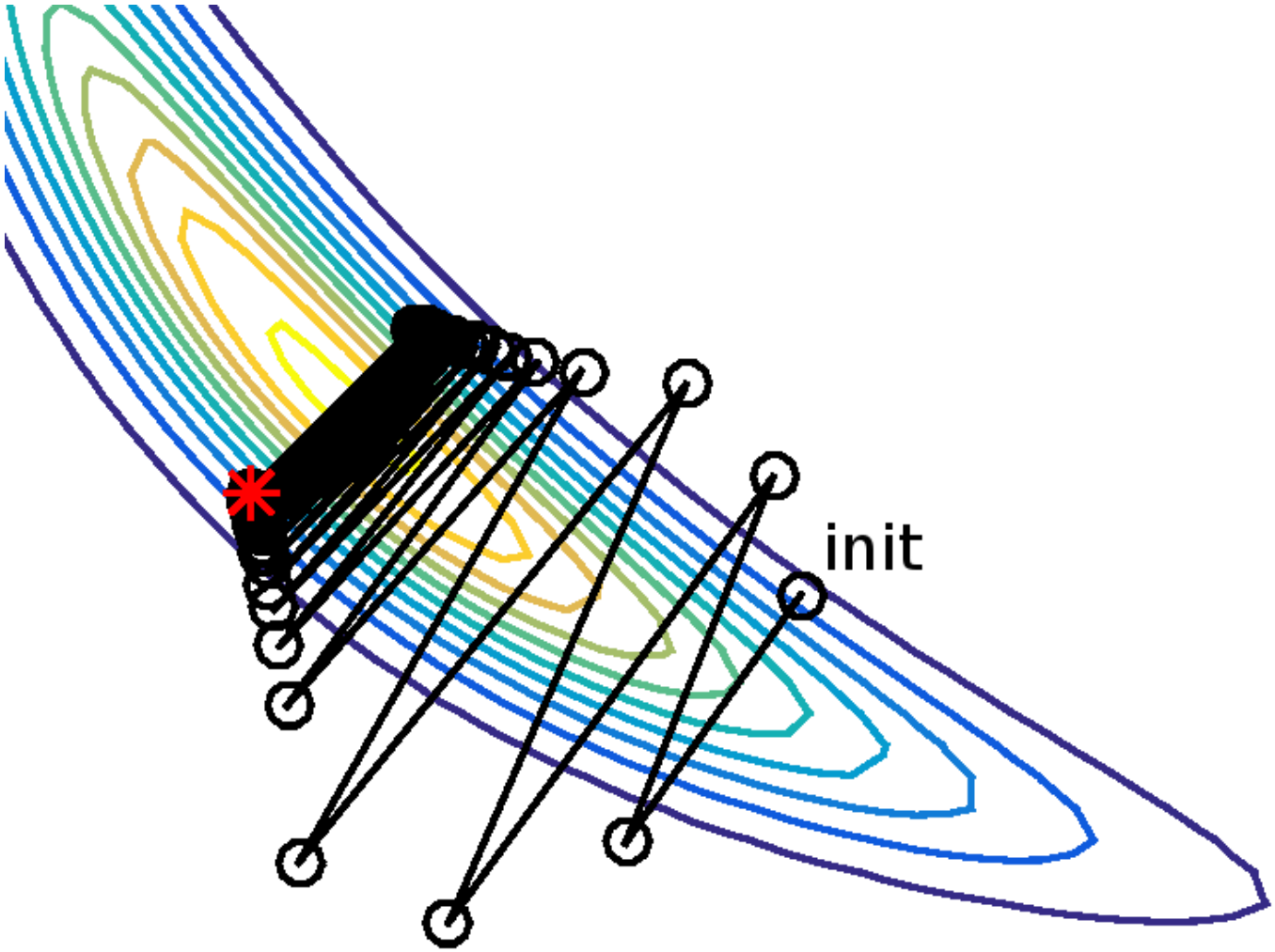
$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{g}_k$, iteration k, stepsize (or learning rate) η_k



Aalto University

Gradient descent, $\eta_k = 0.35$ ($\rightarrow$ oscillates)

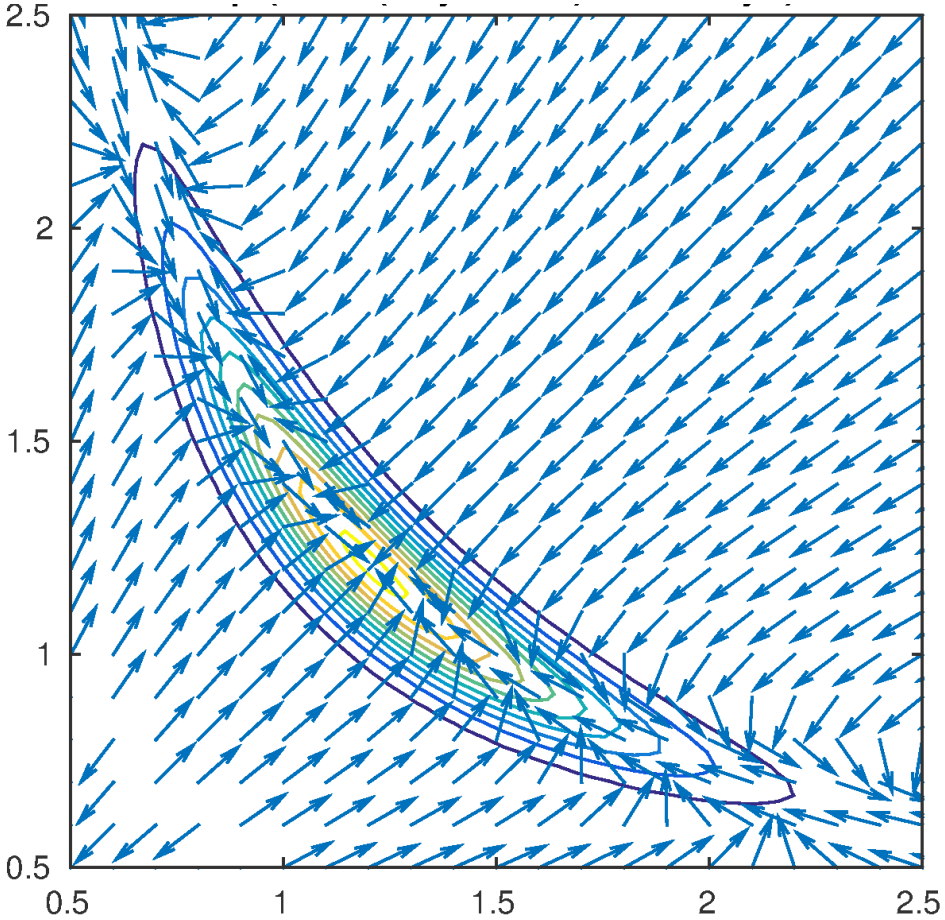
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{g}_k$$



Aalto University

Newton's method, too complex

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_k^{-1}\mathbf{g}_k, \quad \mathbf{H} = \begin{pmatrix} \frac{\partial^2 C}{\partial\theta_1\partial\theta_1} & \cdots & \frac{\partial^2 C}{\partial\theta_1\partial\theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 C}{\partial\theta_n\partial\theta_1} & \cdots & \frac{\partial^2 C}{\partial\theta_n\partial\theta_n} \end{pmatrix}$$

