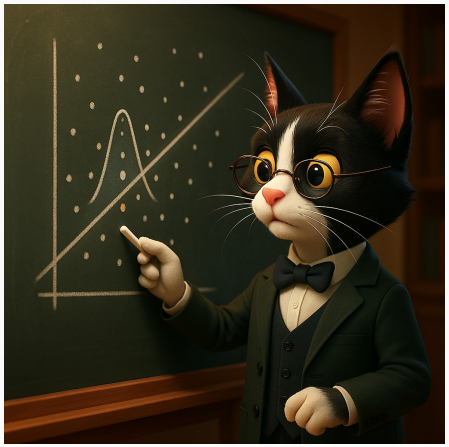
LINEAR MODELS AND INTRO TO DEEP LEARNING

Sergey Nikolenko

Harbour Space University

June 9, 2026



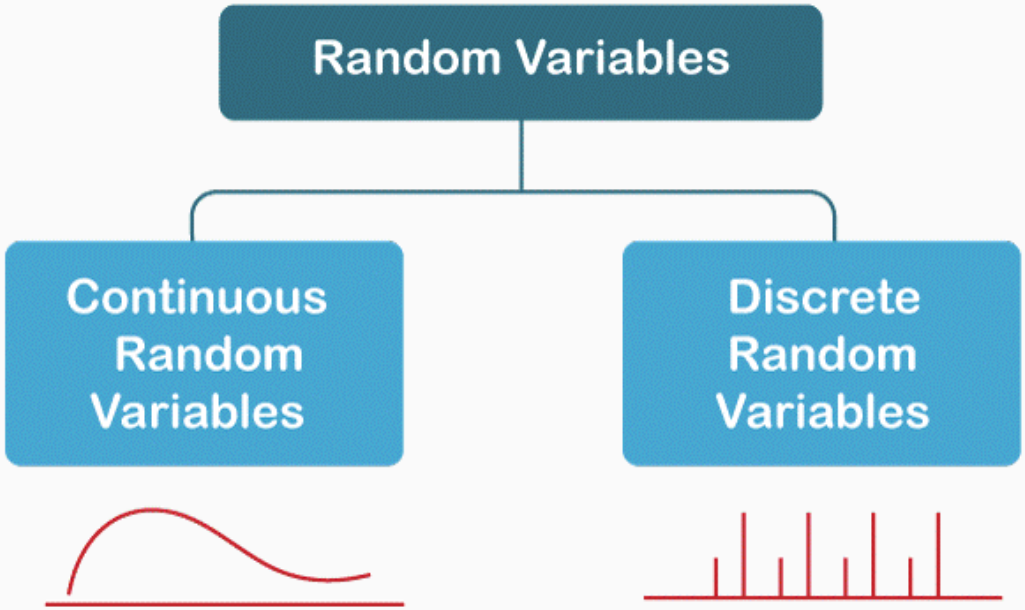
Random facts:

- on June 9, 411 BC, the democratic government in Athens was overthrown in the Athenian coup and replaced by a short-lived oligarchy of the Four Hundred
- on June 9, 53, the Roman emperor Nero married Claudia Octavia, and on June 9, 68 he died by suicide, thus ending the Julio-Claudian dynasty and starting a civil war known as the Year of the Four Emperors
- on June 9, 721, Duke Odo of Aquitaine defeated the Moors (Umayyad Muslim) in the Battle of Toulouse, one of the main battles that stopped the Muslim conquest of Europe
- on June 9, 1732, James Oglethorpe was granted a royal charter to establish a colony in the modern state of Georgia, which he imagined as a place to send "the unemployed and the unemployable"
- on June 9, 1928, Charles Kingsford Smith completed the first trans-Pacific flight in a Fokker Trimotor monoplane, the *Southern Cross*
- on June 9, 1934, Donald Fauntleroy Duck first appeared on screen in *The Wise Little Hen*

Bayesian approach to ML

Basic definitions

- We won't need mathematical definitions of sigma-algebras, probability measures, Borel sets, etc.
- It suffices to understand discrete random variables (probabilities summing to one) and continuous random variables (integral of nonnegative density equals one)



Basic definitions

- *Joint probability* – probability of two events occurring simultaneously, $p(x, y)$; marginalization:

$$p(x) = \sum_y p(x, y).$$

- *Conditional probability* – probability of one event given another occurred, $p(x \mid y)$:

$$p(x, y) = p(x \mid y)p(y) = p(y \mid x)p(x).$$

- *Bayes' theorem* – from above:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_{y'} p(x|y')p(y')}.$$

- *Independence*: x and y are independent if

$$p(x, y) = p(x)p(y).$$

Definitions

- Let us begin with the Bayes' theorem:

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)}.$$

- Here, $p(\theta)$ is the *prior probability*, $p(D|\theta)$ is the *likelihood*, $p(\theta|D)$ is the *posterior probability*, and $p(D) = \int p(D \mid \theta)p(\theta)\mathrm{d}\theta$ is the evidence.

ML vs. MAP

- In statistics, typically we look for the *maximum likelihood hypothesis*:

$$\theta_{ML} = \arg\max_\theta p(D \mid \theta).$$

- Bayesian approaches seek the *posterior distribution*

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

and possibly the *maximum a posteriori hypothesis*:

$$\theta_{MAP} = \arg\max_\theta p(\theta \mid D) = \arg\max_\theta p(D \mid \theta)p(\theta).$$

Direct and inverse problems

- Direct problem: an urn contains 10 balls, 3 of which are black. What is the probability of drawing a black ball?
- Or: an urn contains 10 balls numbered from 1 to 10. What's the probability that the sum of numbers on three consecutively drawn balls is 12?
- Inverse problem: two urns each contain 10 balls; one has 3 black balls, the other 6. Someone took a ball from one urn, and it was black. What's the probability that it was taken from the first urn?
- Notice that in the inverse problem, probabilities immediately become Bayesian (although it can also be reformulated through frequencies).

Direct and inverse problems

- In other words, direct probability problems describe a probabilistic process or model and ask to calculate a specific probability (i.e., predict behavior based on the model).
- Inverse problems contain *hidden variables* (in the example—the number of the urn from which the ball was drawn). They often require building a probabilistic model based on observed behavior.
- Machine learning problems typically belong to the second category.

Probability as frequency

- Classical probability theory, originating from physical experiments, usually views probability as the limit of the ratio of occurrences of a particular experiment outcome to the total number of experiments.
- Standard example: tossing a coin.

Probability as a degree of belief

- But we might want to reason about the "probability" of:
  - Spain winning the FIFA World Cup in 2026;
  - the *Odyssey* having been written by a woman;
  - AGI taking over control from humans by 2030;
  - …
- But "experiments repeated to infinity" are meaningless here—there is exactly one experiment.

Probability as a degree of belief

- Here, probabilities become *degrees of belief*. This is the Bayesian interpretation of probability (as Thomas Bayes originally saw it).
- Fortunately, both interpretations follow the same laws; results show that natural axioms of probabilistic logic lead to a very restricted class of functions (Cox, 19).

Examples

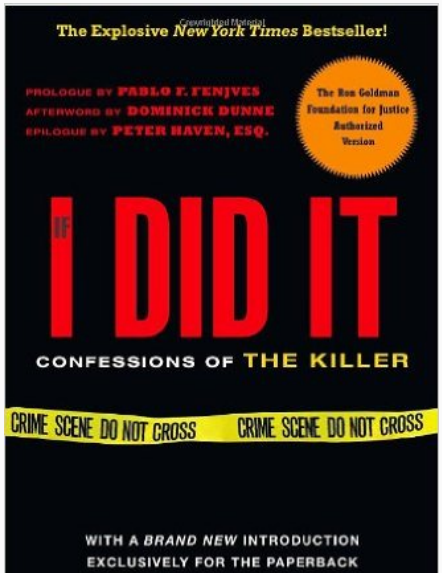
1. A person has two kids. Let's assume that every baby is born either a boy or a girl independently with equal probabilities $\frac{1}{2}$. Two problem settings:
   (1) I've asked if the guy has any boys, and he said yes; what's the probability that he has a girl?
   (2) I've met one of his kids, and it's a boy; what's the probability that the other is a girl?

Examples

2. A murder has happened. Blood at the crime scene is certain to be the killer's, and it's a rare blood type with 1% prevalence... including the defendant.
   (1) Prosecutor says: «The chance that the defendant would have this blood type if he was innocent is only 1%, so with probability 99% he is guilty». What's the prosecutor's mistake?
   (2) Defense attorney says: «A million people live in the city, so about 10000 of them have the same blood type. So all that the blood tells us is that the defendant is guilty with probability 0.01%; that's no proof, please strike that». What is the attorney's mistake?

