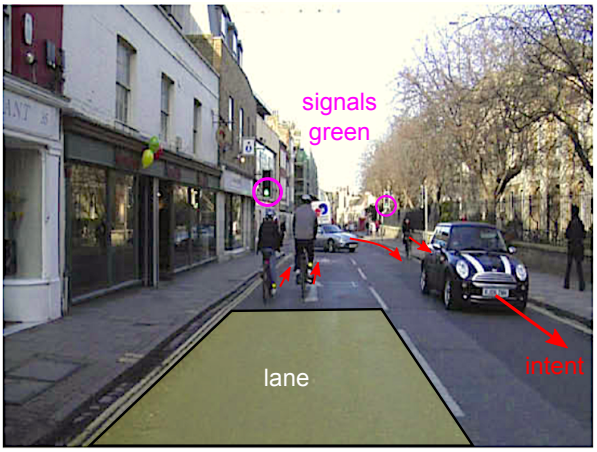
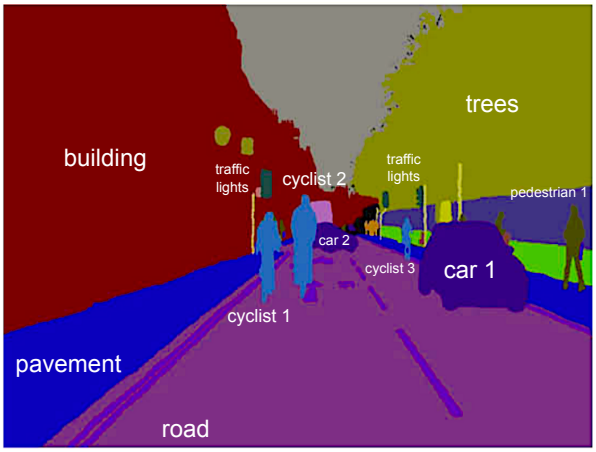
Deep Learning
Part 1: From single neurons to multi-layer perceptrons

Dr. Richard E. Turner (ret26@cam.ac.uk)

Engineering Tripos Part IB
Paper 8: Information Engineering

Example perception problems for a self-driving car





Very hard to hand-design systems to do this reliably
fairly easy to collect lots of labelled data
$\Longrightarrow$ learn systems from data instead

Object recognition

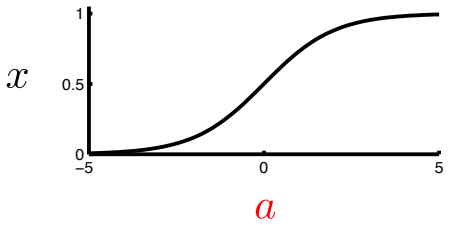
- Focus on **objection recognition**: predict objects present in an image
- A canonical classification problem: approaches to object recognition can be generalised to more complex tasks
- All state-of-the-art approaches use **neural networks**
- Outline of lectures:
  - single neurons and logistic classification (2 lectures)
  - neural networks and multi-layer perceptrons (1 lecture)
  - convolutional neural networks (2 lectures)
  - worked examples integrated through out (no specific examples class)

Current state-of-the-art

https://www.clarifai.com/demo

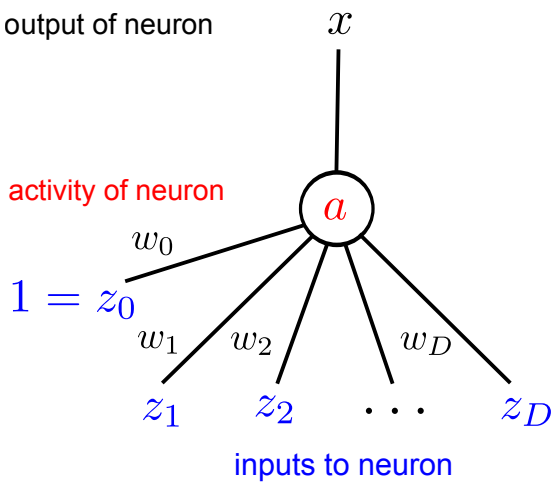
A single neuron

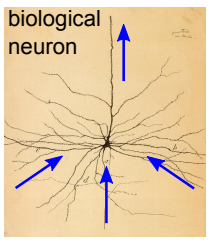
Artificial neuron computes a non-linear function of inputs:



$$x(a) = \frac{1}{1+\exp(-a)} \qquad x \in (0,1)$$

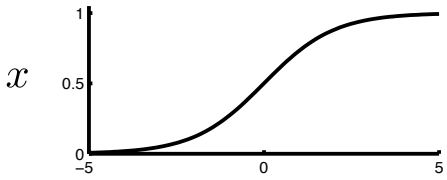
$$a = w_0 + \sum_{d=1}^{D} w_d z_d$$

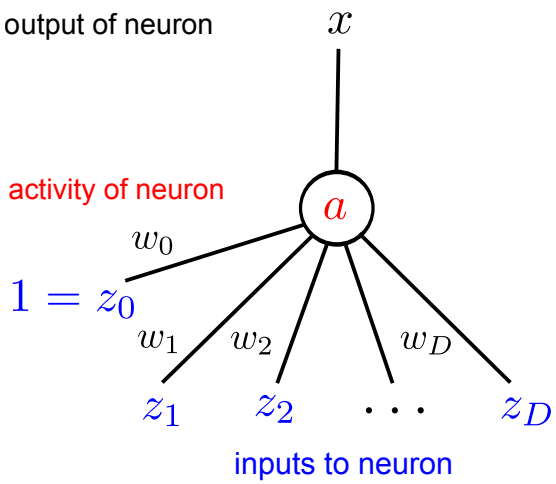




A single neuron

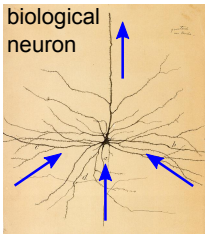
Artificial neuron computes a non-linear function of inputs:



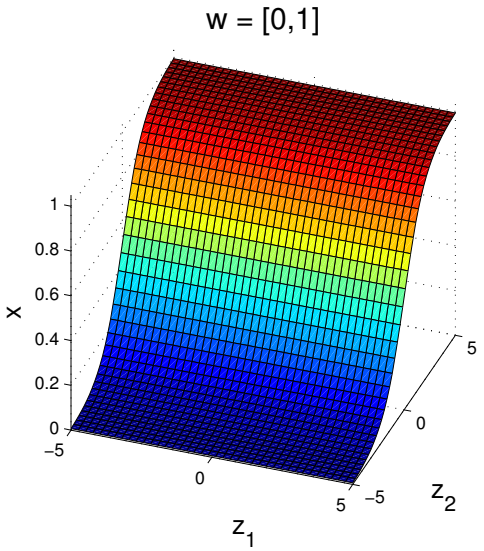


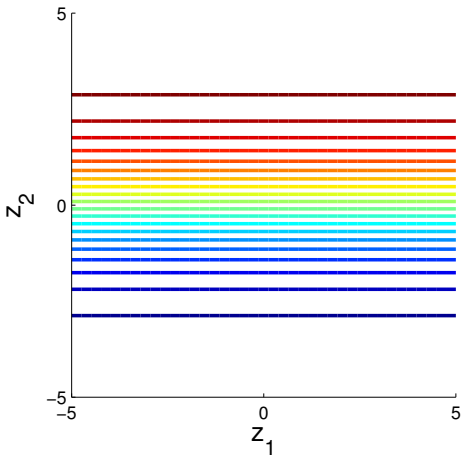
$$x(a) = \frac{1}{1+\exp(-a)} \qquad x \in (0,1)$$

$$a = \sum_{d=0}^{D} w_d z_d$$



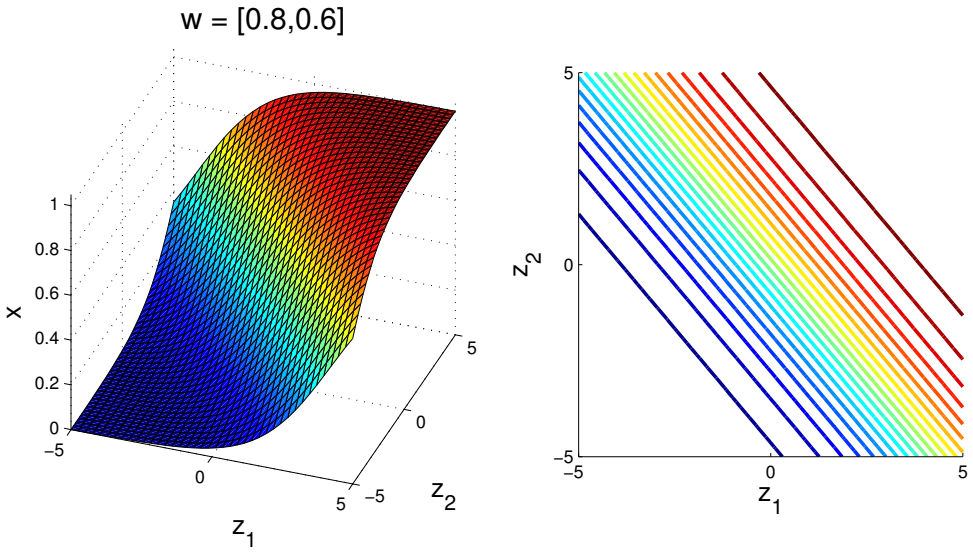
Input-output function of a single neuron