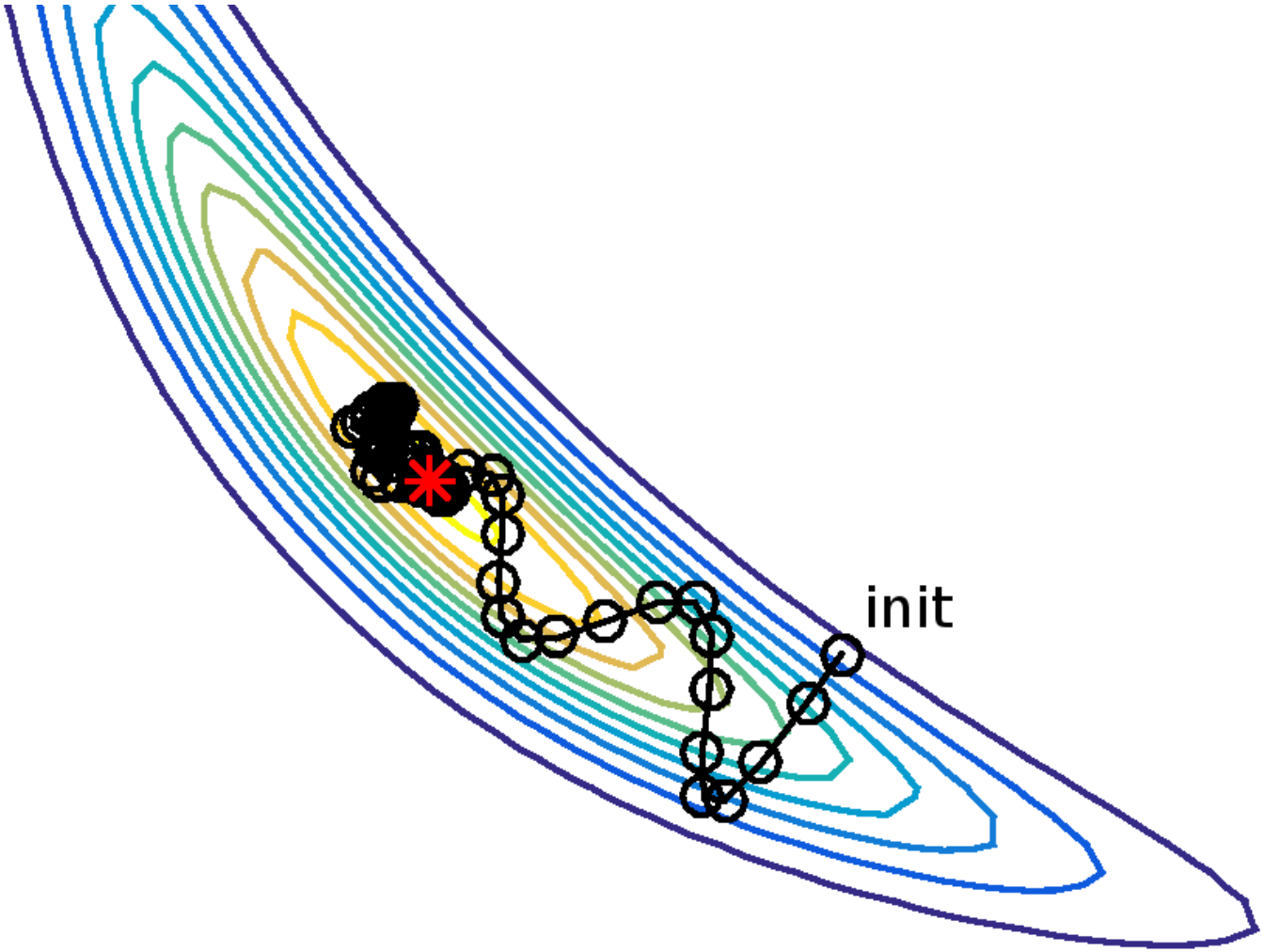


Less oscillations.
Points to the wrong direction in places (solvable).
Computational complexity: $\#\text{params}^3$ (prohibitive).
There are approximations, but not very popular.

Aalto University

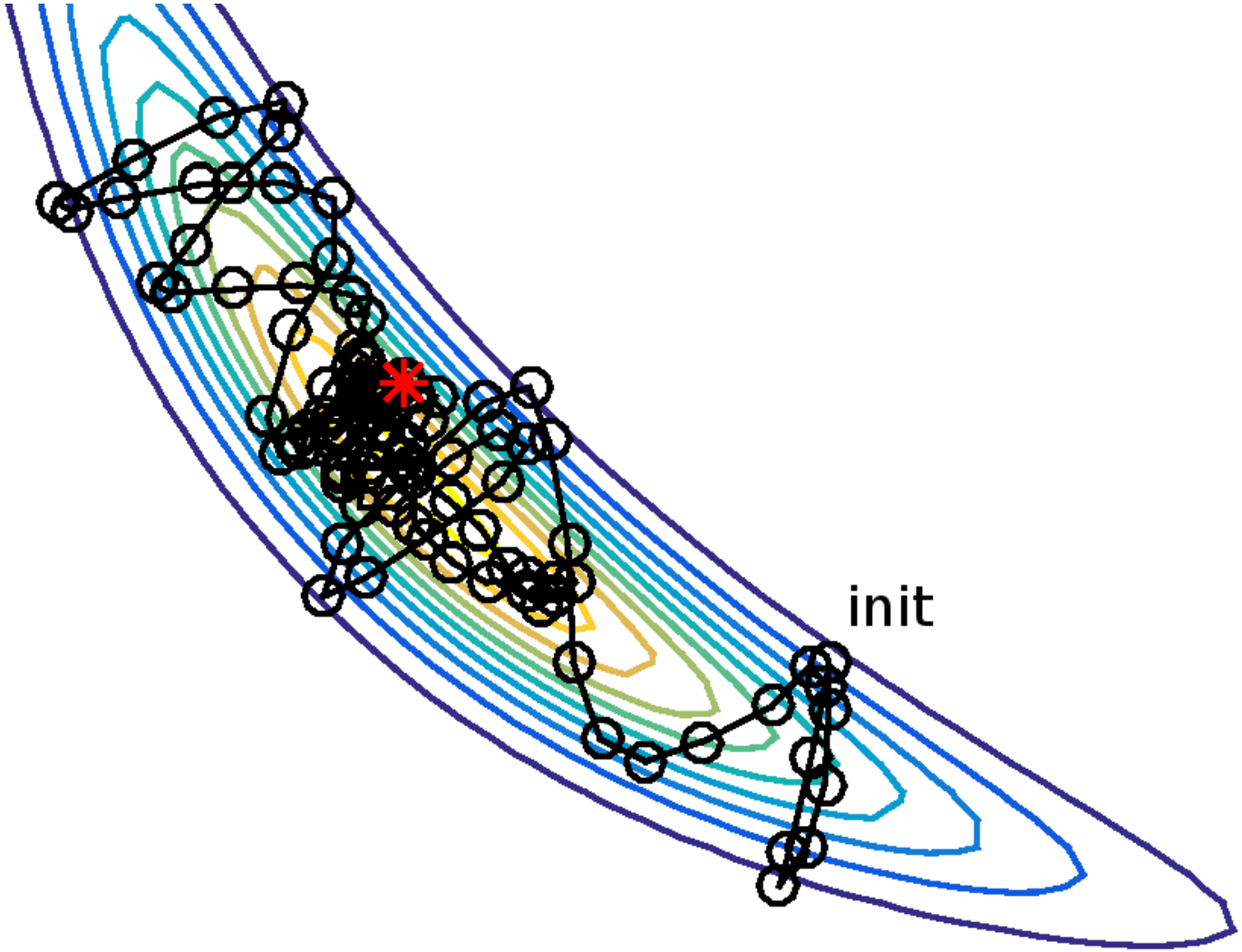
Momentum method (Polyak, 1964)

$$\mathbf{m}_{k+1} = \alpha \mathbf{m}_k - \eta_k \mathbf{g}_k$$
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{m}_{k+1}$$



Aalto University

Momentum method with noisy gradient



Aalto University

Mini-batch training

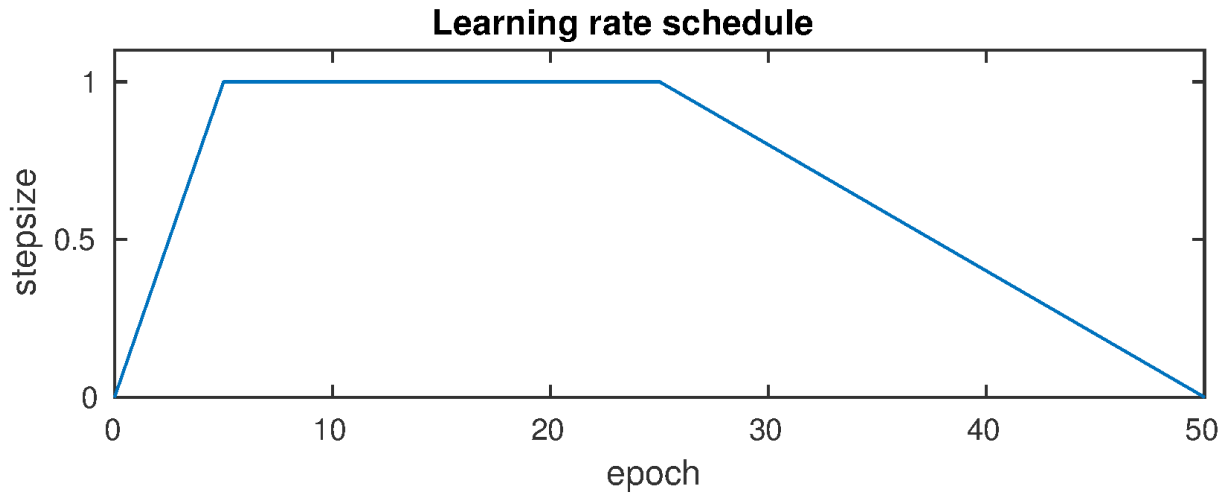
No need to have an accurate estimate of $\mathbf{g}$.