Examples

3. Real cases.
   (1) The prosecutor said that O.J. Simpson already had a history of domestic violence with his wife. The attorney pointed out: «That's awful but only one out of 2500 women subject to domestic violence actually get killed, so this is irrelevant for the murder». The court agreed with the defense; is this correct inference?
   (2) Two babies of the same mother, Sally Clark, died from SIDS; the prosecutor said that the combined probability of two SIDS cases in a row (obtained from single-case statistic) was about 1 in 73 million; is he right?

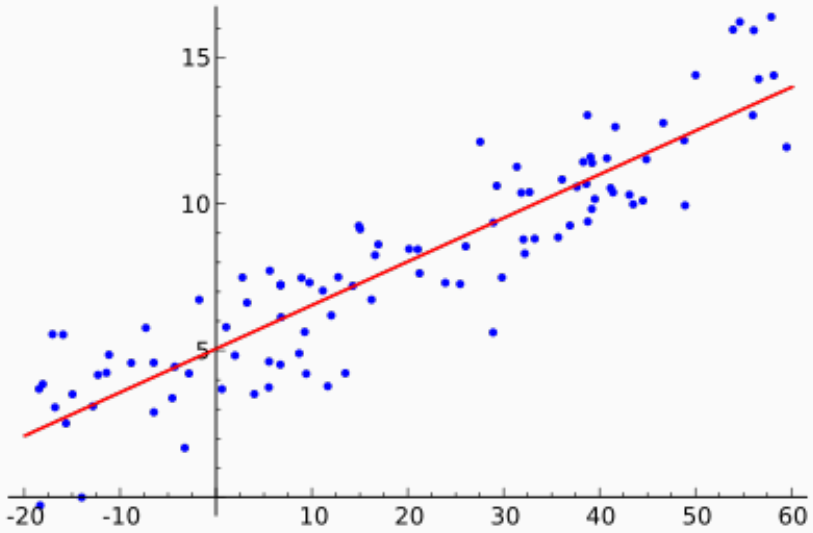


Linear regression

Linear regression

- For example, *linear regression*.
- Linear model: consider a linear function

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{p} x_j w_j = \mathbf{x}^\top \mathbf{w}, \quad \mathbf{x} = (1, x_1, \dots, x_p).$$



- How can we find optimal parameters $\hat{\mathbf{w}}$ by training data of the form $(\mathbf{x}_i, y_i)_{i=1}^N$?

Linear regression

- How can we find optimal parameters $\hat{\mathbf{w}}$ by training data of the form $(\mathbf{x}_i, y_i)_{i=1}^N$?
- Least squares estimation: we will minimize

$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^\top \mathbf{w})^2.$$

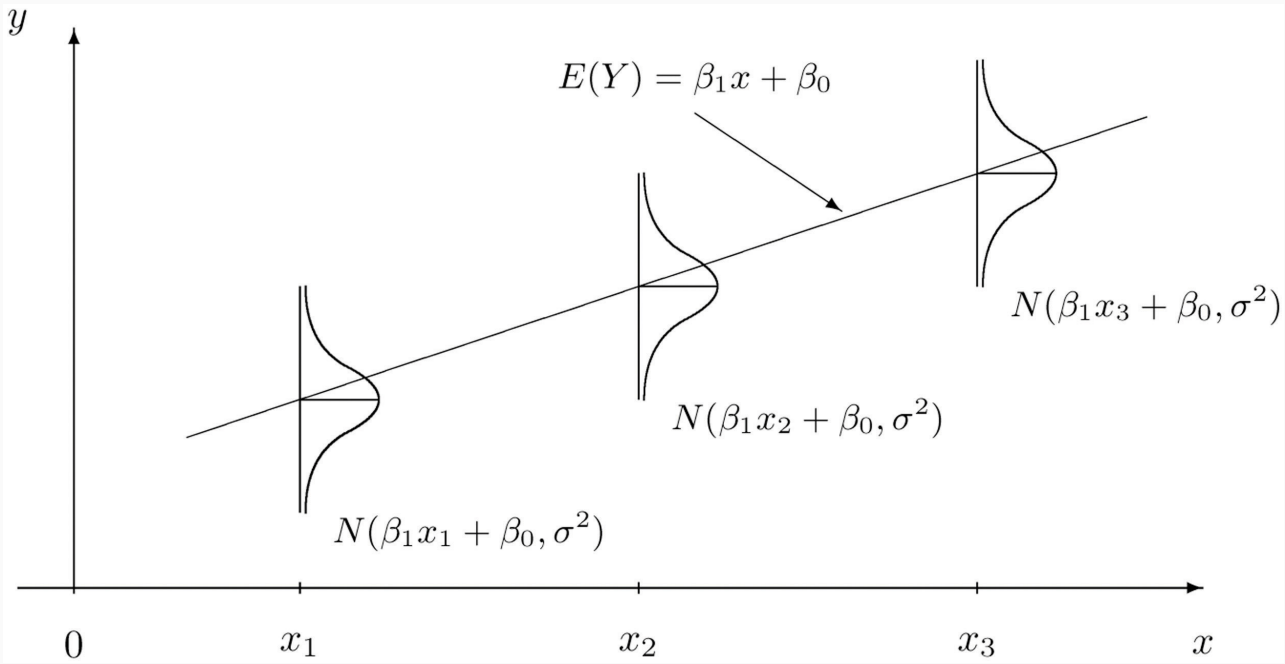
- There is even an exact solution, but that's not important right now.

Linear regression

- What is important: suppose that noise (error in the data) has a normal distribution, i.e., observed variable t is

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \text{ that is}$$

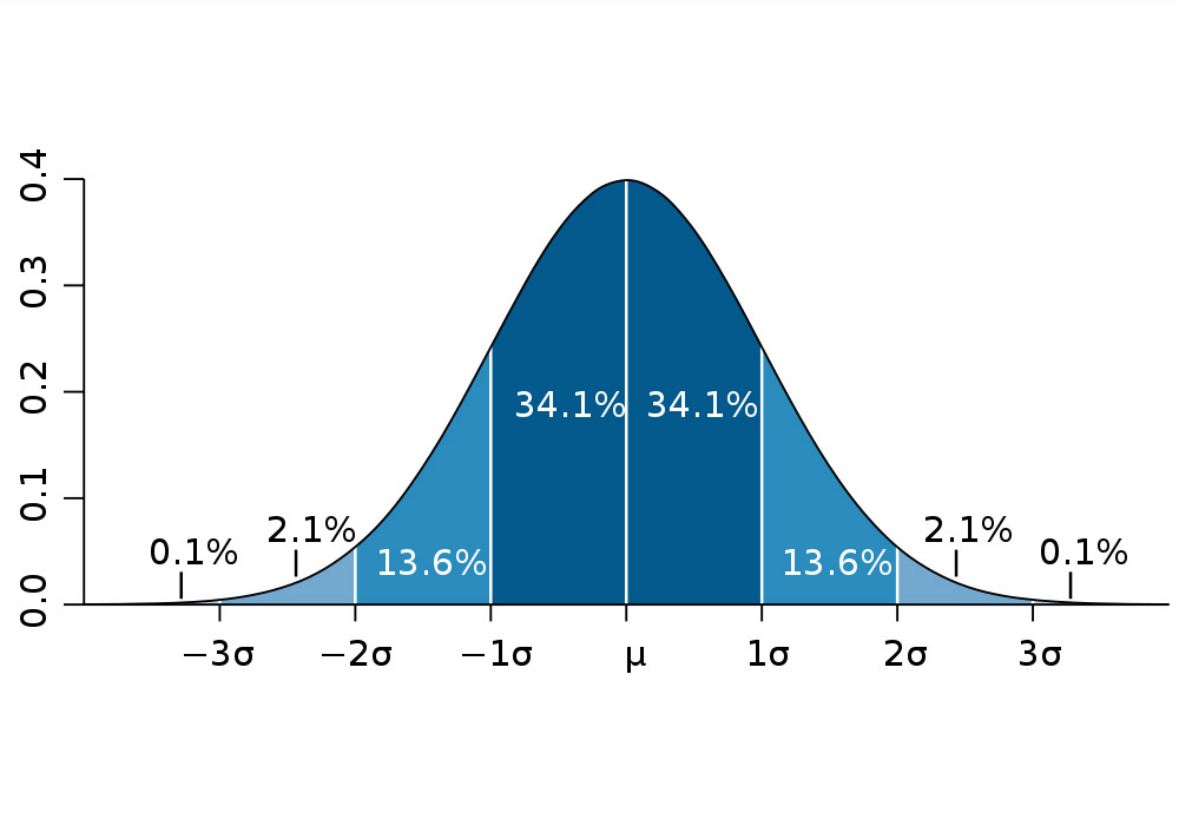
$$p(t \mid \mathbf{x}, \mathbf{w}, \sigma^2) = N(t \mid y(\mathbf{x}, \mathbf{w}), \sigma^2).$$



Linear regression

- Aside – normal distribution:

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



- Why is it so important?