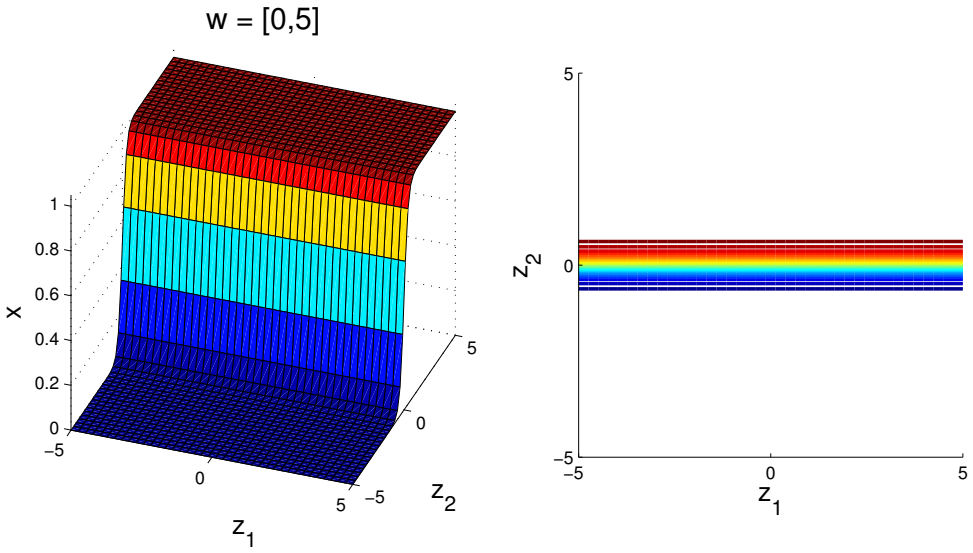
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-output function of a single neuron



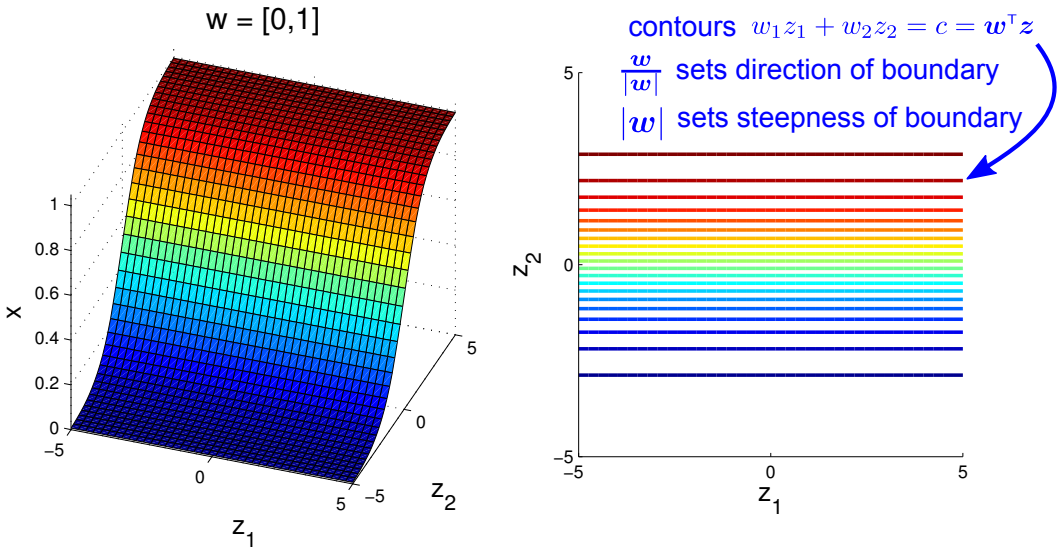
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-output function of a single neuron