Use only a small batch of training data at once.

Leads into many updates per epoch (=seeing data once).

E.g. 600 updates with 100 samples per epoch in MNIST.

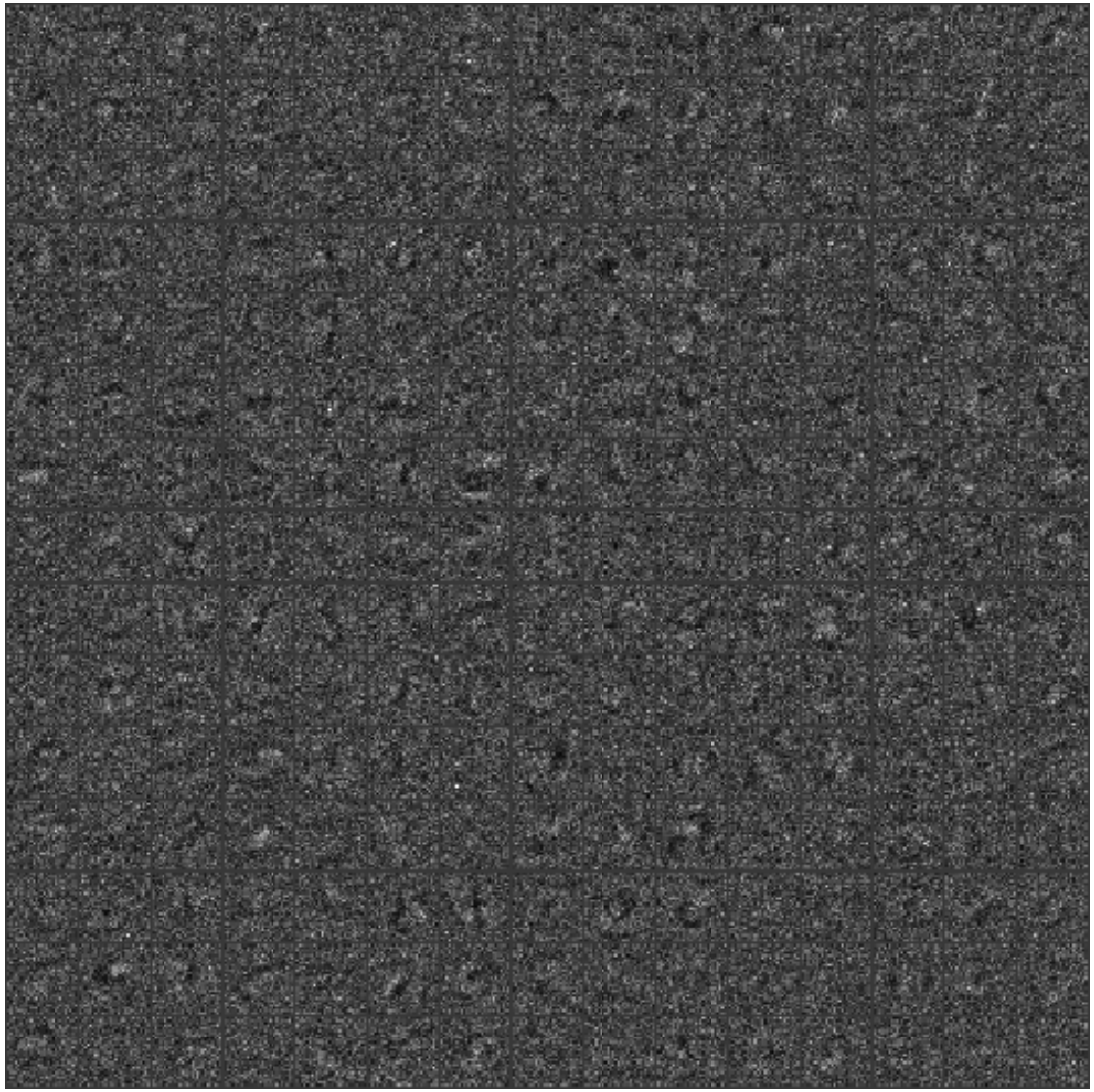
Important to anneal stepsize η_k towards the end, e.g.



Adaptation of η_k possible (Adam, Adagrad, Adadelta).