- Consider a dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with correct answers $\mathbf{t} = \{t_1, \dots, t_N\}$.
- We assume that the data points are independent identically distributed:

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} N\left(t_n \mid \mathbf{w}^\top \phi(\mathbf{x}_n), \sigma^2\right).$$

- We take the logarithm (we omit $\mathbf{X}$ below for brevity):

$$\ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right)^2.$$

- And we see that to maximize the likelihood w.r.t. $\mathbf{w}$ we need to minimize the mean squared error!

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right) \phi(\mathbf{x}_n).$$

- We can also get a posterior distribution, introducing prior distributions (also normal).
- And then the predictive distribution

$$p(y \mid \mathbf{x}, D) = \int_{\mathbf{w}} p(y \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w} \mid D) \mathrm{d}\mathbf{w}$$

...but that's beside the point right now.

- Main conclusion: in many regression problems it makes sense to minimize the sum of squared deviations, this corresponds to normally distributed noise.

Bayesian regularization

Bayesian regularization

- And now let us look at regression from the pure Bayesian perspective.
- Recall that in Bayesian inference, we
  (1) find the posterior distribution on the hypotheses/parameters:

$$p(\theta \mid D) \propto p(D|\theta)p(\theta)$$

  (and/or find the maximal a posteriori hypothesis $\arg\max_\theta p(\theta \mid D)$);
  (2) find the predictive distribution:

$$p(x \mid D) \propto \int_{\theta \in \Theta} p(x \mid \theta)p(D|\theta)p(\theta)\mathrm{d}\theta.$$

Bayesian regularization

- We have not yet had any priors in our study of linear regression.
- Let us introduce a prior; e.g., the normal distribution:

$$p(\mathbf{w}) = N(\mathbf{w} \mid \mu_0, \Sigma_0).$$

- Consider a dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with values $\mathbf{t} = \{t_1, \dots, t_N\}$; we again assume that they are independent and identically distributed:

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} N\left(t_n \mid \mathbf{w}^\top \phi(\mathbf{x}_n), \sigma^2\right).$$

Bayesian regularization

- Then the problem is to compute

$$p(\mathbf{w} \mid \mathbf{t}) \propto p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \sigma^2) p(\mathbf{w})$$
$$= N(\mathbf{w} \mid \mu_0, \Sigma_0) \prod_{n=1}^{N} N\left(t_n \mid \mathbf{w}^\top \phi(\mathbf{x}_n), \sigma^2\right).$$

- Let us compute!

Bayesian regularization

- We get

$$
\begin{aligned}
p(\mathbf{w} \mid \mathbf{t}) &= N(\mathbf{w} \mid \mu_N, \Sigma_N), \\
\mu_N &= \Sigma_N \left(\Sigma_0^{-1} \mu_0 + \frac{1}{\sigma^2} \Phi^\top \mathbf{t} \right), \\
\Sigma_N &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^2} \Phi^\top \Phi \right)^{-1}.
\end{aligned}
$$

- And now the log likelihood.

Bayesian regularization

- If we take the prior distribution around zero:

$$p(\mathbf{w}) = N(\mathbf{w} \mid 0, \frac{1}{\alpha}\mathbf{I}),$$

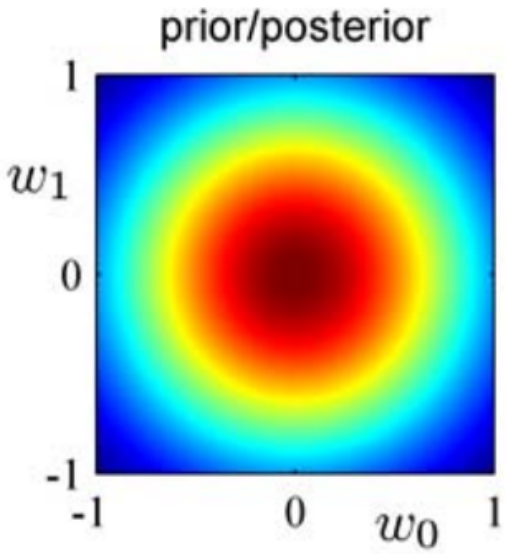
we get the log likelihood as

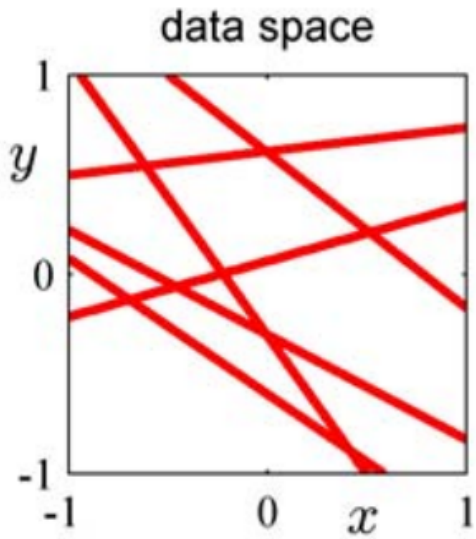
$$\ln p(\mathbf{w} \mid \mathbf{t}) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right)^2 - \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w} + \text{const},$$

i.e., precisely ridge regression!

