Deep Learning Basics

(and a bit of unsupervised learning)

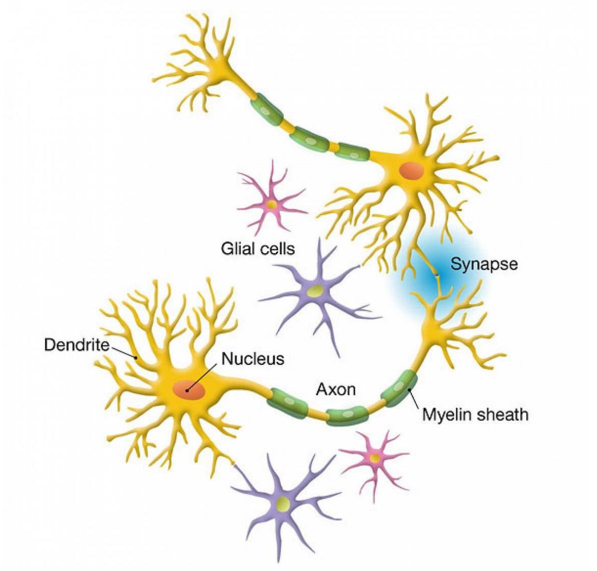
Wikipedia says...

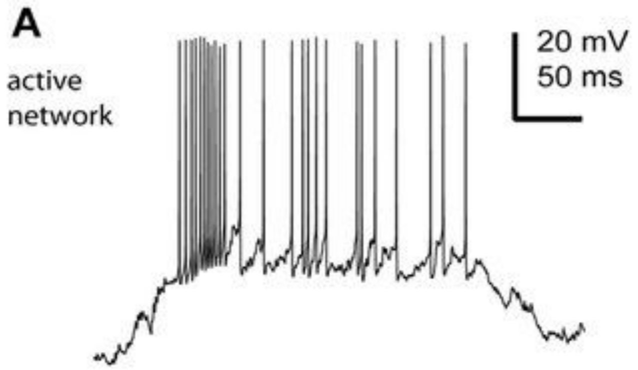
Deep learning is part of a broader family of machine learning methods, which is based on **artificial neural networks** with **representation learning**. Learning can be supervised, semi-supervised or unsupervised

Artificial neural networks (ANNs)

Meet the neuron

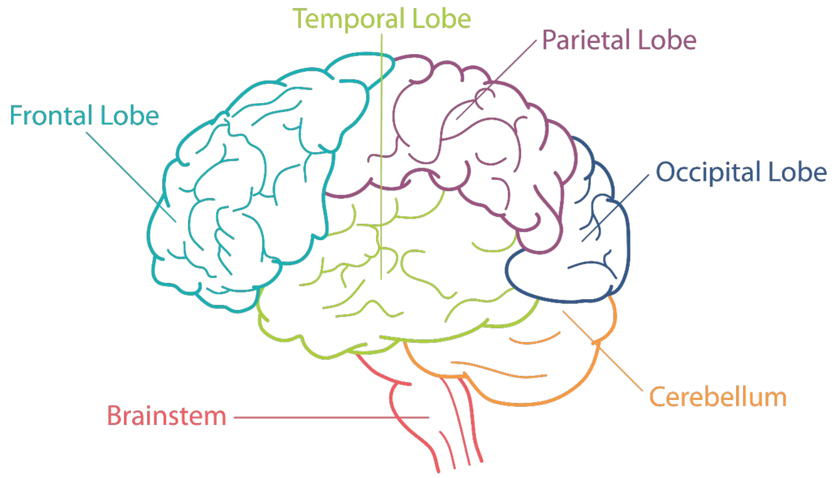
- Structure and function
  - Dendrites (input)
  - Body (sum)
  - Axon (output)
- Information
  - Threshold → spike
  - Synapses/synaptic weights





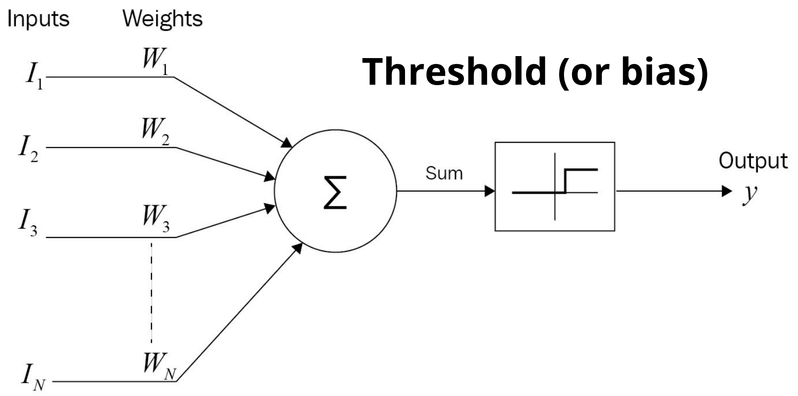
Brain is pretty remarkable

- ~10^{11} neurons
- ~10^{14} synapses
- Many simple interacting units form the basis of intelligence?
- A single “learning algorithm”?
  - Visual cortex recruited for auditory processing in blind people

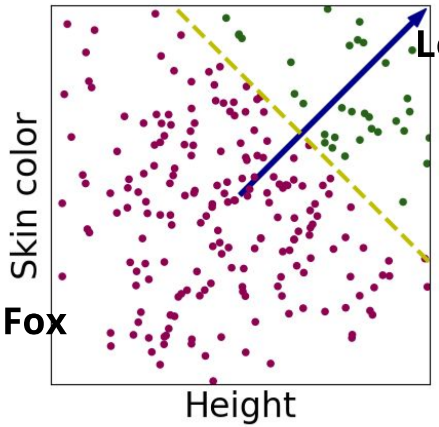


Artificial neuron

- ML from neuroscience: reverse-engineering the brain
  - Computational principles and functionality
- (but also neuroscience from ML: AI models shed light on basic questions)
- The simplest model
  - McCullochs and Pitts, 1943; Rosenblatt, 1958; Widrow, 1962
  - At the time, much into OR/AND/XOR functions
- Just a linear discriminator?
  - This was the first major dip in the popularity of neural networks (1970-1980 “winter”)



Threshold (or bias)