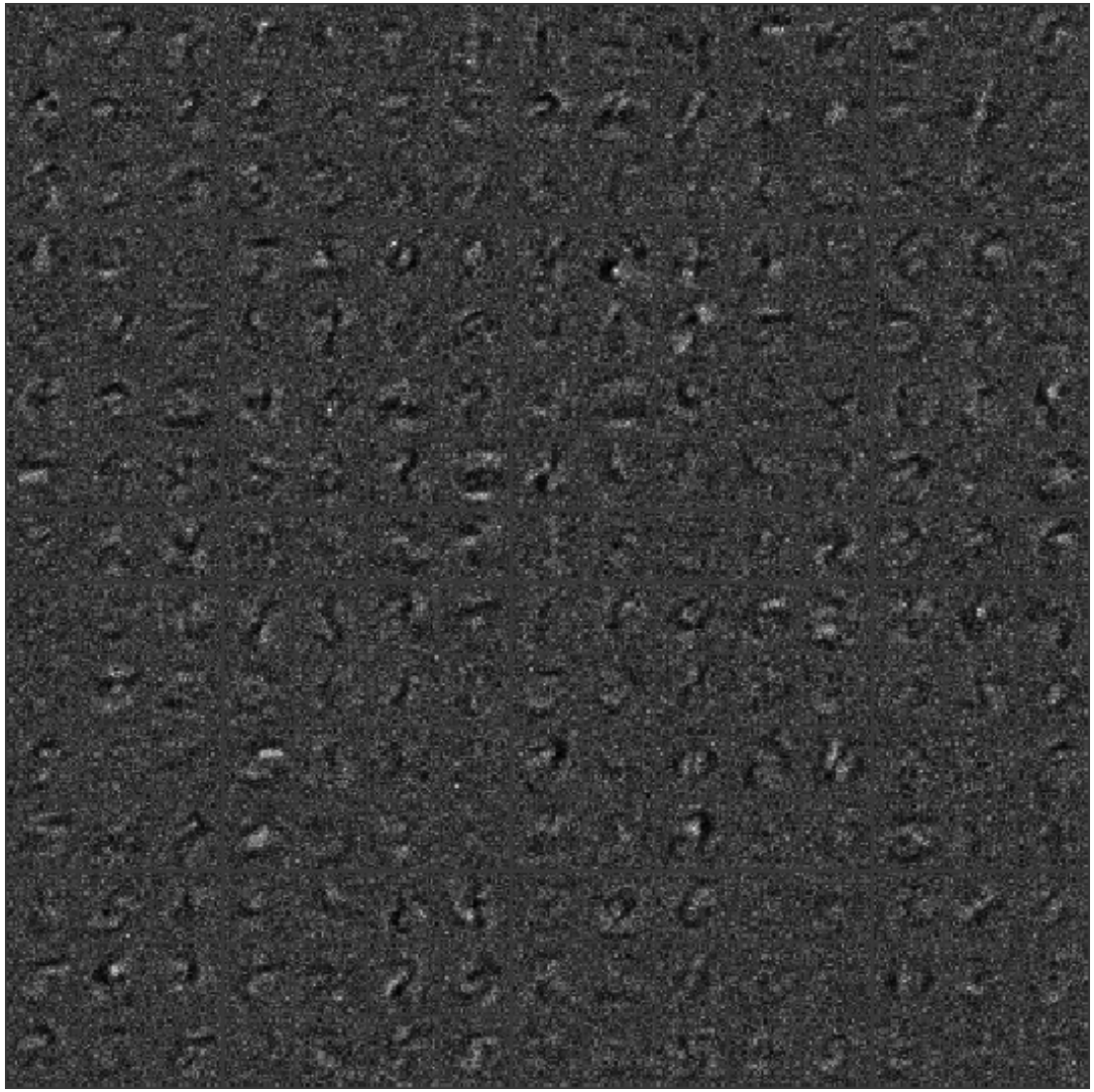
Aalto University

$\mathbf{W}^{(1)}$ after epoch 1



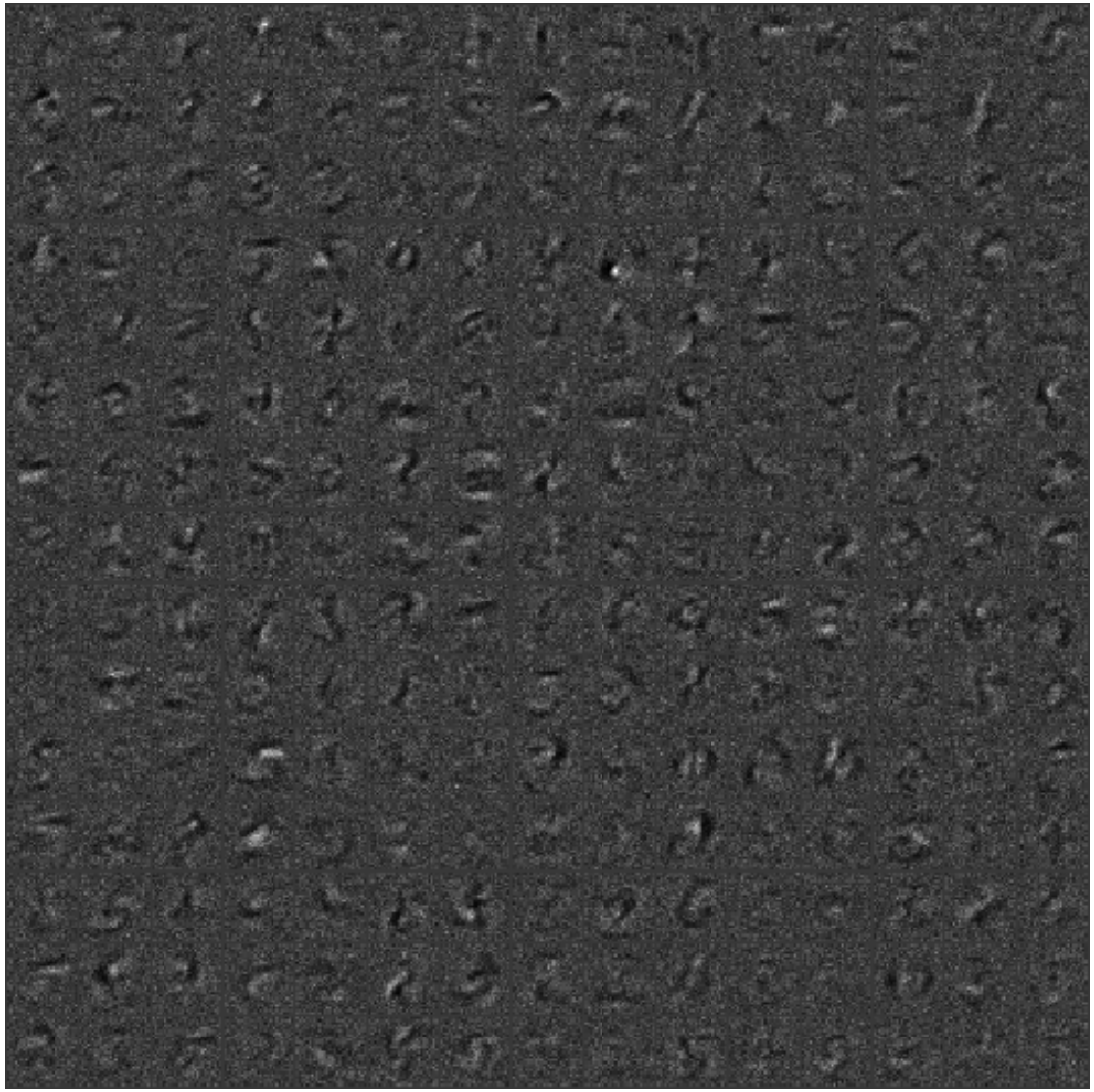
Aalto University

$\mathbf{W}^{(1)}$ after epoch 2



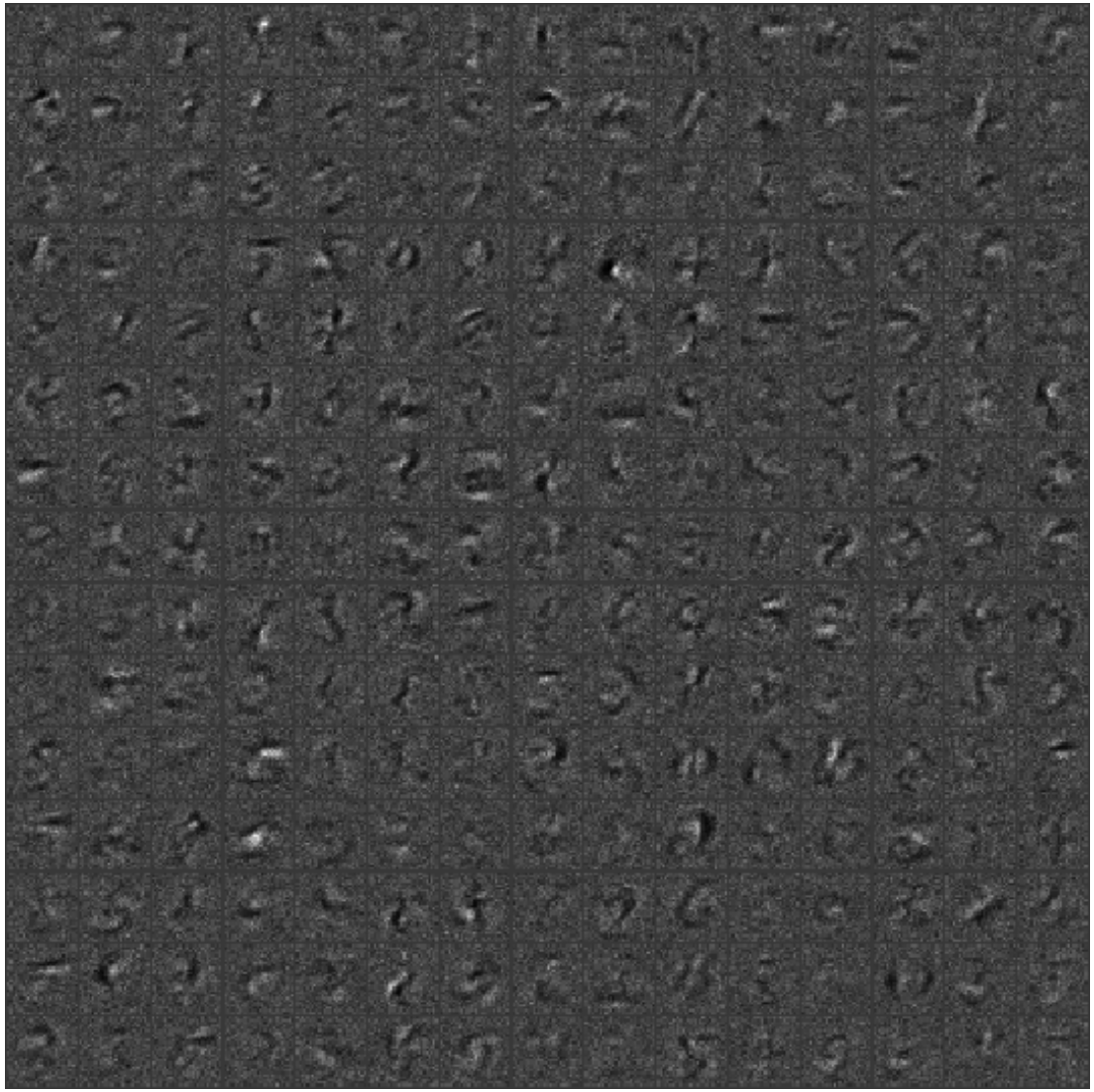
Aalto University

$\mathbf{W}^{(1)}$ after epoch 3



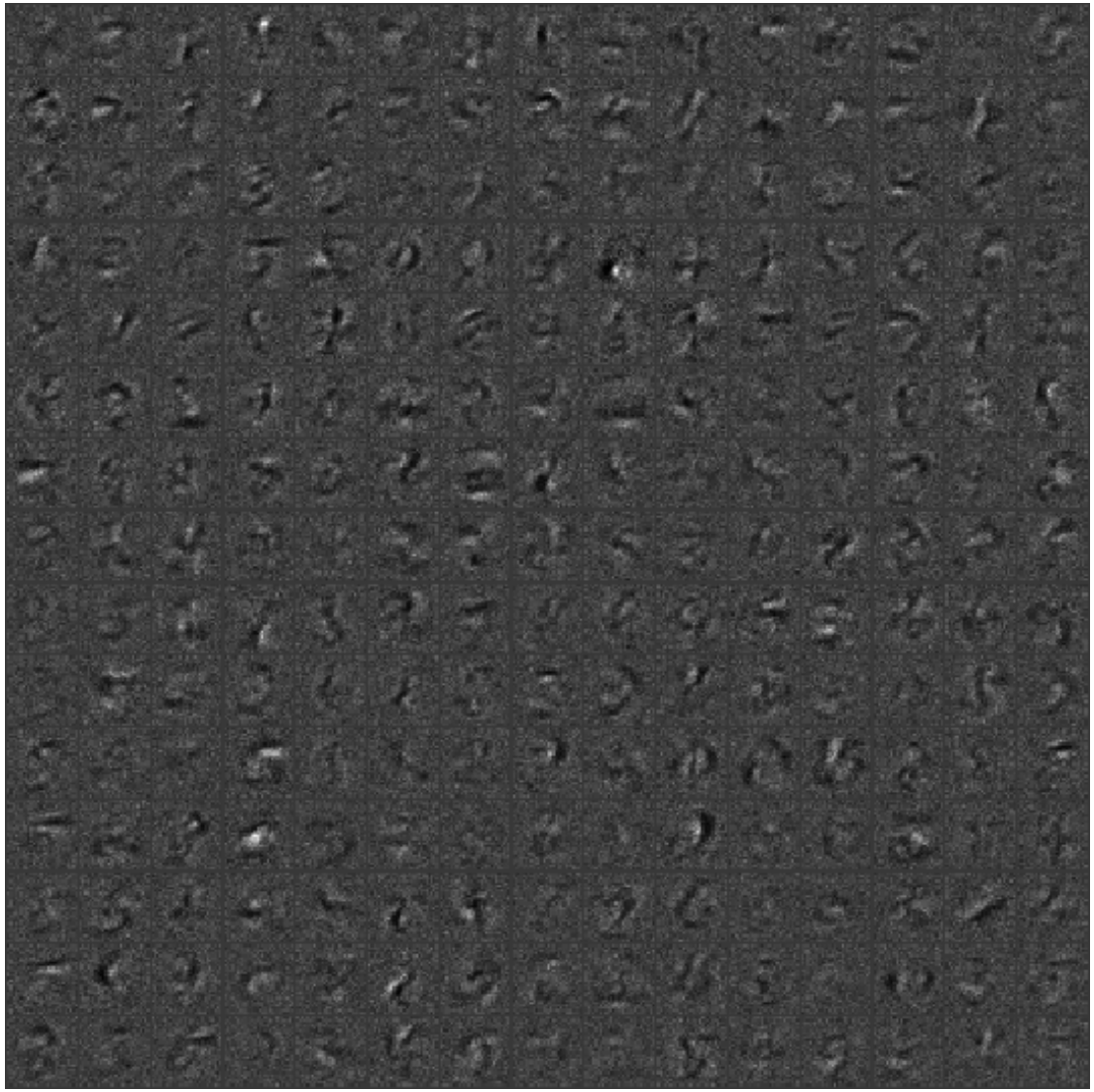
Aalto University

$\mathbf{W}^{(1)}$ after epoch 4



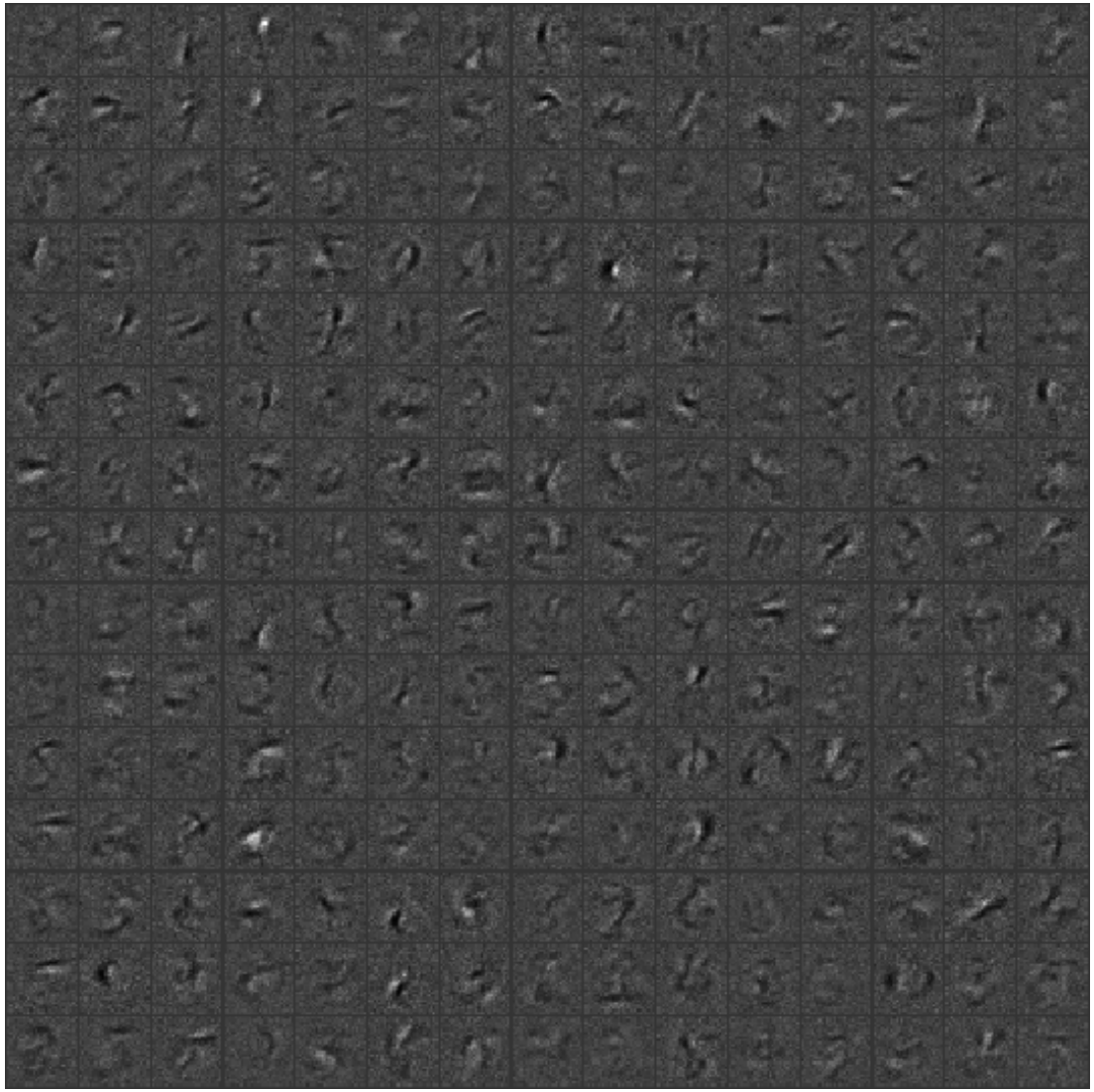
Aalto University

$\mathbf{W}^{(1)}$ after epoch 5



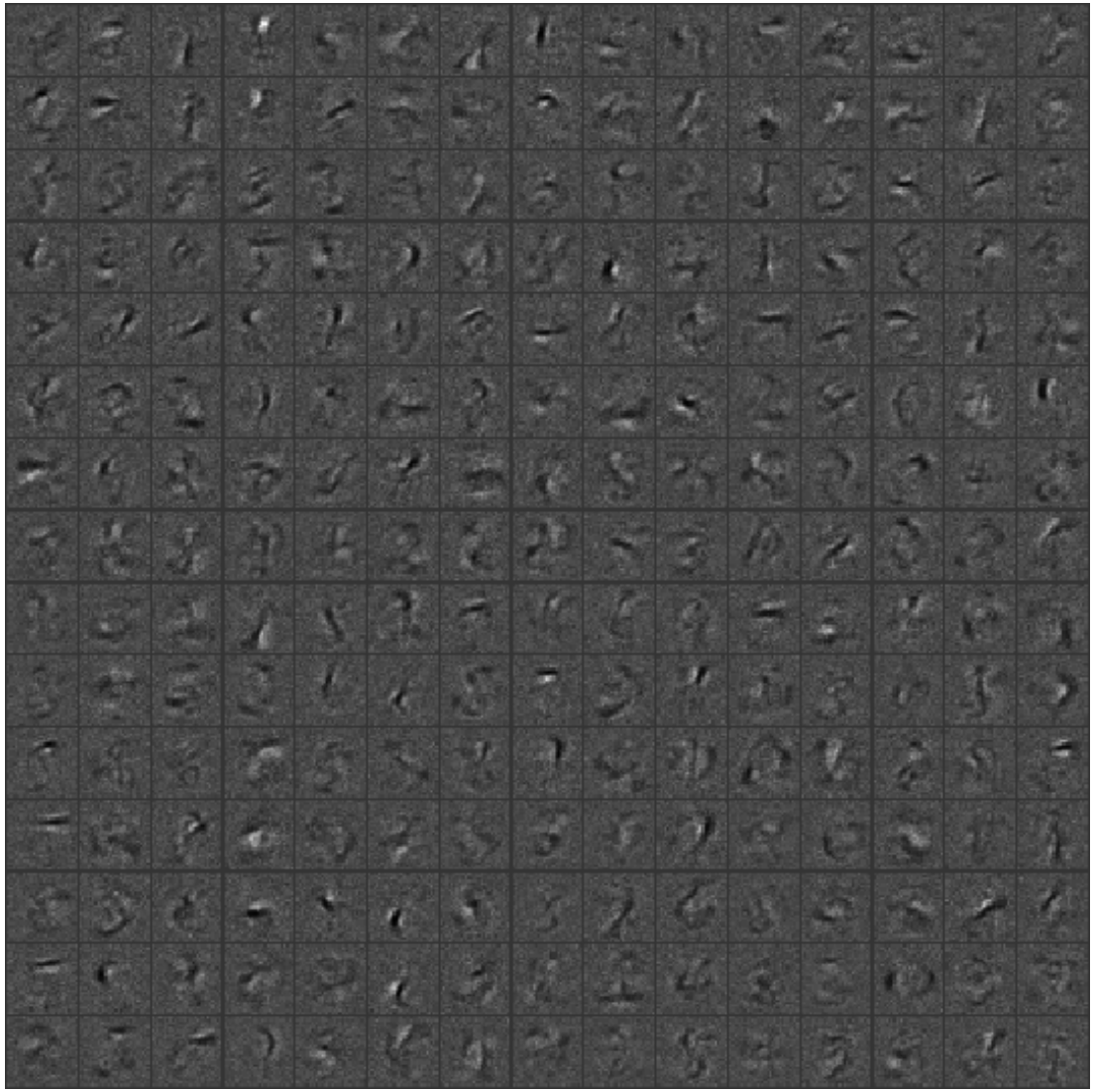
Aalto University