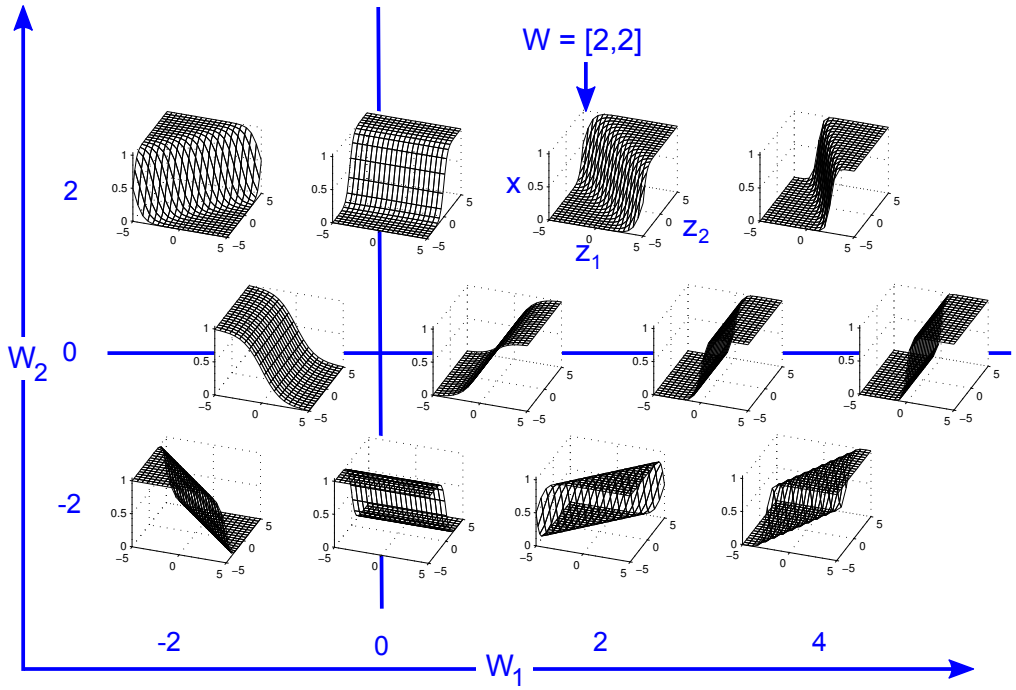
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-output function of a single neuron

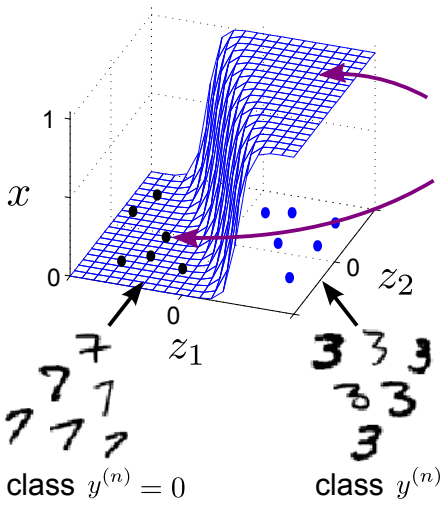


$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Weight space of a single neuron



Training a single neuron



desired result of training:

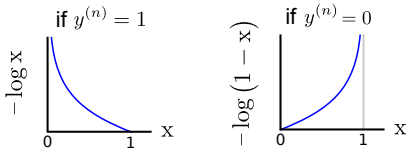
neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $y^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $y^{(n)} = 0$

inputs class labels

training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{y^{(n)}\}_{n=1}^{N}$

consider single term in sum

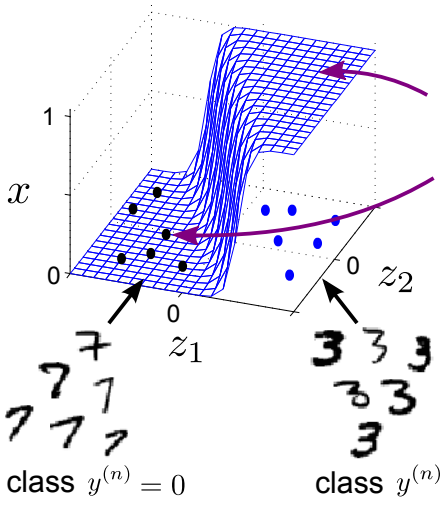


objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

quantifies quality of $\mathrm{x}(\boldsymbol{z}\,; \boldsymbol{w})$ in terms of "fit" to training data