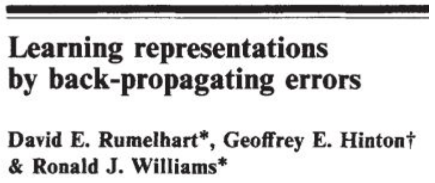


LeBron James

Michael J. Fox

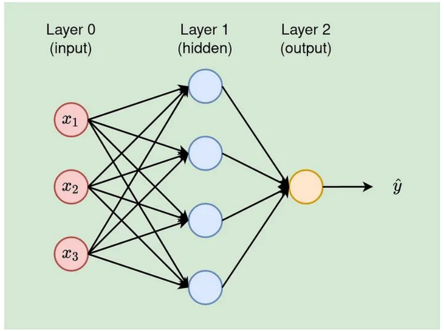
Artificial network

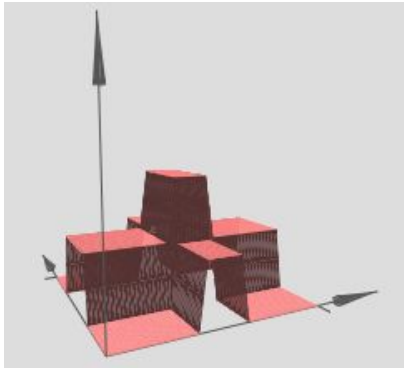
- 1980s: the second wave
  - Connectionism: intelligence = interacting network + distributed representations
- Yet, the magic happens when we stack (at least) two layers of neurons
  - Multilayer perceptron
  - “Universal approximation theorem”
- And we invent “backpropagation” (1986)



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*





it’s just a function!

$$y = f(x|\theta) \equiv \mathrm{net}(x|W)$$

Backpropagation (aka: backward propagation of errors)

Backpropagation is just: computing the gradient!!

- It takes a special form that basically makes the network work in reverse
  - From the output layer all the way back to the input layer
  - And what are we passing back? Basically, the output error
  - So: backward + propagation + errors
- But the goal is to just compute

$$\frac{\partial \mathrm{net}(x|W)}{\partial w_{ij}^{l}}$$

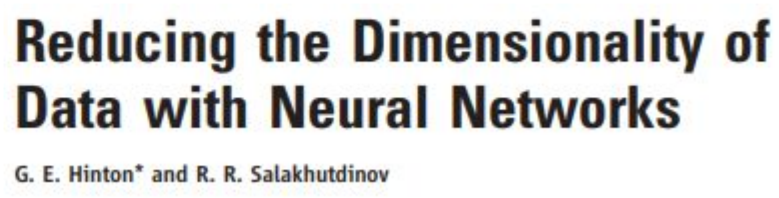
- (because, remember?, gradient is what we need for optimizing the parameters of our function)

Too deep (two layers are just for the theory)

“The second wave of neural networks research lasted until the mid-1990s. Ventures based on neural networks and other AI technologies began to make **unrealistically ambitious claims** while seeking investments. When AI research did not fulfillthese unreasonable expectations, investors were disappointed”

“We now know that algorithms that have existed since the 1980s work quite well, but this was not apparent circa 2006. The issue is perhaps simply that these **algorithms were too computationally costly** to allow much experimentation with the hardware available at the time”

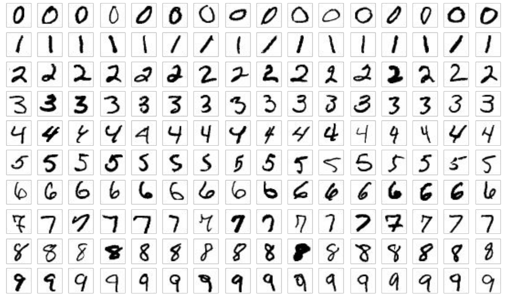
The third wave (2006-today)


Reducing the Dimensionality of Data with Neural Networks

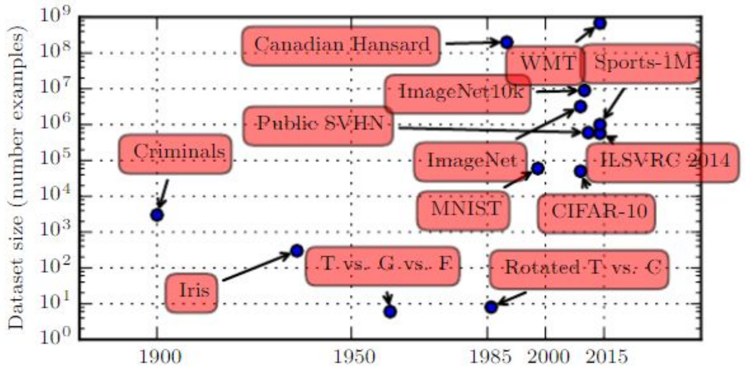
G. E. Hinton* and R. R. Salakhutdinov

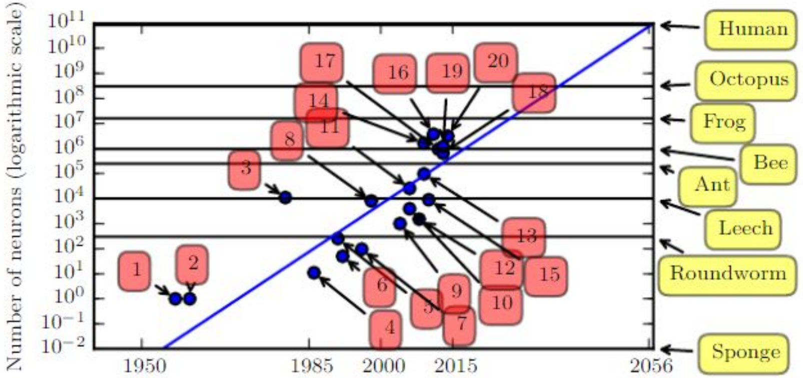
- “Deep learning” enters the scene
- Initial focus on new unsupervised learning techniques and the ability of deep models to generalize well from small datasets
- But then we discovered very large datasets...
- ... and the insatiability of deep networks
- There seems not to be a limit on how deep we can go...