$\mathbf{W}^{(1)}$ after epoch 10



Aalto University

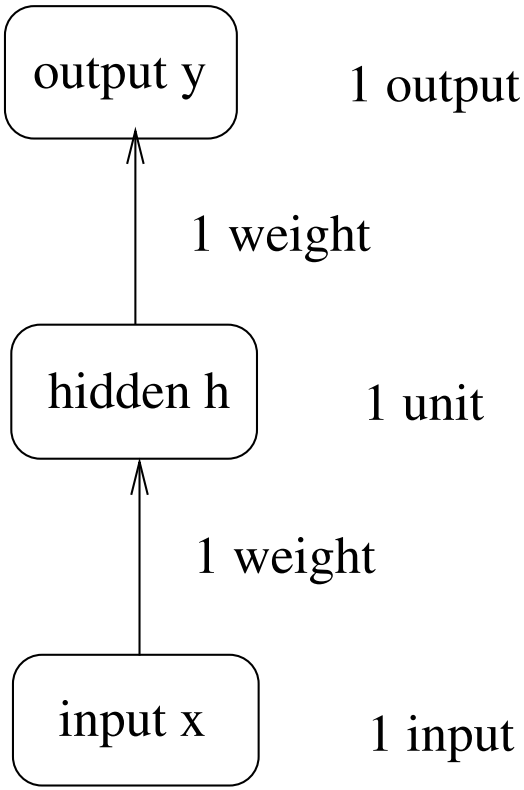
$\mathbf{W}^{(1)}$ after epoch 50 (final)



Aalto University

Backpropagation (Linnainmaa, 1970)

Computing gradients in a network.



First with scalars. Use chain rule:

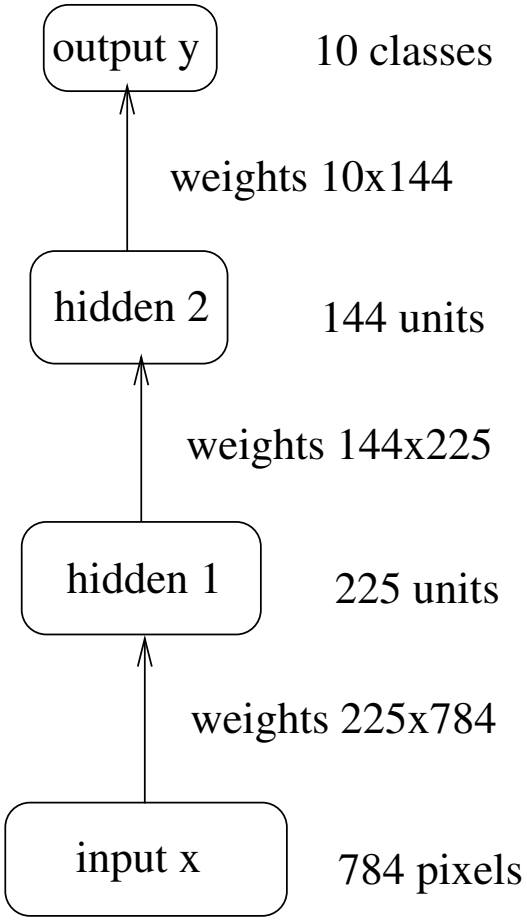
$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial w_2}$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial w_1}$$

Aalto University

Backpropagation

With vectors, consider multiple paths:



$$\frac{\partial C}{\partial W_{ij}^{(3)}} = \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial W_{ij}^{(3)}}$$

$$\frac{\partial C}{\partial W_{jk}^{(2)}} = \sum_i \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial h_j^{(2)}}\frac{\partial h_j^{(2)}}{\partial W_{jk}^{(2)}}$$

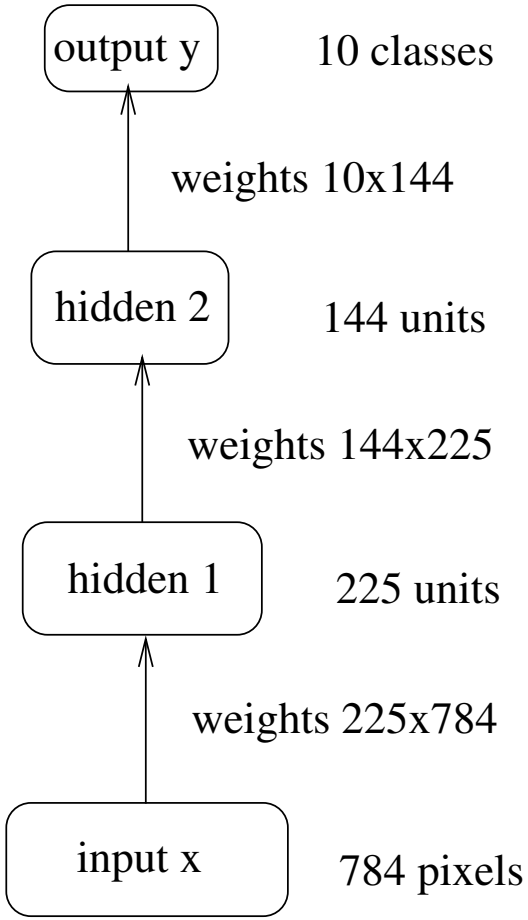
$$\frac{\partial C}{\partial W_{kl}^{(1)}} = \sum_{i,j} \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial h_j^{(2)}}\frac{\partial h_j^{(2)}}{\partial h_k^{(1)}}\frac{\partial h_k^{(1)}}{\partial W_{kl}^{(1)}}$$

How many paths as a function of depth?