Training a single neuron



desired result of training:

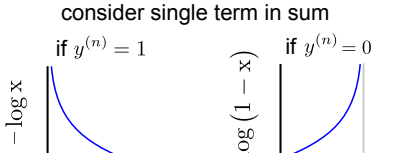
neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $y^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $y^{(n)} = 0$

inputs class labels

training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{y^{(n)}\}_{n=1}^{N}$

consider single term in sum



objective function:

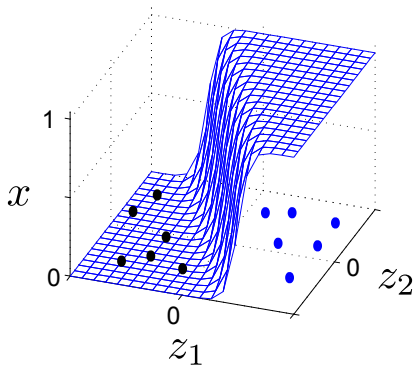
$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

surprise $-\log p(\text{outcome})$ when observing $y^{(n)}$

relative entropy between $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$ and $y^{(n)}$

} small when neuron output 'close' to training data

Training a single neuron: a problem



inputs class labels

training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{y^{(n)}\}_{n=1}^{N}$

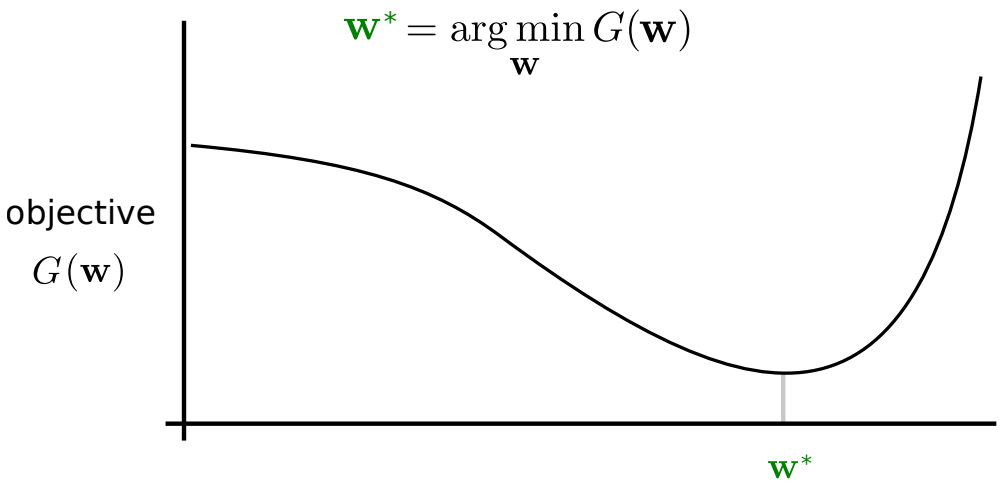
objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

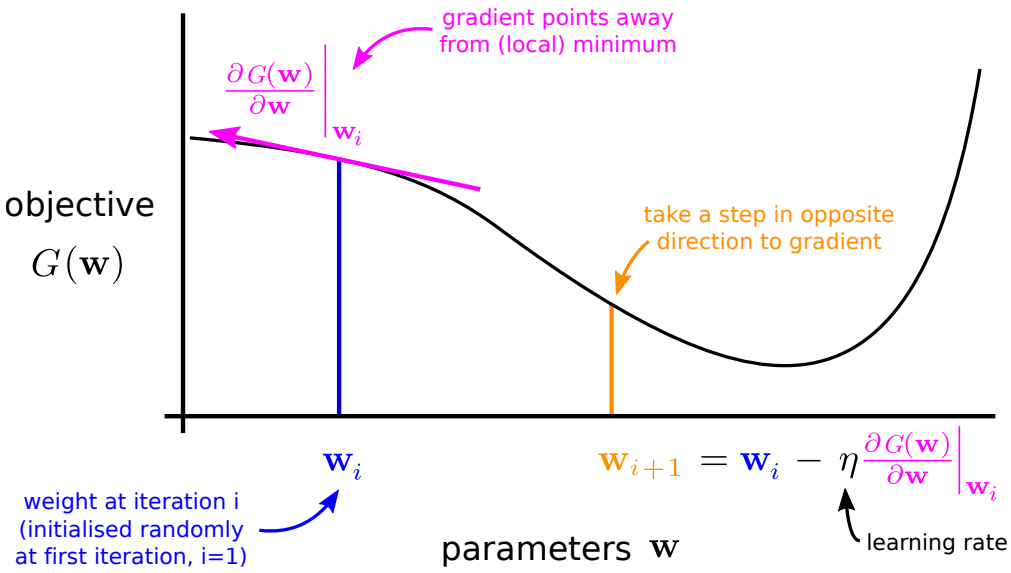
$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$$ choose weights that minimise network's surprise about training data