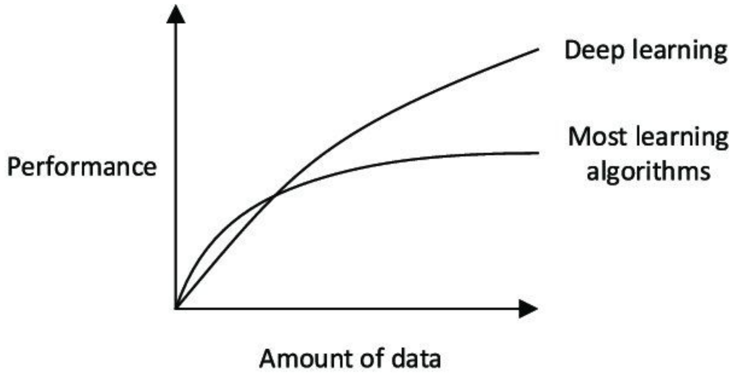
MNIST - "the drosophila of machine learning"



“Larger is better”



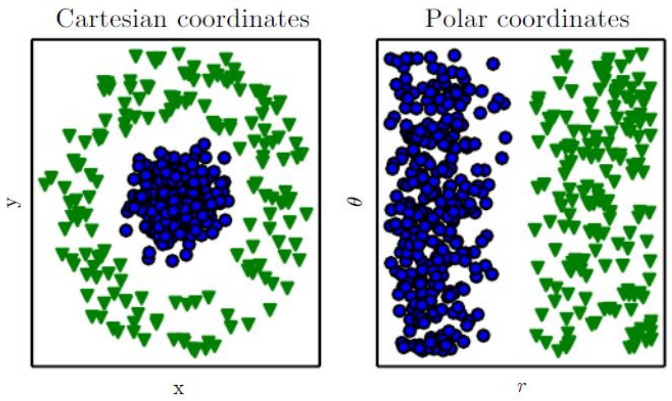


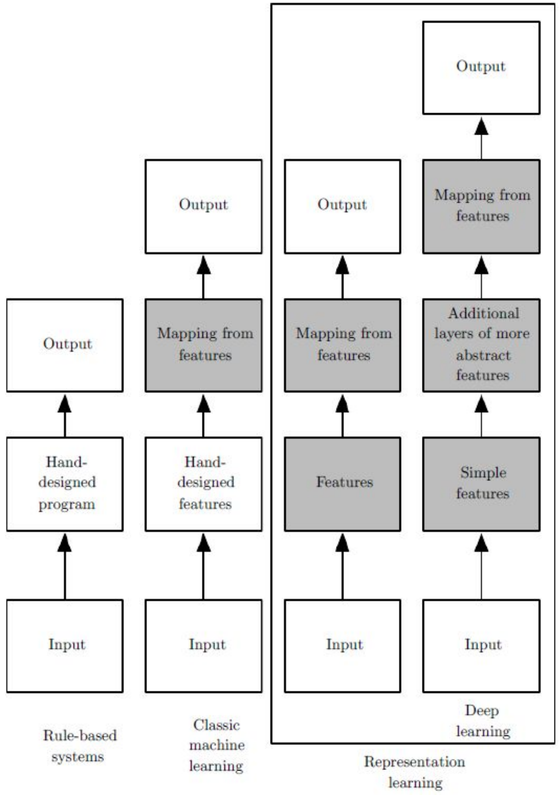


Representations

Representation learning

- It’s really a fancy term… for a property that seems to tell Deep Learning apart from other algorithms…
- My definition: The ability to (possibly non-linearly) **map your features** to a new feature space, where your **categories are close-to linearly separable**





Deep networks at their best

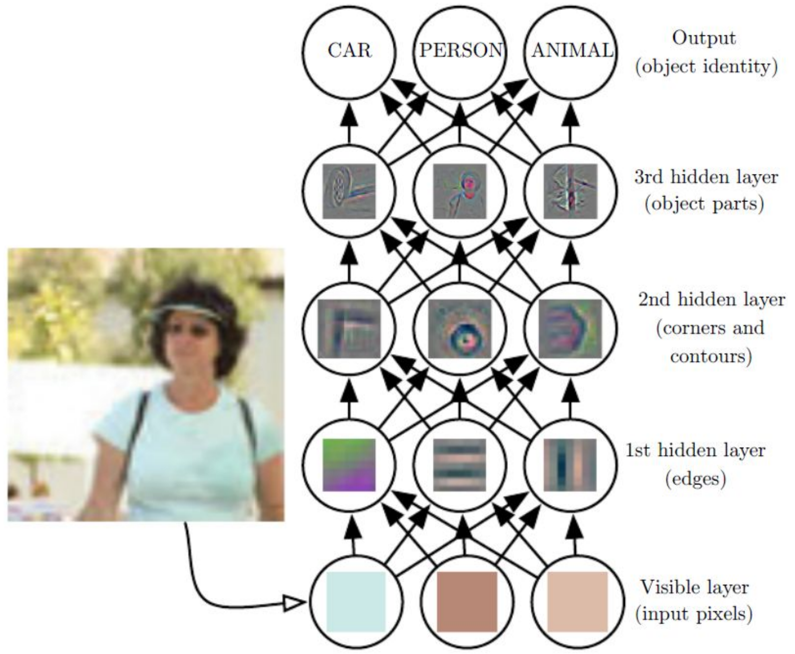
Tasks that are easy for people to perform but hard for people to describe formally—problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images, that have very large input spaces (pixels, voxels, waves), with lots of “noise”

Why?

A hierarchy of hidden factors

- Each individual pixel carries very little information about the real content of an image
- There are hidden factors that influence simultaneously large parts of the data we observe (“red car”, “there is a tree”)
- Such hidden factors are often related to our goals
    - Knowing that there is a car, it is red, it is sporty, it is still can help in almost all the things you want to accomplish about the picture
- Intuitively, such hidden factors often live in a hierarchy (a tree is something that has a trunk and branches; a branch entails smaller branches and leaves…)
- But they are very hard to encode
    - Want to detect cars? Look for wheels! Ok, a wheel is round. No, if you look at it from an angle, it seems more like an oval. What color should we look for? Etc.