Aalto University

Backpropagation

Dynamic programming avoids exponential complexity.
Store intermediate results



$$\frac{\partial C}{\partial h_j^{(2)}} = \sum_i \frac{\partial C}{\partial y_i} \frac{\partial y_i}{\partial h_j^{(2)}}$$

$$\frac{\partial C}{\partial h_k^{(1)}} = \sum_j \frac{\partial C}{\partial h_j^{(2)}} \frac{\partial h_j^{(2)}}{\partial h_k^{(1)}}$$

to get all layers L simply as

$$\frac{\partial C}{\partial W_{ij}^{(L)}} = \frac{\partial C}{\partial h_i^{(L)}} \frac{\partial h_i^{(L)}}{\partial W_{ij}^{(L)}}$$

Aalto University

Backpropagation, tiny example

$$y = w_2 h + \text{noise}$$
$$h = w_1 x$$
$$C = (y - 1.5)^2$$
$$\frac{\partial C}{\partial w_2} = 2(w_2 w_1 x - 1.5) w_1 x$$
$$\frac{\partial C}{\partial w_1} = 2(w_2 w_1 x - 1.5) w_2 x$$

Note a scaling issue: If w_1 is doubled and w_2 is halved,
- output y stays the same.
- system is twice as sensitive to changes in w_2.
- gradient of w_2 is doubled!

Aalto University

Table of Contents

Aalto University

Exponential growth/decay forward

Recall the model

$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$
$$\mathbf{h}^{(2)} = \text{relu}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$
$$\mathbf{h}^{(1)} = \text{relu}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

Ignoring softmax and biases, we can write

$$y_i = \sum_{j,k,l} \mathbf{1}\left(h_j^{(2)} > 0\right) \mathbf{1}\left(h_k^{(1)} > 0\right) W_{ij}^{(3)} W_{jk}^{(2)} W_{kl}^{(1)} x_l$$

Exponential growth/decay of forward signals!

Aalto University

Exponential growth/decay backward

Given indicators $\mathbf{1}(\cdot)$, model is linear

$$y_i = \sum_{j,k,l} \mathbf{1}\left(h_j^{(2)} > 0\right) \mathbf{1}\left(h_k^{(1)} > 0\right) W_{ij}^{(3)} W_{jk}^{(2)} W_{kl}^{(1)} x_l$$

$$\frac{\partial y_i}{\partial x_l} = \sum_{j,k} \mathbf{1}\left(h_j^{(2)} > 0\right) \mathbf{1}\left(h_k^{(1)} > 0\right) W_{ij}^{(3)} W_{jk}^{(2)} W_{kl}^{(1)}$$

Exponential growth/decay of gradient, too!

$\Rightarrow$ Scale of initialization important.

Aalto University

Initialization

Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$
where size of $\mathbf{W}$ is $n_i \times n_j$.

Outline of derivation:
Assume independent signals.
Retain variance forward: $\sqrt{\frac{2}{n_j}}$

Retain variance backward: $\sqrt{\frac{2}{n_i}}$

(2 is from relu indicators being on half the time.)
Strike a balance between the two (Glorot and Bengio, 2010).

(Other ideas, relevant for RNNs: sparse initialization (Sutskever et al. 2013), orthogonal initialization (Saxe et al., 2014))

Aalto University

Batch normalization (Ioffe and Szegedy, 2015)

Recent idea: Normalize hidden units to zero-mean and unit variance over mini-batch.

- Allows much higher learning rates.
  (gradient closer to Newton's method (Raiko et al., 2012))
- Less careful about initialization. (solves scaling issues)
- Acts as a regularizer.

Aalto University

Table of Contents

Aalto University

Expressivity

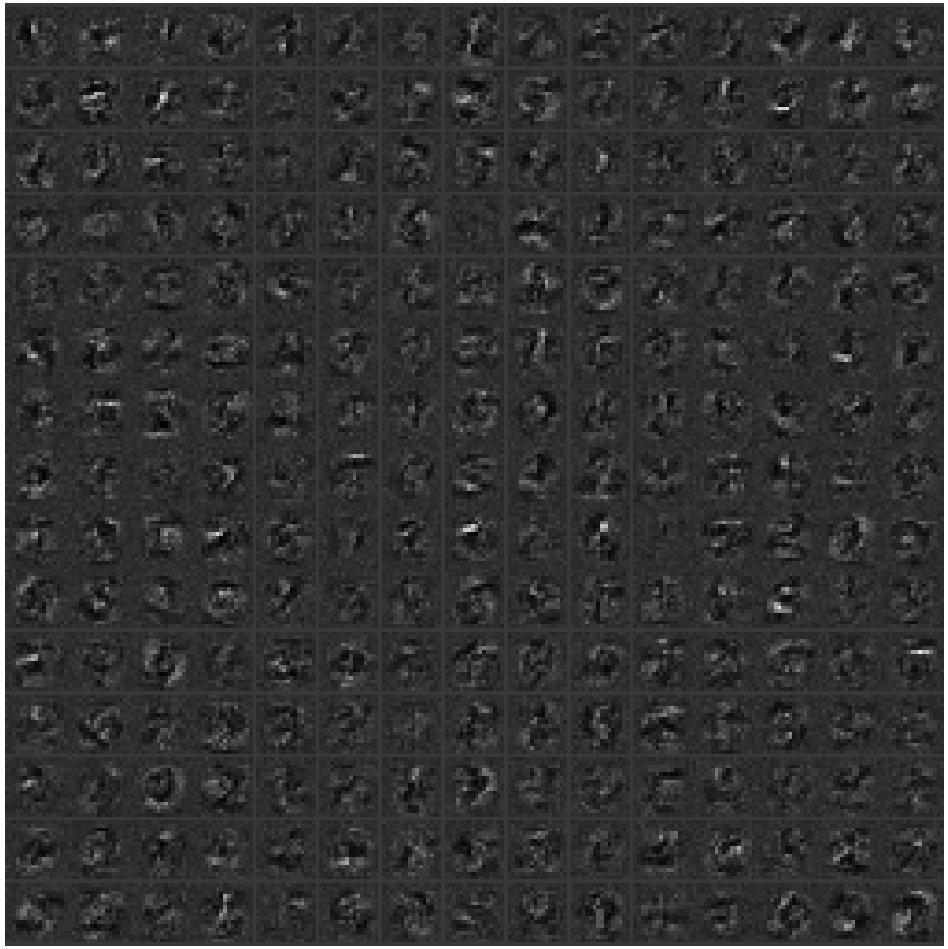
Neural networks are very powerful (universal appr.).
Easy to perform great on the training set (overfitting).
Regularization improves generalization to new data.
Use held-out validation data to choose hyperparameters (e.g. regularization strength).
Use held-out test data to evaluate performance.