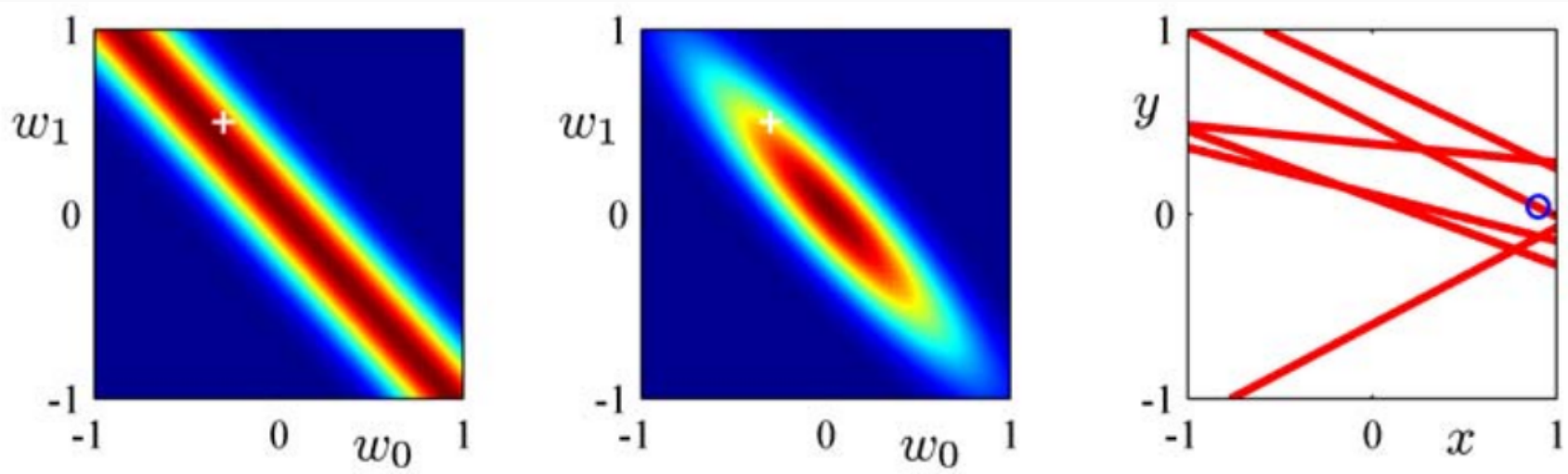
Example

likelihood

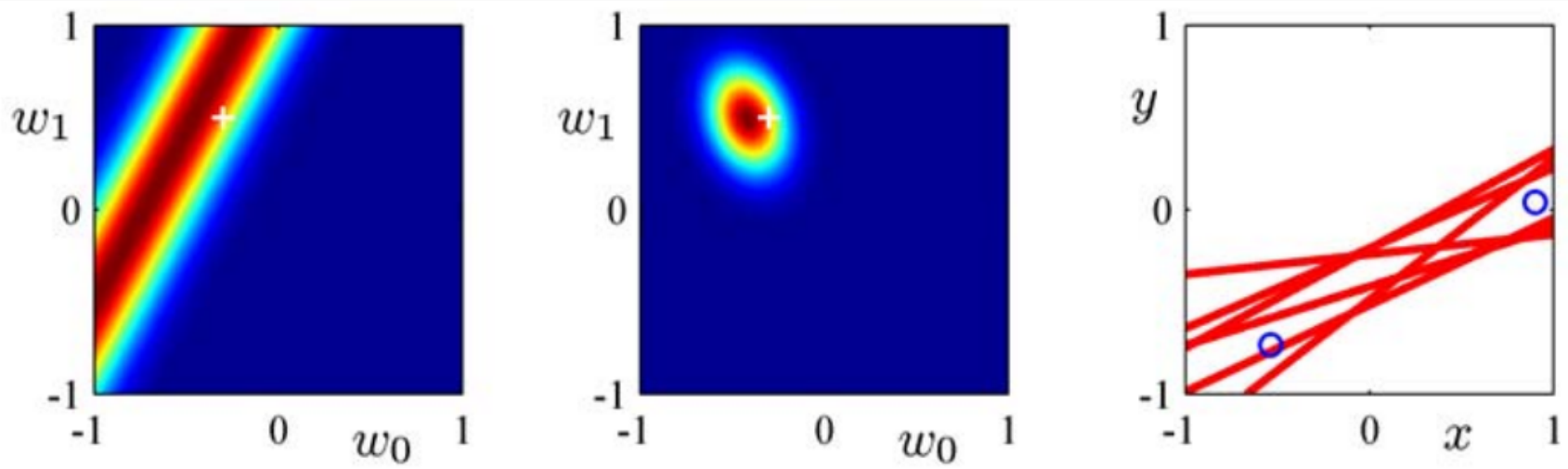




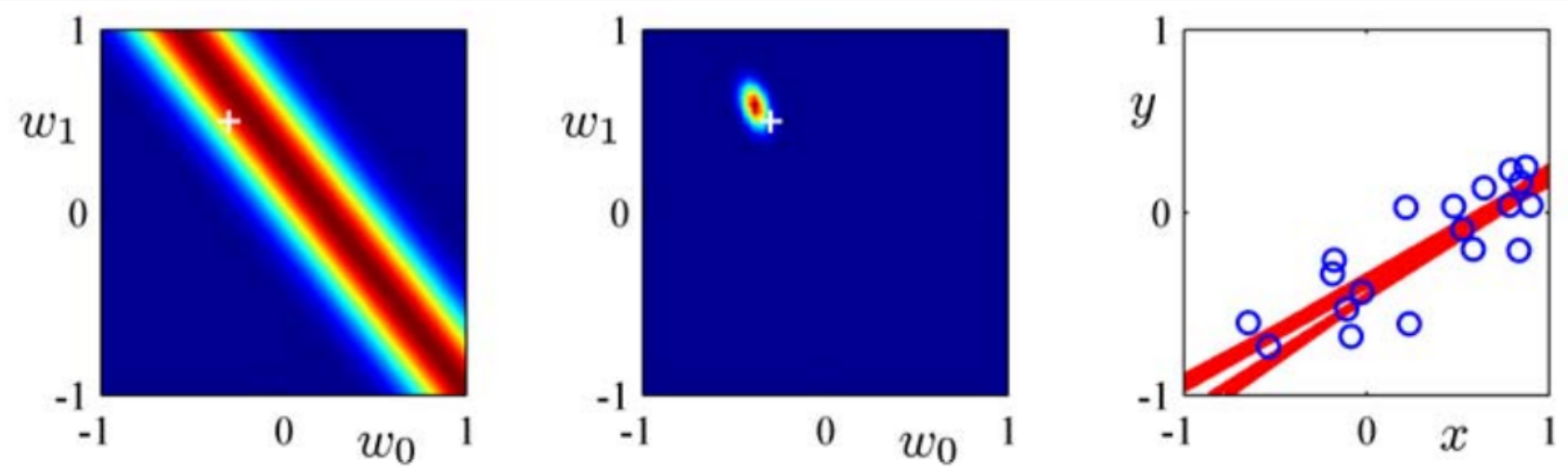
Example



Example



Example



Generalization

- A slight generalization – a more general prior distribution:

$$p(\mathbf{w} \mid \alpha) = \left[\frac{q}{2} \left(\frac{\alpha}{2} \right)^{1/q} \frac{1}{\Gamma(1/q)} \right]^M e^{-\frac{\alpha}{2} \sum_{j=1}^{M} |w_j|^q}.$$

Exercise. Compute the log likelihood.

Classification

Classification problems

- For classification problems it is even more clear: we want to classify a vector $\mathbf{x}$ to one of K classes C_k.
- Suppose that class C_k has density $p(\mathbf{x} \mid C_k)$, find prior distributions $p(C_k)$, and then compute $p(C_k \mid \mathbf{x})$ by Bayes' theorem.
- For two classes:

$$p(C_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_1)p(C_1) + p(\mathbf{x} \mid C_2)p(C_2)}.$$

- We rewrite:

$$p(C_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_1)p(C_1) + p(\mathbf{x} \mid C_2)p(C_2)} = \frac{1}{1 + e^{-a}} = \sigma(a),$$

where

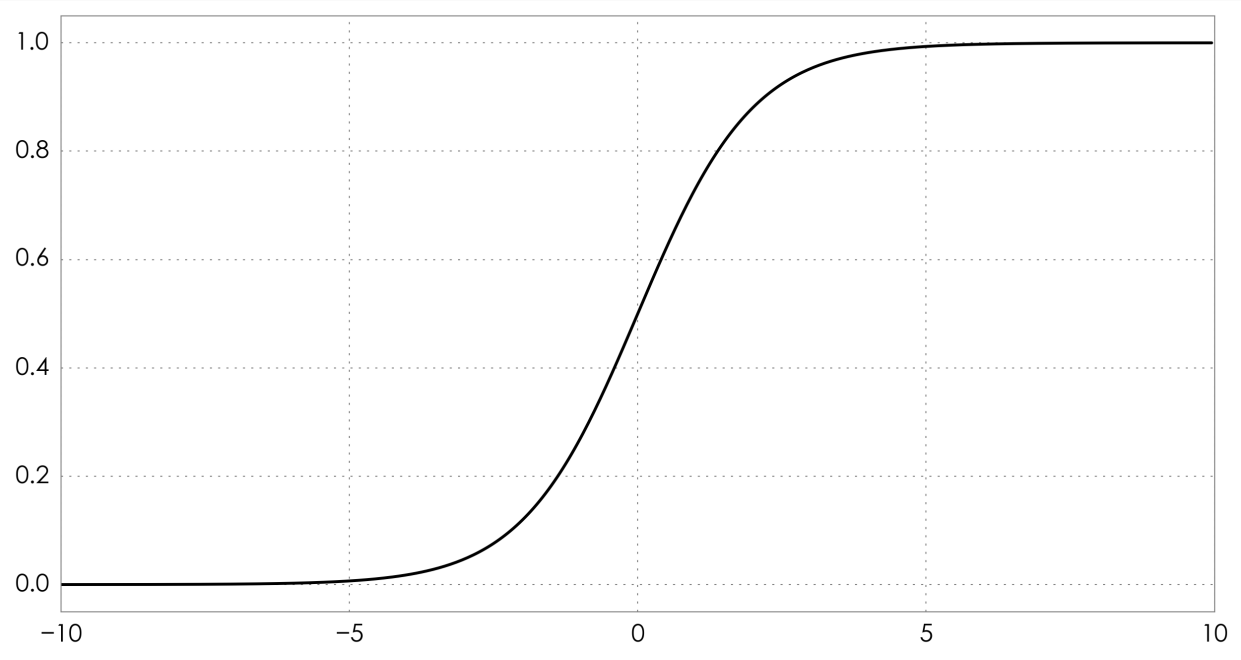
$$a = \ln \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_2)p(C_2)}, \qquad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

Classification problems

- $\sigma(a)$ is the *logistic sigmoid*:

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

- $\sigma(-a) = 1 - \sigma(a)$.
- $a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ – *logit function*.