Hierarchical representations


CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

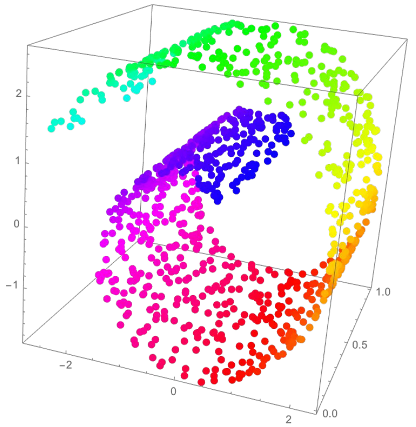
Deep learning solves this central problem by introducing representations that are expressed in terms of other, simpler representations

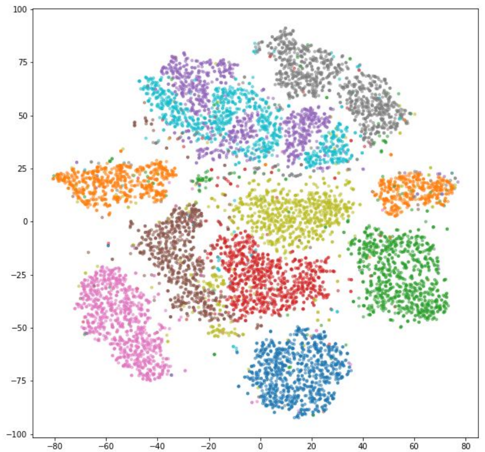
No free-lunch theorem

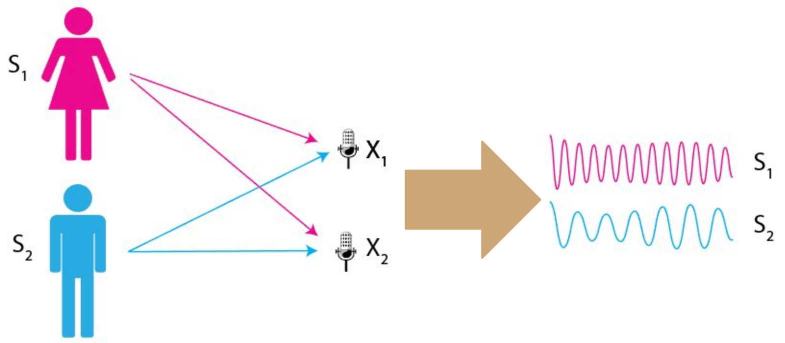
Unsupervised learning - the basics

Looking for simpler representations of our data

1. Dimensionality reduction ("compress")
   a. Your data has N dimensions; find a compressed representation in K (K << N) dimensions
   b. Finding a "manifold" that the data lies near
2. Clustering ("sparsify")
3. Independent representations ("separate")

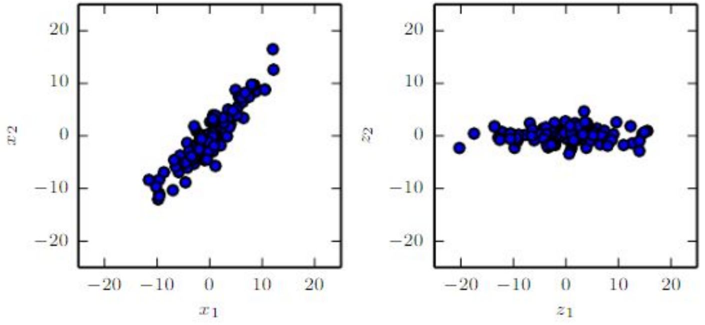


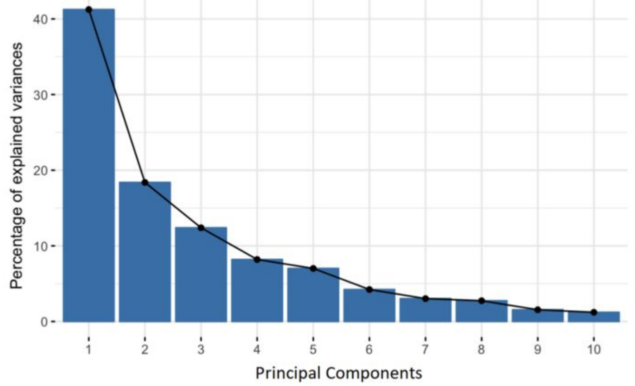




Principal component analysis (PCA)

- IDEA 💡: rotate your axes so that, in the new reference system, your data become uncorrelated
  - Example: height and weight → size and “fat”
  - How? (Whispering): find the eigenvalues and eigenvectors of the covariance matrix
- This is (simple) *separation*
- *Compression*: keep only the k axes with largest variance