Aalto University

Power demo: Training first layer only

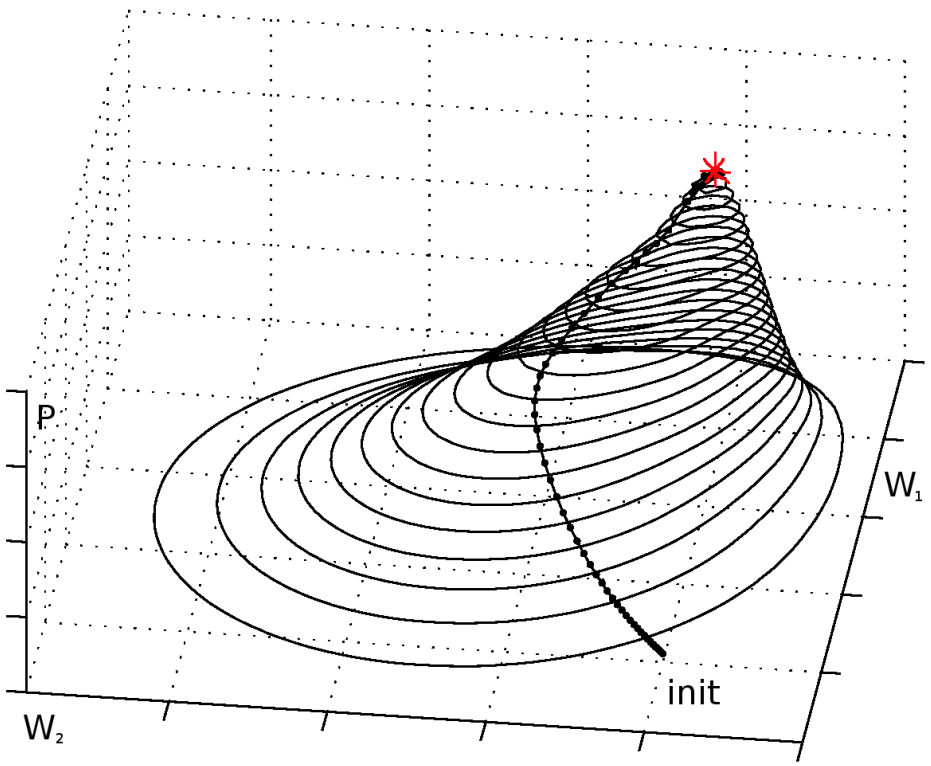
No regularization, training $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ only.

0.2% error on training set, 2% error on test set.



Aalto University

What is overfitting?



Probability theory states how we should make predictions (of $\mathbf{y}_{\text{test}}$) using a model with unknowns $\boldsymbol{\theta}$ and data $\mathbf{X} = \{\mathbf{x}_{\text{train}}, \mathbf{y}_{\text{train}}, \mathbf{x}_{\text{test}}\}$:

$$P(\mathbf{y}_{\text{test}} \mid \mathbf{X}) = \int P(\mathbf{y}_{\text{test}}, \boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}$$
$$= \int P(\mathbf{y}_{\text{test}} \mid \boldsymbol{\theta}, \mathbf{X}) P(\boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}.$$

Probability of observing $\mathbf{y}_{\text{test}}$ can be acquired by summing or integrating over all different explanations $\boldsymbol{\theta}$. The term $P(\mathbf{y}_{\text{test}}|\boldsymbol{\theta}, \mathbf{X})$ is the probability of $\mathbf{y}_{\text{test}}$ given a particular explanation $\boldsymbol{\theta}$ and it is weighted with the probability of the explanation $P(\boldsymbol{\theta}|\mathbf{X})$. However, such computation is intractable. If we want to choose a single $\boldsymbol{\theta}$ to represent all the probability mass, it is better not to overfit to the highest probability peak, but to find a good representative of the mass.

Posterior probability mass matters

Center of gravity $\neq$ maximum

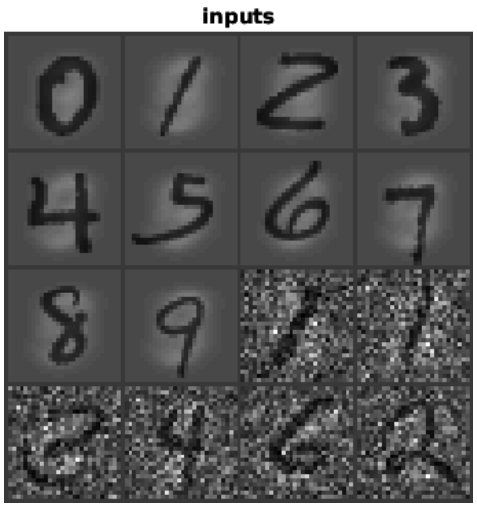
Aalto University

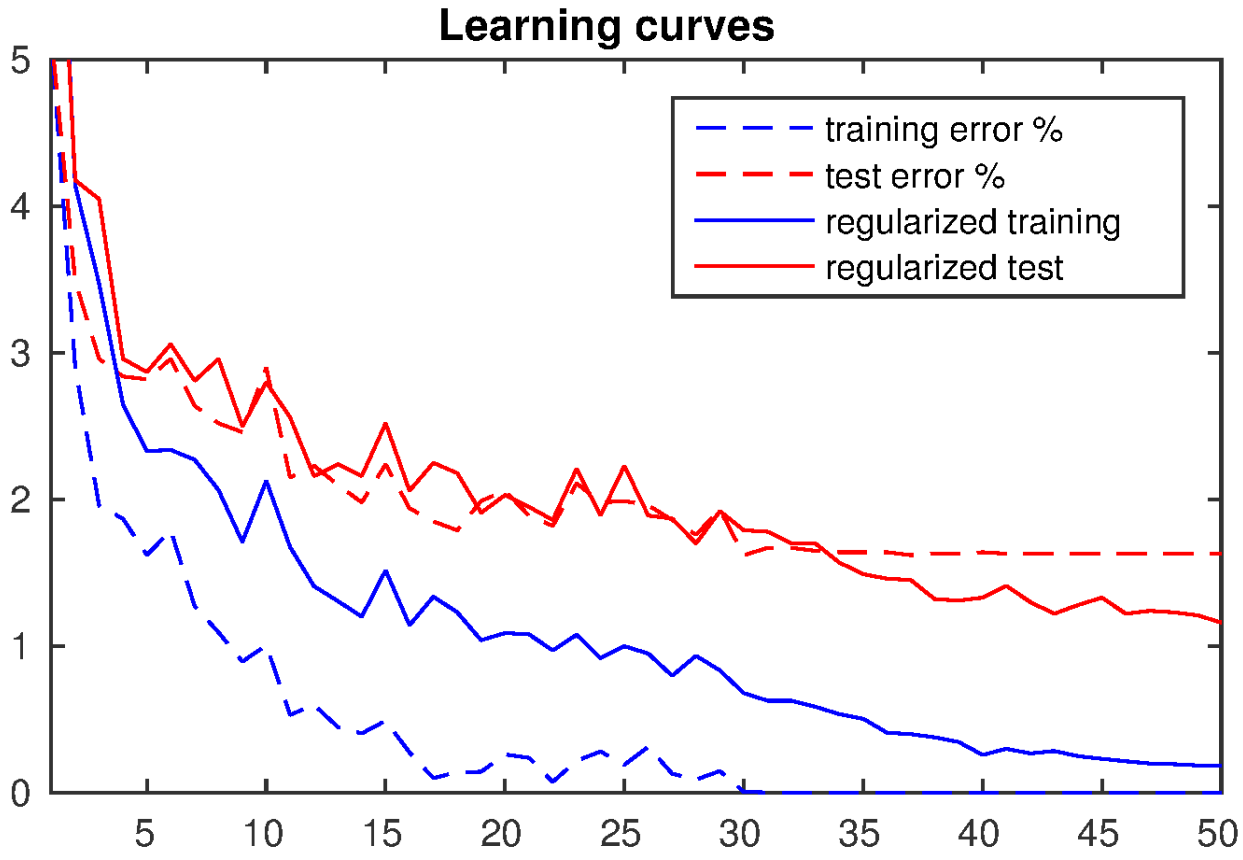
Regularization methods

- Weight decay (Tikhonov, 1943): $C = \cdots + \lambda \|\mathbf{W}\|^2$
- Inject noise to inputs (Sietsma and Dow, 1991)
- Limited size of network
- Early stopping
- Sparsity (either signals $\mathbf{h}$ or weights $\mathbf{W}$)
- Weight sharing (e.g. convolutional)
- Auxiliary tasks (e.g. unsupervised)
- Ensembles, variational methods
- ...

Aalto University

Effect of noise regularization





Without regularization training error goes to zero and learning stops. With noise, test error keeps dropping.

Aalto University

Thanks for listening!

Thanks to Huiling Wang for the optimization example.

Aalto University

Possible exercises (1/2)

- Follow the Theano tutorial on neural networks: `http://deeplearning.net/tutorial/mlp.html`
- Load the MNIST example Matlab code `iki.fi/raiko/summerschool` and change something (network size, regularization, initialization, learning rate schedule ...). Inspect the results.
- Load the optimization example Matlab code and solve the Newton's method pointing the wrong way based on `arxiv.org/pdf/1405.4604v2.pdf`.

Aalto University

Possible exercises (1/2)

- Work out the math for the backpropagation algorithm in the MNIST classification example.
- Take a trained network and arrange the hidden units for a better visualization: Make outgoing weight vectors of neighboring units close to each other. (Hint: Try using simulated annealing or the self-organizing map.)

Aalto University