- This, in particular, leads to *logistic regression*: we optimize $\mathbf{w}$ directly.
- For a dataset $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$:

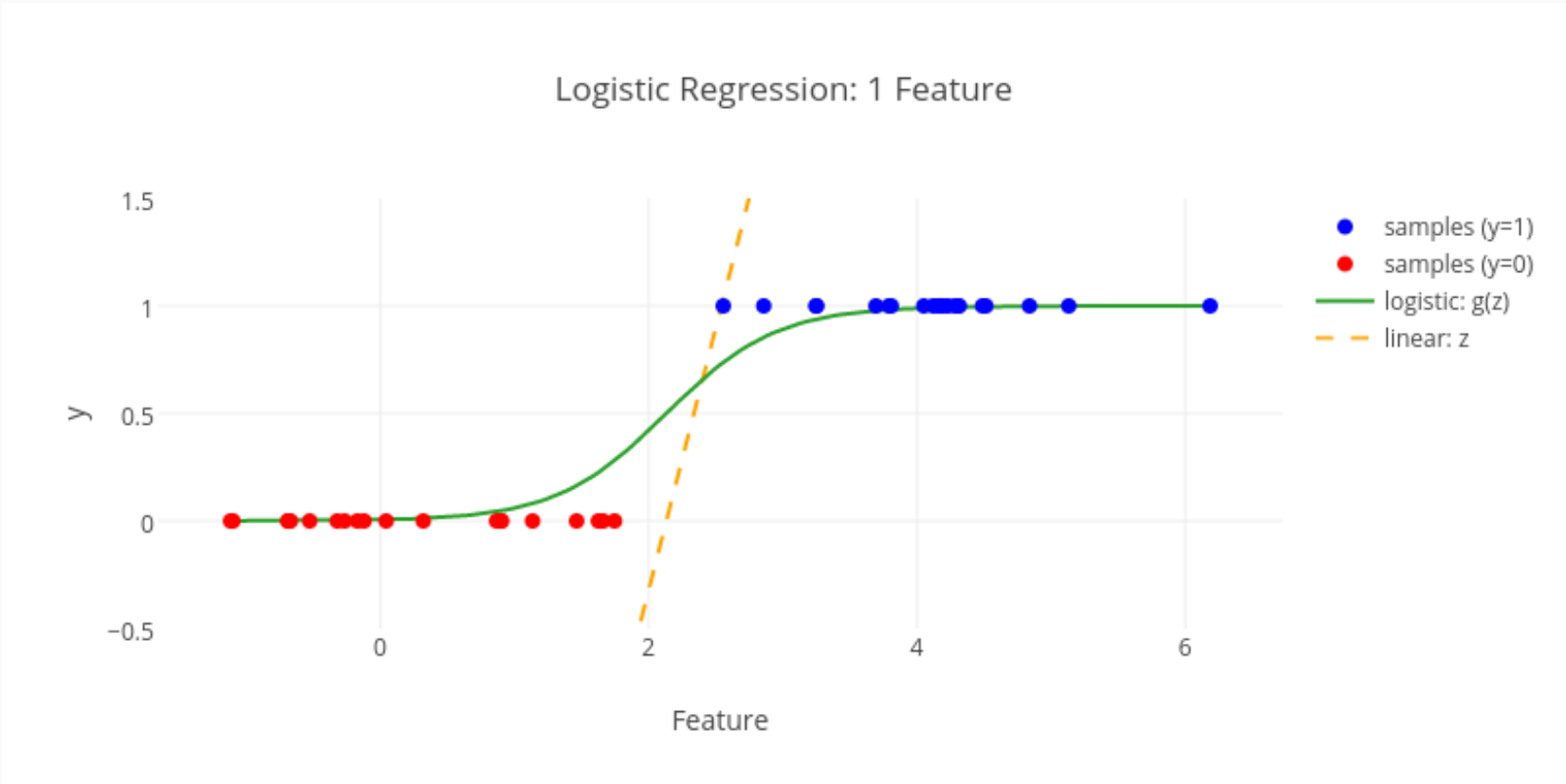
$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}, \quad y_n = p(C_1 \mid \phi_n).$$

- We find maximal likelihood parameters, minimizing $-\ln p(\mathbf{t} \mid \mathbf{w})$:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

Classification problems

- And we get a sigmoid that optimally separates the data and that even tries to model probabilities:



Several classes

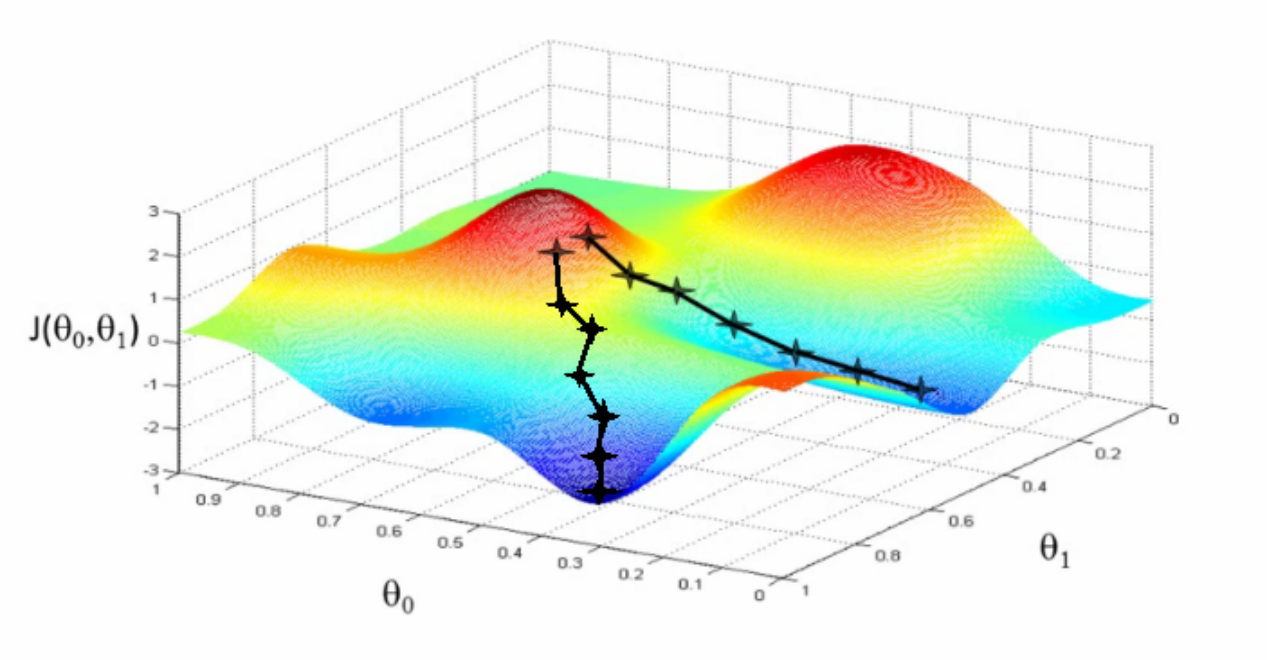
- In case of several classes we get

$$p(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k)p(C_k)}{\sum_j p(\mathbf{x} \mid C_j)p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}}.$$

- Here $a_k = \ln p(\mathbf{x} \mid C_k)p(C_k)$.
- $\frac{e^{a_k}}{\sum_j e^{a_j}}$ is the normalized exponent (softmax).
- Conclusion: for a classification problem it makes sense to minimize the cross-entropy $\sum_{n=1}^{N} [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)]$ and softmax (rather than classification error, which is problematic).
- One question remains: how do we optimize all this?
- How do we optimize complicated functions in general?

Gradient descent

- Gradient descent: virtually the only way to optimize complicated non-convex functions.



- Take the gradient $\nabla E(\mathbf{w})$ w.r.t. weights, move in that direction.

Gradient descent

- E.g., for logistic regression we can optimize

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

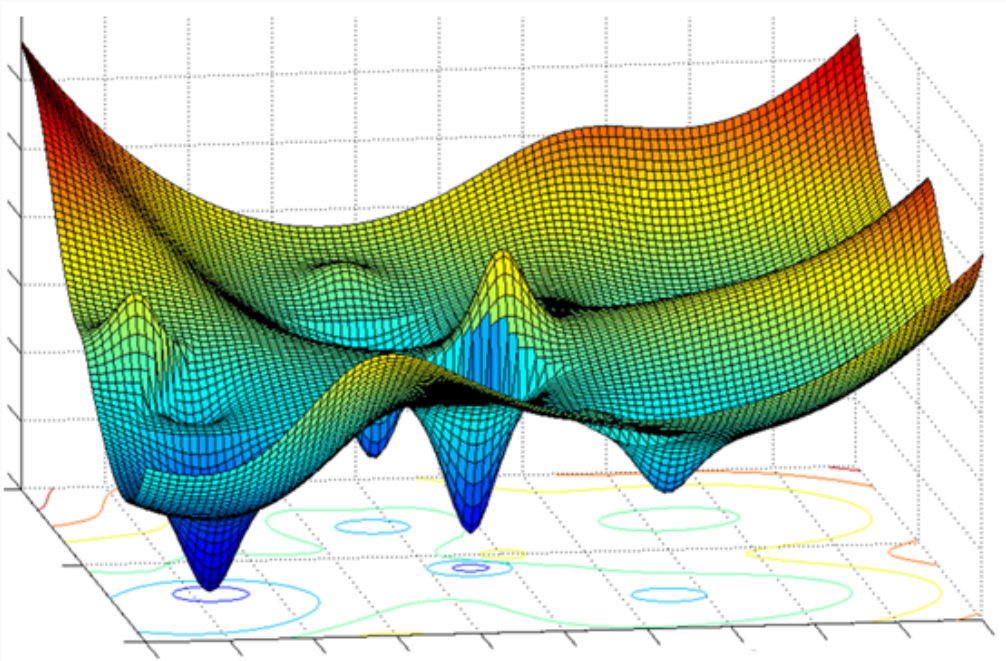
- We use the fact that $\sigma' = \sigma(1 - \sigma)$.
- Take the gradient:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \phi_n.$$

- And then we can simply use gradient descent (or do even better with IRLS).