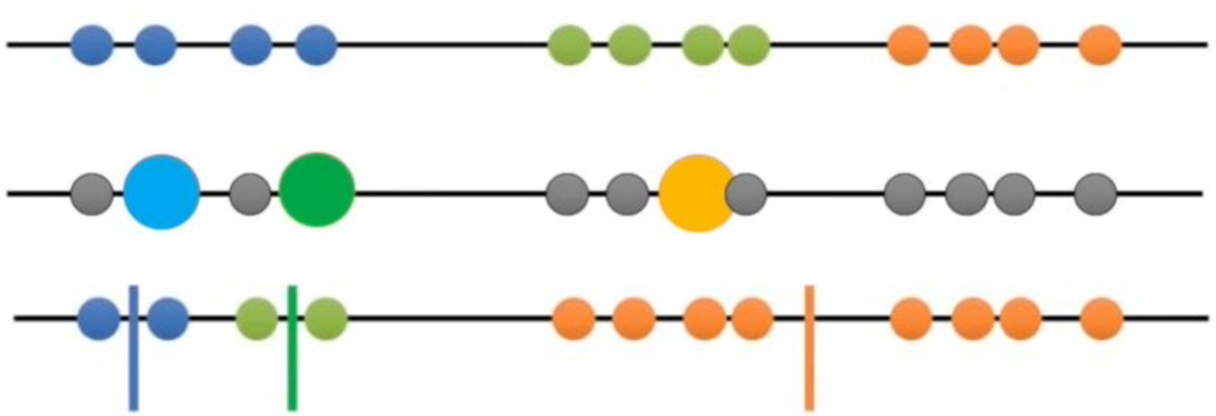


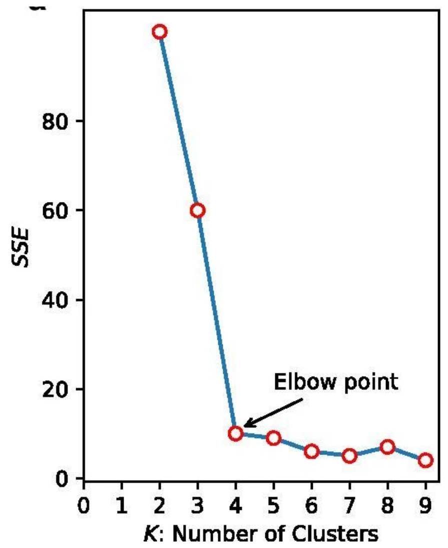


K-means clustering

https://t.ly/l9CF

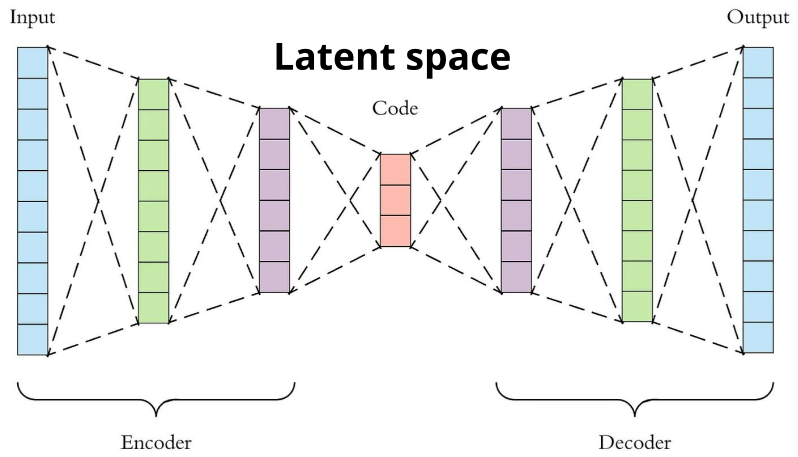
- IDEA 💡: choose k random “centroids”, make them “gather” nearest data points, re-compute the centroids, repeat till convergence
- Different initial conditions → different clusters (try several times)
- What’s the best clustering? (“average dispersion”)
- How to choose K (*a posteriori*): elbow point (also for PCA!)

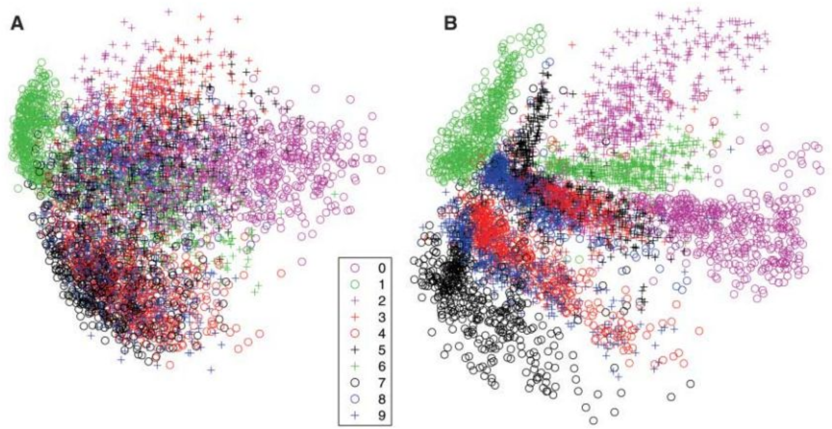




Autoencoders

- An autoencoder is a neural network that is trained to attempt to copy its input to its output…
- … passing through a tight “information bottleneck”
- Three parts: encoder, “latent space” (or code), decoder
- “Non-linear” PCA





Special ingredients

Machine learning recap

- Massaging the data
- Choosing your model
- Choosing your objective
- Training your model
- Testing your model

Massaging your data

- We have the best representation learning in town, let’s just put the data in
- Arguably this is false (false, not FALSE!)
- It’s the same
  - A deep network can perform (approximate) cartesian-to-polar change… but it’s hard…
  - Normalizing helps (and ~Gaussian inputs are more friendly)
- It’s almost the same
  - “Deep massages”: batch normalization, layer normalization, position encoding…
  - Strategies for weight initialization (hidden “good” statistics of weights)
- And it’s different
  - Data augmentation (very deep massage)

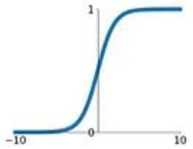
Choosing your model

- I’ll have an ANN!
- Yet there are so many details that can make the difference…
- Many of the achievements in Deep Learning have come from “shaping” the network in the right way

Activation function

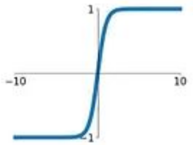
Sigmoid

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



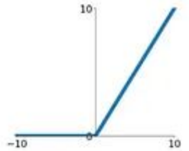
tanh

$\tanh(x)$



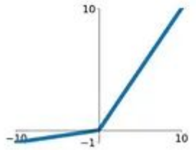
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

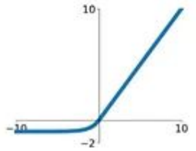


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



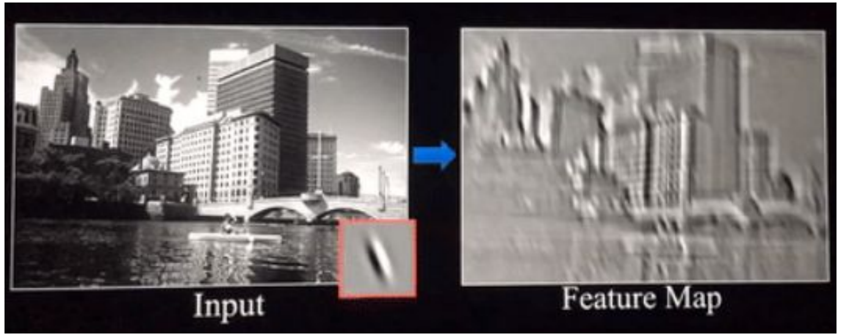
For output layer: linear (regression), sigmoid (two-class classification, multi-label classification), soft-max (general classification)