Gradient descent

- Gradient descent is a local optimization procedure.



- But there are no global ones… we will only talk of local improvements of gradient descent, but there will never be a guarantee with these methods.

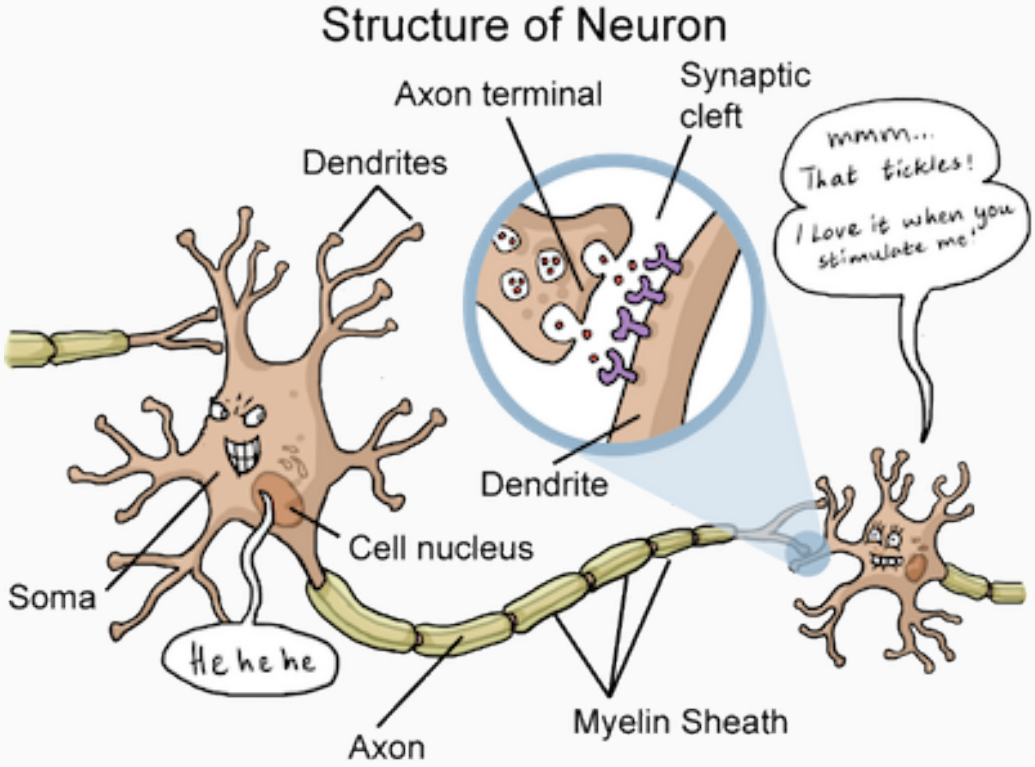
The brain and ANN history

Human brain as a computer

- The human brain is a pretty slow computer, but it can do some pretty amazing things.
- How does it do it?
- Lots of neurons: 10^{11} neurons, about 7000 synapses each, so in total about 10^{14}–10^{15} connections (synapses).

Human brain as a computer

- Each neuron:
  - gets a signal through the dendrites from other neurons;
  - sends signals from time to time through the axon;
  - synapses are connections between one neuron's axon and other neurons' dendrites attached to it.

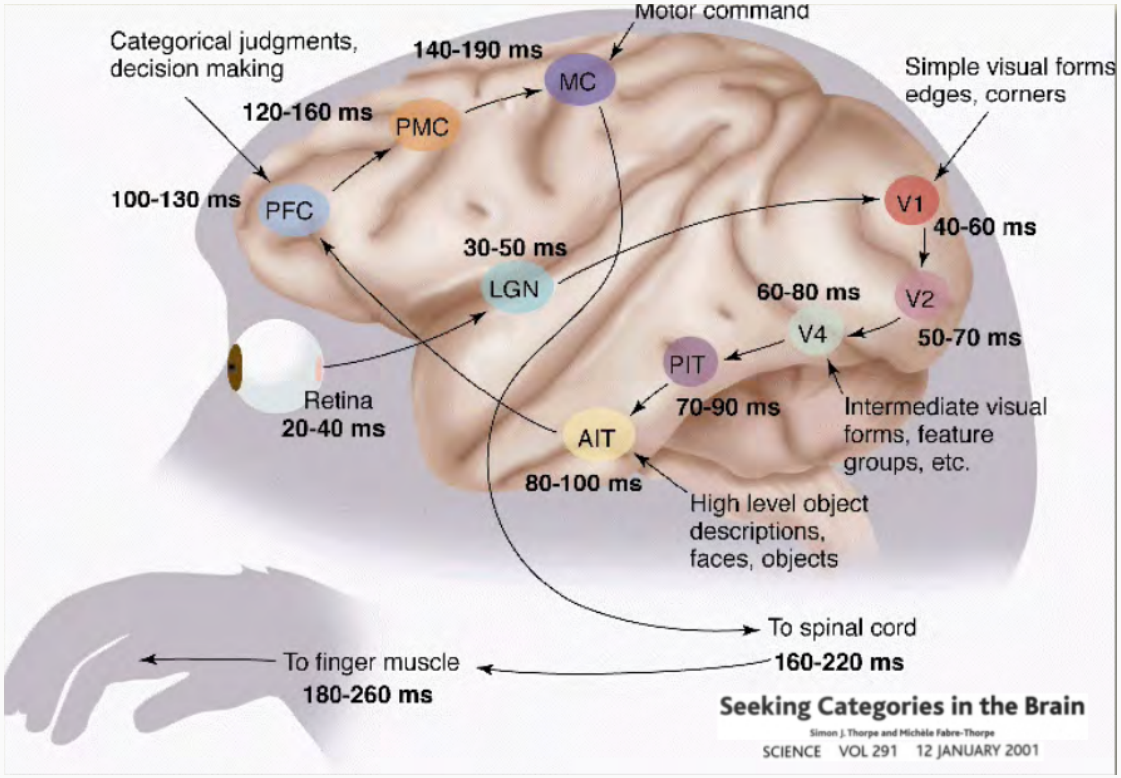


Human brain as a computer

- A neuron produces spikes stochastically.
- The firing rate is only 10 to 200 Hz.
- But we can recognize a face in a couple hundred milliseconds.
- This means that sequential computations are very short.
- But the brain is heavily parallelized.

Human brain as a computer

- Here is how we process visual input:



Feature learning

- Another part of the story: feature learning.
- The brain can train very well on a very small data sample.
- And it can adapt to new data sources.
- How does it do it? Can we do the same?

Feature learning

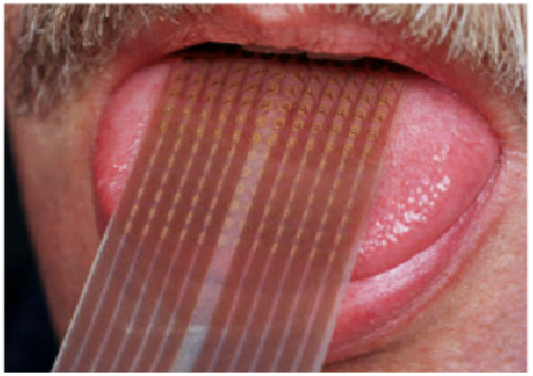
- Systems for processing unstructured data look like

$$\text{input} \to \text{features} \to \text{classifier}$$

- For a long time good features had to be handmade:
  - MFCC for speech recognition;
  - SIFT for image processing;
  - ...
- Feature engineering: can we learn good features automatically?

Plasticity

- Third point: neuroplasticity.



- It appears that there is a single learning algorithm (“The Master Algorithm”) that can be applied to very different situations.

The mathematical essence

- Generally speaking, all we do in machine learning is approximation and optimization of various functions.
- For example:
  - a function from a picture to what it shows;
  - even simpler: binary function from an image's pixels to "is it a cat or not";
  - it's a rather complicated function, and it's defined as a dataset (values at a few points);
  - how do we approximate it?