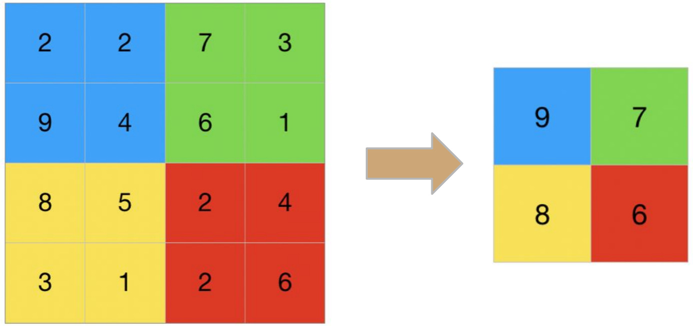
$$\mathrm{softmax}(x_1 | x_2,\ x_3) = \frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}}$$

$$\mathrm{softmax}(x_1 | x_2,\ x_3) + \mathrm{softmax}(x_2 | x_1,\ x_3) + \mathrm{softmax}(x_3 | x_1,\ x_2) \equiv 1$$

Convolutional Neural Networks (CNN)

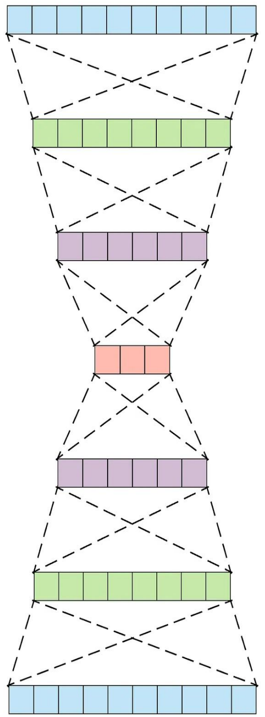
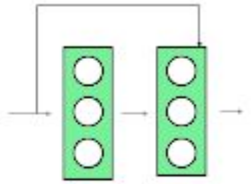
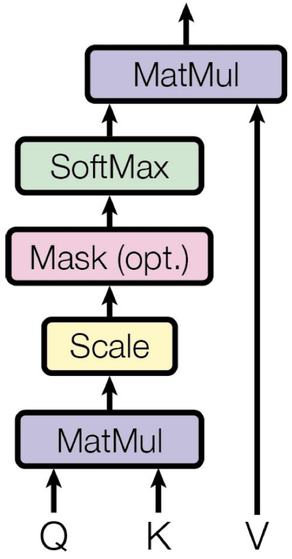
- Weight sharing to leverage invariance in the data
  - A vertical edge (“raw” feature) is useful wherever it occurs
- Pooling makes detected “more complex” features less “local”
- Better generalization, less parameters
- Not just images (speech, for example)





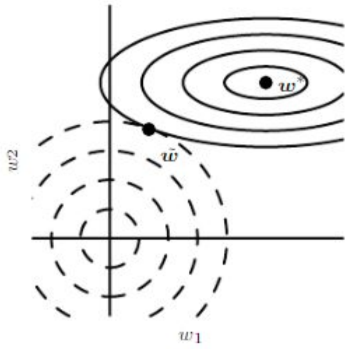
Other structural variants

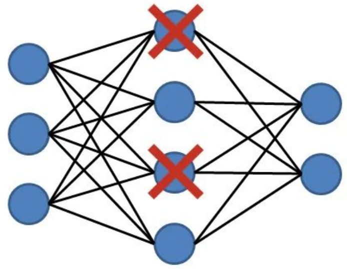
- "Information funnels"
- Skip connections
- Transformers
- ...

Improving generalization (regularization and friends)

- The “tightening”
  - L1 and L2
  - Early stopping (sort of L2 in disguise)
  - Noise: input (~L2), output (uncertain labels), weights (robustness)
  - Sparse representations
- The structural
  - Parameters sharing
  - Distillation
- The initialization
  - Smart strategies (“sparse initialization”)
  - Pre-training
  - Transfer learning (“using information, strategies, and skills in new situation or context”)
- The “deep”
  - Dropout
  - Semi-supervised learning and multi-task learning
  - Adversarial examples and learning





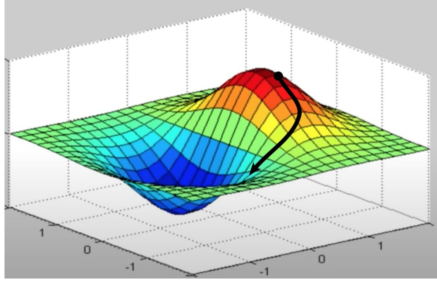
Choose your objective

- For basic applications, **nothing new here**
- In more advanced applications (variational autoencoder (VAE), generative adversarial networks (GAN)) “new” types of objective functions… (but it’s for the next course :-)
- “Surrogate objectives” (just an aside)
  - You can’t optimize what you really want; you choose something related…
  - … and it can turn out to be even better (accuracy vs log-likelihood)

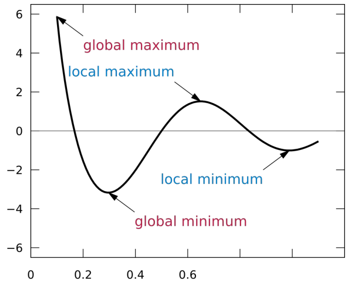
$$\text{cross\_entropy} = -\sum_i t_i \log(p_i) = -\log(p_{\text{cat}})$$

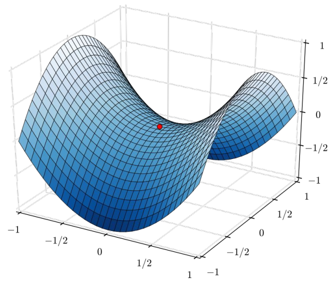
p_{cat} high (~1): small error
p_{cat} low (~0): large error

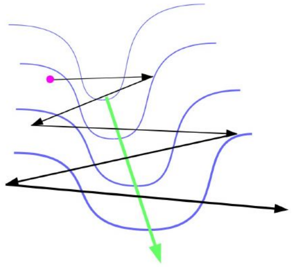
Training your model

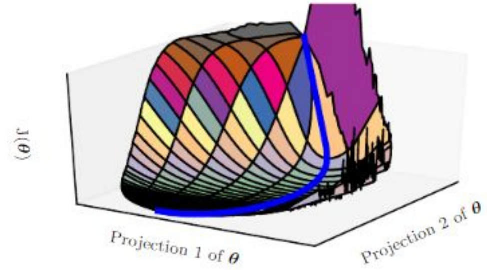


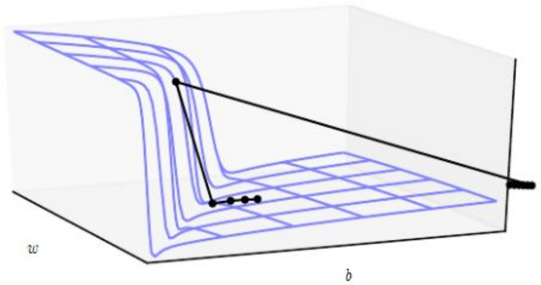
- We want to go down along a (multi-dimensional) surface…
- … basically, just sensing where the slope points toward
- **Challenges**: “Shallow minima”, saddle points, narrow valleys, plateaus, and “walls”





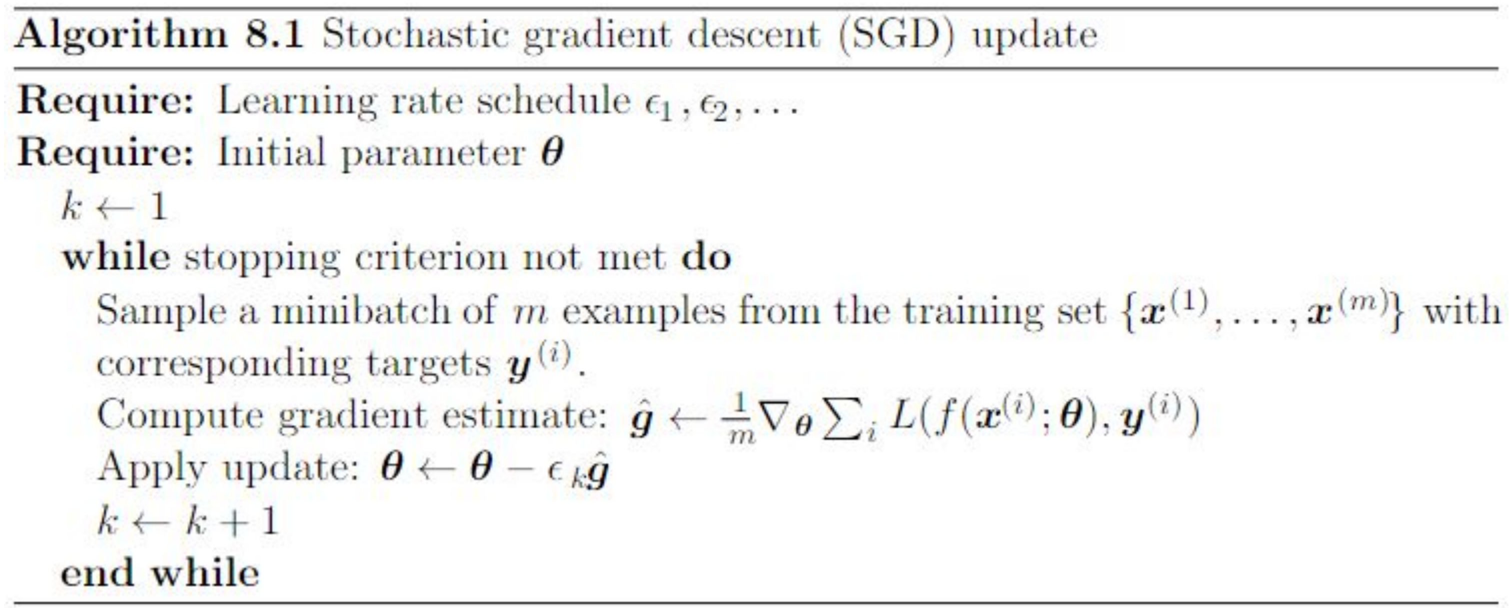






Stochastic gradient descent (SGD)

- Batch training
  - Second-order (curvature) methods: conjugate gradients, BFGS, … - we won’t cover them
- Stochastic gradient descent and **mini-batches**



Algorithm 8.1 Stochastic gradient descent (SGD) update

```
Require: Learning rate schedule $\epsilon_1, \epsilon_2, \dots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while
```

$$\sum_{k=1}^{\infty} \epsilon_k = \infty,$$

$$\sum_{k=1}^{\infty} \epsilon_k^2 < \infty.$$

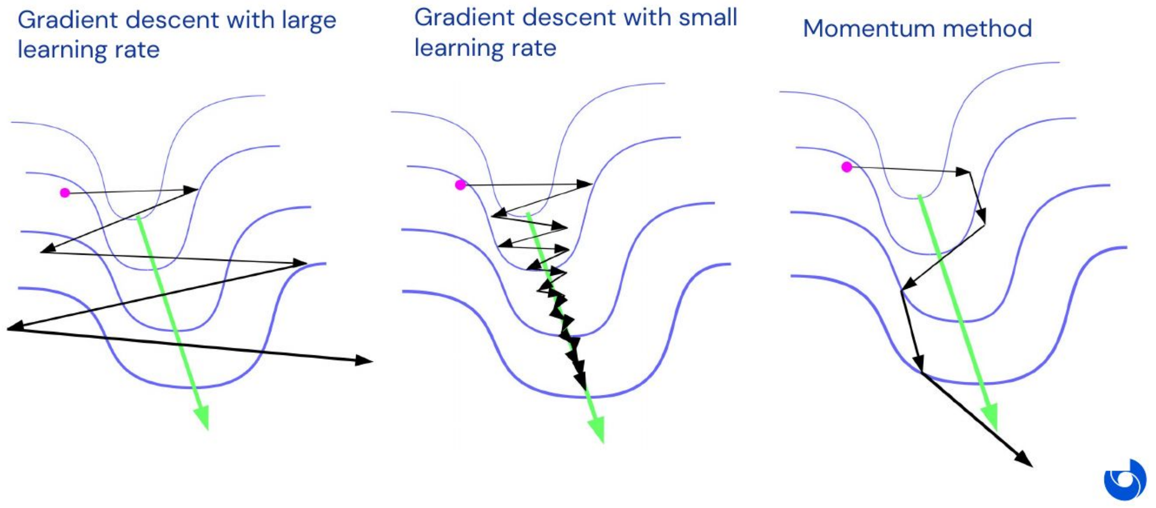
Momentum

“accelarating”

$$v_{k+1} = \alpha\, v_k - g_k$$

“average”

$$\left(v_{k+1} = \alpha\, v_k - (1-\alpha)\, g_k \right)$$


Gradient descent with large learning rate
Gradient descent with small learning rate
Momentum method