The mathematical essence

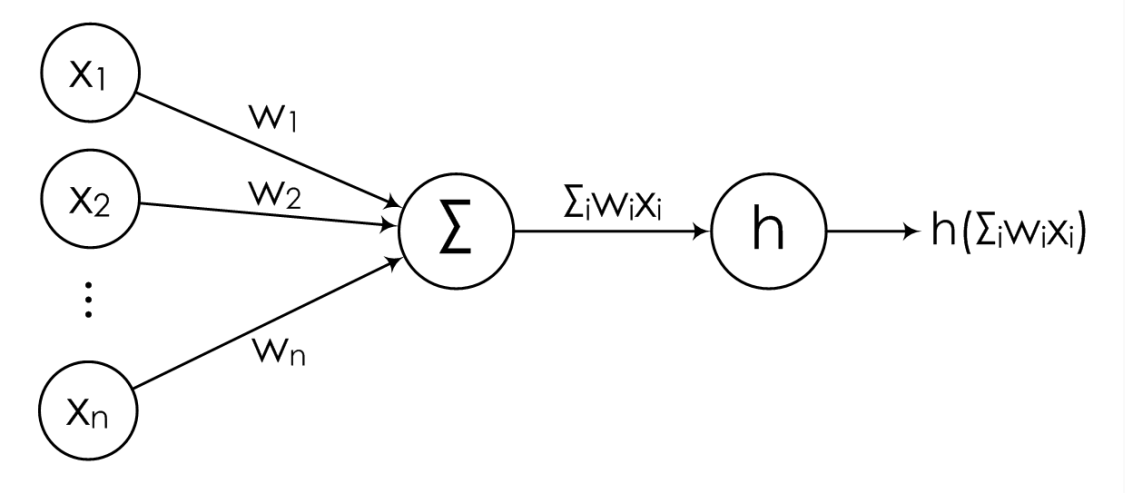
- Common approach in machine learning:
  - construct a class of models, usually parametric;
  - tune the parameters so that the model fits the data well;
  - i.e., we optimize a certain error function or quality metric.
- Neural networks are basically universal approximators, a very wide and flexible parametric class of functions.
- The problem is how to train them well.

ANN history

- Let's try to mimic nature: get a large highly connected thing of small simple things.
- The idea of modeling neurons with simple functions is very-very old (McCulloch, Pitts, 1943).
- Artificial neural networks (even with several layers) were proposed, among others, by Turing (1948).
- Rosenblatt, 1958: perceptron (one artificial neuron, will see in a minute): a linear separating surface.
- Training by gradient descent also appeared at the same time.

ANN history

- One neuron is modeled as follows:



- Linear combination of inputs follows by a nonlinear activation function:

$$y = h(\mathbf{w}^\top \mathbf{x}) = h\left(\sum_i w_i x_i\right).$$

- How can you train a perceptron and what will be the result?

- Training by gradient descent.
- Original Rosenblatt's perceptron ($h = \mathrm{id}$):

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}\in\mathcal{M}} y(\mathbf{x})\left(\mathbf{w}^\top \mathbf{x}\right),$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla_{\mathbf{w}} E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta t_n \mathbf{x}_n.$$

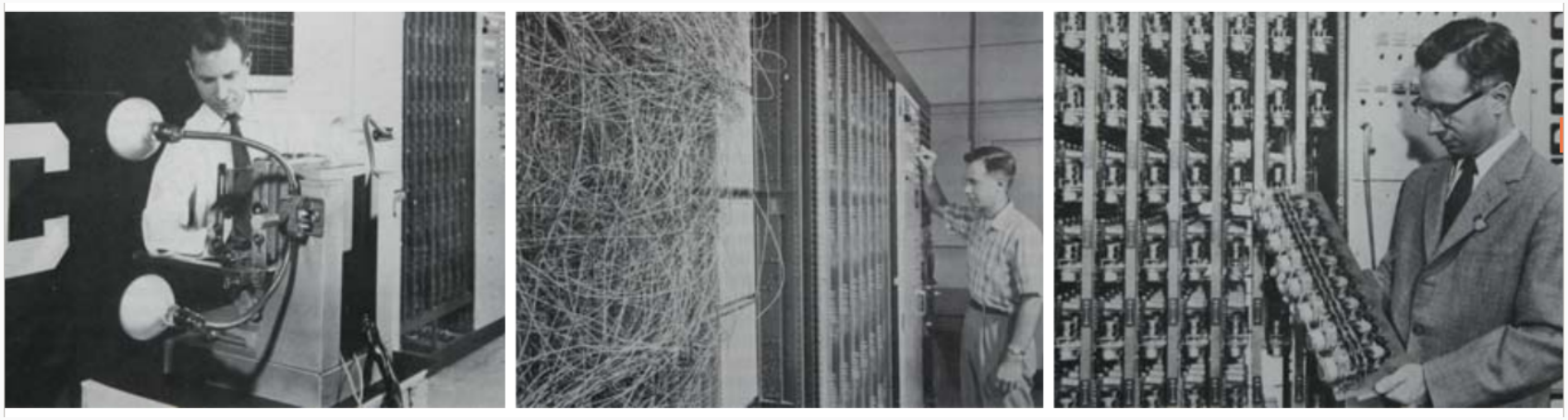
- Or, e.g., we can do binary classification by plugging in logistic sigmoid as the activation function $h(x) = \sigma(x) = \frac{1}{1+e^{-x}}$:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1-y_i)\log\left(1-\sigma(\mathbf{w}^\top \mathbf{x}_i)\right)\right).$$

- Basically logistic regression.

ANN history

- The first perceptron (Rosenblatt, 1958) learned to recognize letters.



ANN HISTORY

- Hailed as a huge success:

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

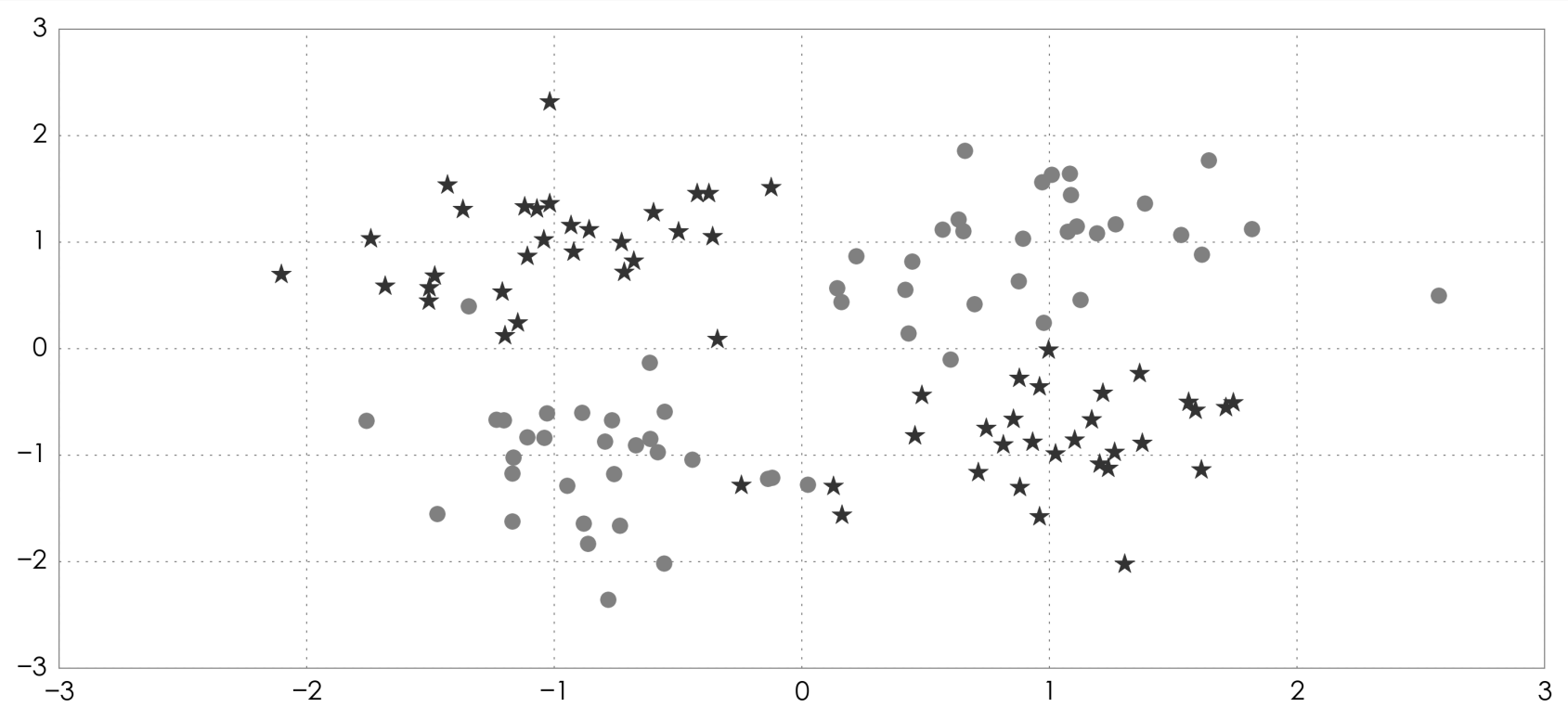
In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

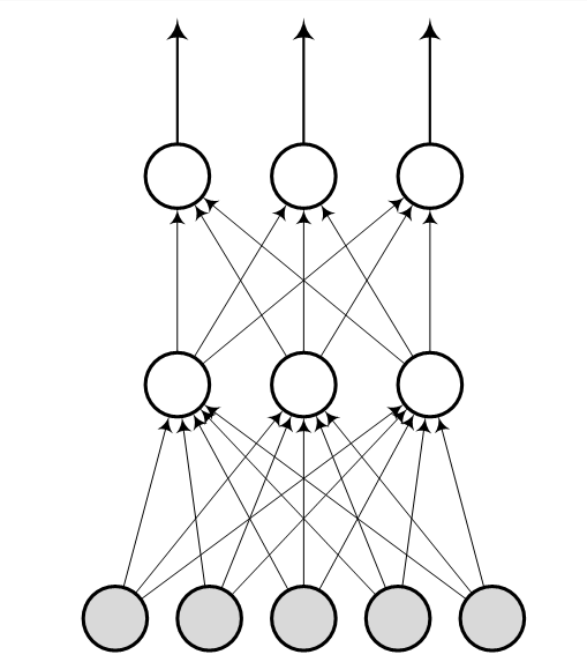
ANN history

- Minsky's critique: even with a nonlinearity, one perceptron can only have a linear separating surface.