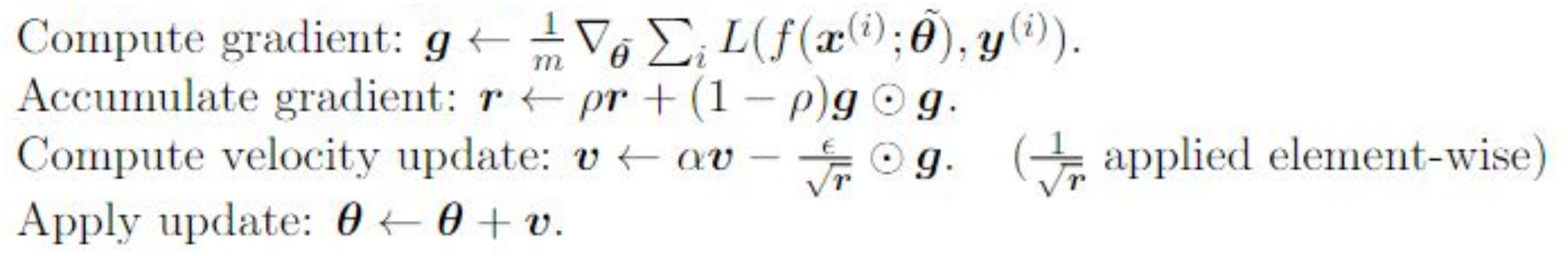
Nesterov momentum (momentum++)

Adaptive learning rates

- RProp
  - Use just the sign of the gradient
  - The sign stays the same? Increase the learning rate (for that parameter)
  - Does it change? Decrease the learning rate
- Not good for stochastic methods
  - The sign can change just because we are using another piece of data
- RMSProp
- RMSProp + momentum
- Adam
- Actual learning rate still there… but much less relevant

$$\mathrm{sign}(g) = \frac{g}{|g|} \simeq \frac{g}{\sqrt{\langle g^2 \rangle_{\mathrm{recent}}}}$$

RMSProp + momentum



Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)};\tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$.

Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$.

Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. $\quad$ ($\frac{1}{\sqrt{\boldsymbol{r}}}$ applied element-wise)

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$.

Adam

Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$

Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$

Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$

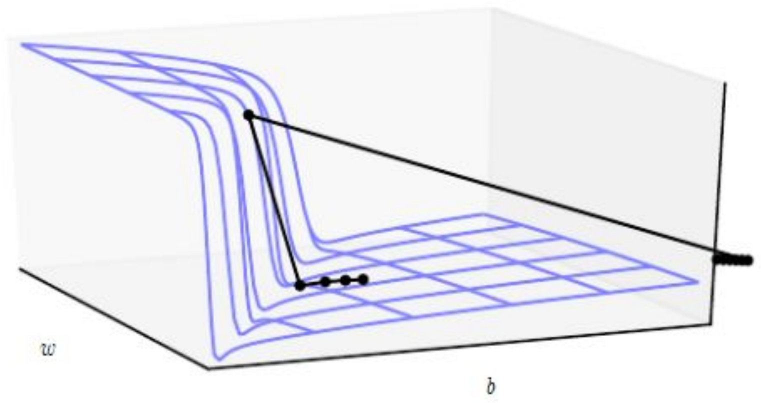
Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$

Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$ (operations applied element-wise)

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

Gradient clipping

- Big idea 💡: clip the gradient!



$$g_{\text{clip}} = \begin{cases} \text{if } g > 1 \rightarrow 1 \\ \text{if } g < -1 \rightarrow -1 \\ \text{else} \rightarrow g \end{cases}$$

Hyperparameters

- All the parameters in the model that you do not change during training...
- Yet that affect your final result (sometimes GREATLY)
- Most common
  - Number of hidden units
  - **Learning rate**
  - Convolution kernel width
  - Weight decay coefficient
  - Dropout rate
- Meta-optimization
  - Manual
  - Systematic (“grid-search”)

Testing your model

Nothing new here

(apart being extra-careful with this gigantic models)

A very partial DL timeline

- 2006: Reducing the dimensionality of data with neural networks
- 2008: Andrew NG's group in Stanford starts advocating for the use of **GPUs** for training Deep Neural Networks
- 2009: Convolutional deep belief networks for unsupervised learning and hierarchical representations
- 2010: **Rectified Linear Units** Improve Restricted Boltzmann Machines
- 2012: Improving neural networks by preventing co-adaptation of feature detectors (**dropout**), Imagenet classification using deep convolutional neural networks (**AlexNet**), Adadelta: an adaptive learning rate method
- 2013: **Auto-encoding variational Bayes**
- 2014: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, **Adam**: a method for stochastic optimization, **Generative Adversarial Networks**
- 2015: Deep residual learning for image recognition, Training very deep networks, **U-Net**: convolutional networks for biomedical image segmentation, **Batch normalization**: accelerating deep network training
- 2016: **Mastering the game of Go** with deep neural networks and tree search, **Tensorflow**: a system for large-scale machine learning, You only look once: unified, real-time object detection (**Yolo**)
- 2017: **Attention is all you need**, Unpaired Image-to-Image Translation using Adversarial Networks
- 2019: **Geoffrey Hinton, Yann LeCun, and Yoshua Bengio wins Turing Award**
- 2020: Language Models are Few-Shot Learners (**GPT**)
- 2021: Zero-Shot Text-to-Image Generation (**Dall-e**)

The Roger Bannister of AI (jokingly)

- On May 6, 1954, Roger Bannister run a mile in 03:59:40
- The 4-minute barrier, until then, seemed so unreachable, almost a physical, ineludible limitation
- Just 46 days later, John Landy finished in 3:58 (a year later, three runners broke the 4-minute barrier in a single race)
- In reality, it was a great intellectual moment!
- A testimony of a incredibly dynamic field (Dir of AI @ Apple 10 years later)
- And more: it all started "generative"...
- And the future seems to lie there...

"that one particular morning when I bumped into Jeff completely changed my future career"
Ruslan Salakhutdinov

See you on tuesday (with Andrea)