ANN history

- Well, sure! You need to make a network of perceptrons:



- More about that later (actually, the whole course is about that).
- But first let's see what people do with a single perceptron.

ANN history

- 1960s: studies of the perceptron.
- (Minsky, Papert, 1969): XOR cannot be modeled by a perceptron.
- For some reason, this was thought to be a big problem.
- (Brison, Hoback, 1969): backpropagation.
- (Hinton, 1974): rediscovered backpropagation and popularized it.
- Second half of the 1970s – multilayer ANN, basically in the modern form.
- Deep models appeared in early 1980s! It's not much of an idea.

ANN history

- But by the early 1990s neural networks did not live up to the hype (in part for technical reasons).
- John Denker, 1994: «neural networks are the second best way of doing just about anything».
- So in the 1990s, neural networks were all but forgotten again.
- Except for image processing (Yann LeCun) and a few more groups (Geoffrey Hinton, Yoshua Bengio).

ANN history

- Deep learning revolution begins in 2006: Geoffrey Hinton's group learned how to train Deep Belief Networks (DBN).
- Main idea: *unsupervised pretraining*.
- Next component: more powerful computers, computations on GPUs.
- And much larger datasets.
- Then we got new regularization and optimization techniques (we'll talk about those), and pretraining by now is not really necessary.
- These, together with the architectures, are the main components of the deep learning revolution.

Real neurons

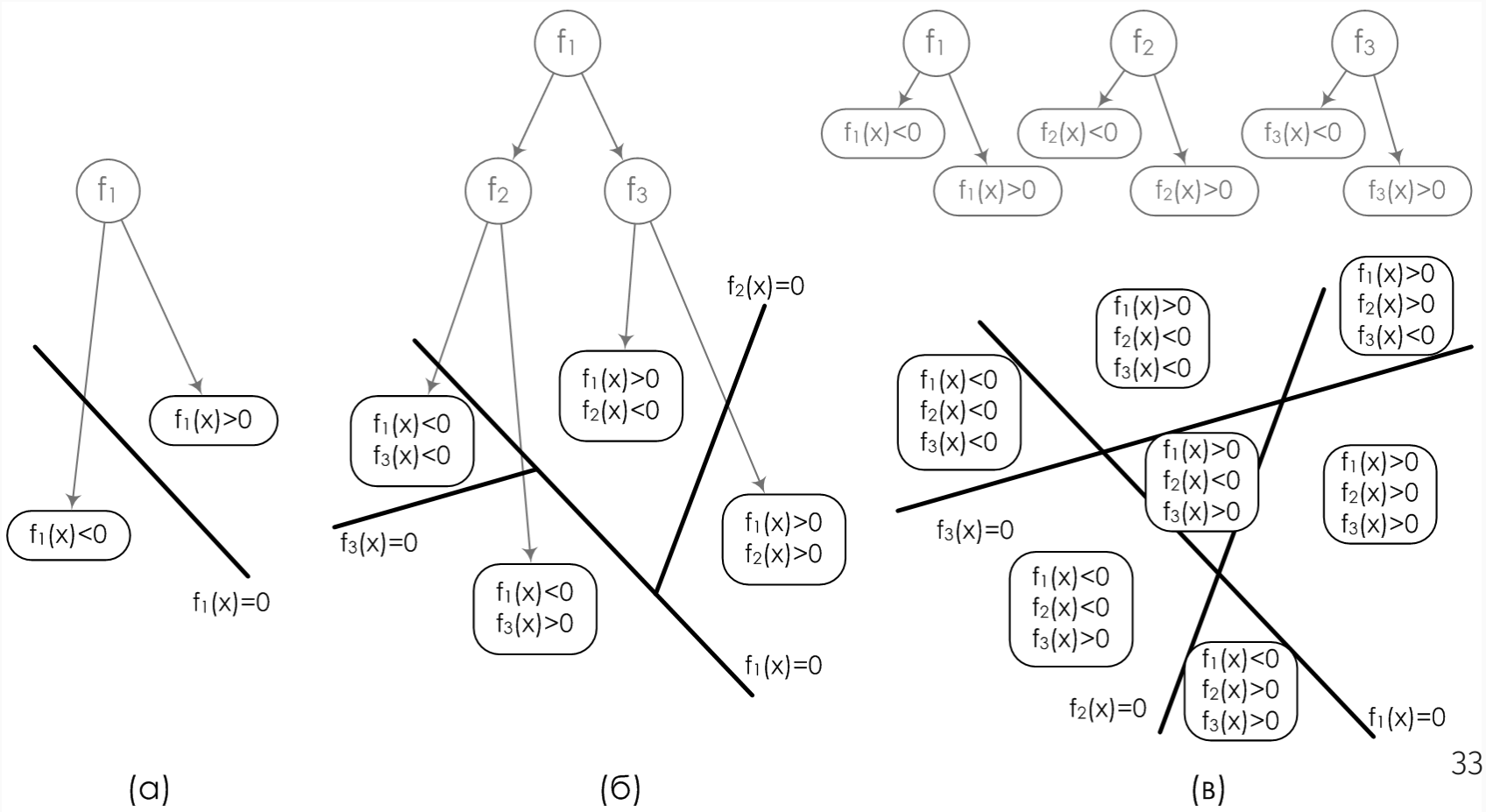
- Do actual neurons do gradient descent?
- Probably not (we'll see that backprop would be hard to do).
- Although Geoffrey Hinton had an interesting talk about it. But still, no.
- *Hebbian learning* – increase the weight between neurons that fire together:

$$\Delta w_i = \eta x_i x_j.$$

- Led to *Hopfield networks* that kinda do associative memory.
- Current research: *spike-timing-dependent plasticity*; i.e., the increase depends on the actual temporal properties of signals.

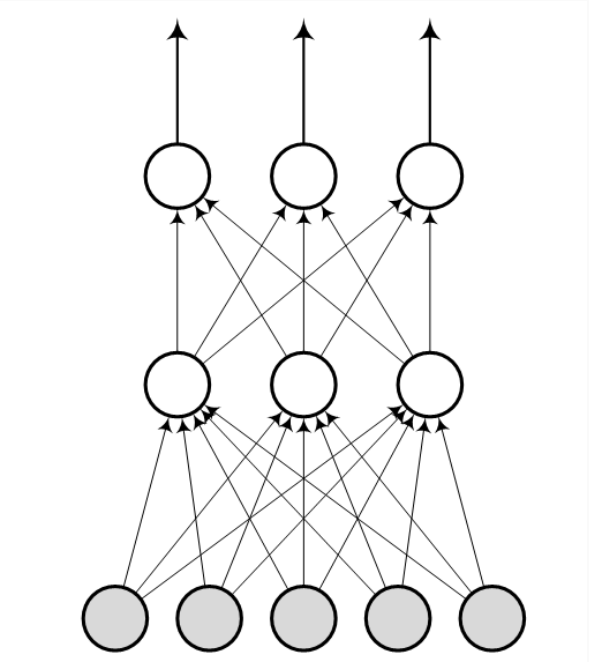
Combining perceptrons into a network

- A network of perceptrons; outputs of one are inputs of another.
- Hornik, 1990: a two-layer ANN can approximate any function.
- This is just theory, but in practice, deep networks are indeed more expressive – distributed representations:



Combining perceptrons into a network

- Note also how they NNs are usually organized in *layers*.
- This is natural and suggests easy parallelization.



Combining perceptrons into a network

- So we approximate a very complicated function with a large composition of simple functions.
- How do we train it?
- Simple answer: gradient descent. But there are complications here.

Thank you!

Thank you for your attention!