There is no analytic solution to this optimisation problem: what should we do?

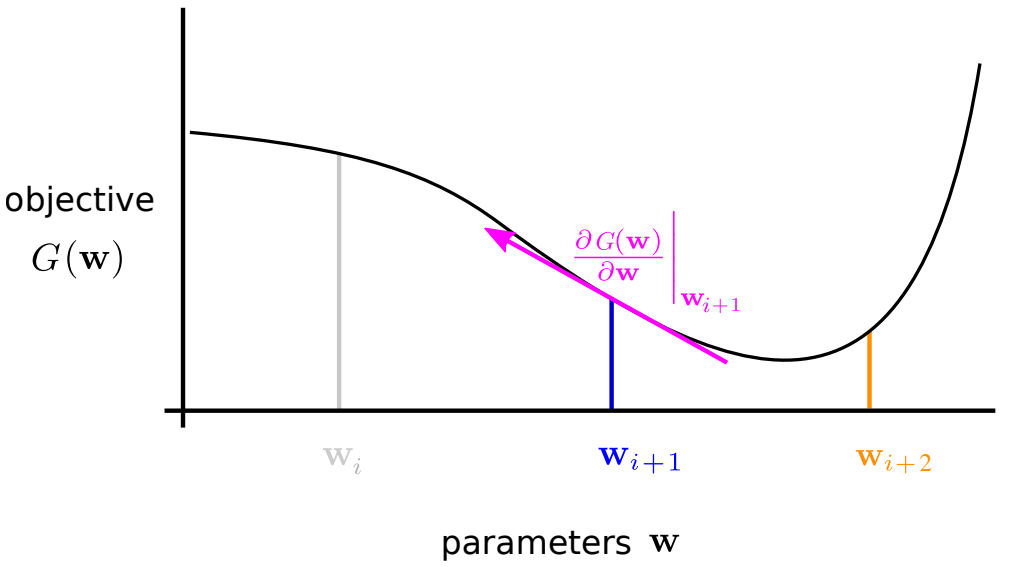
Goal of training



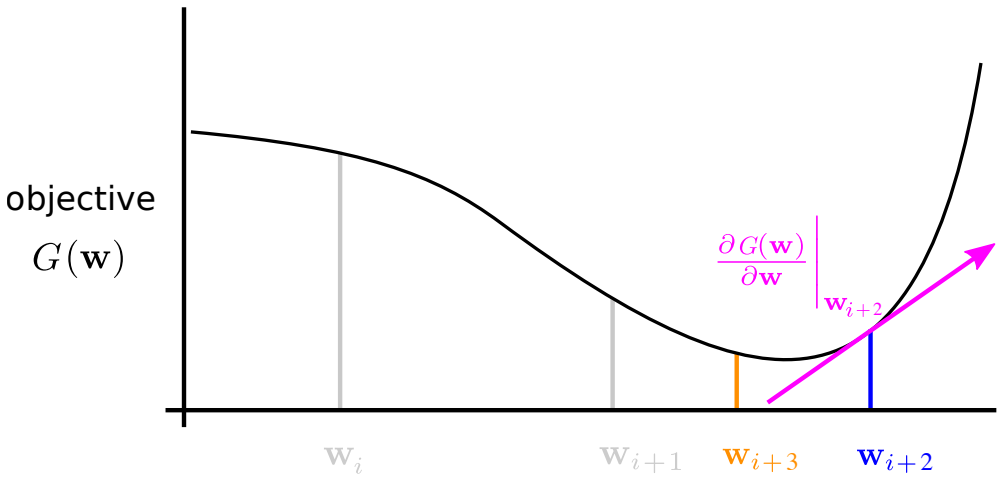
Training using gradient descent



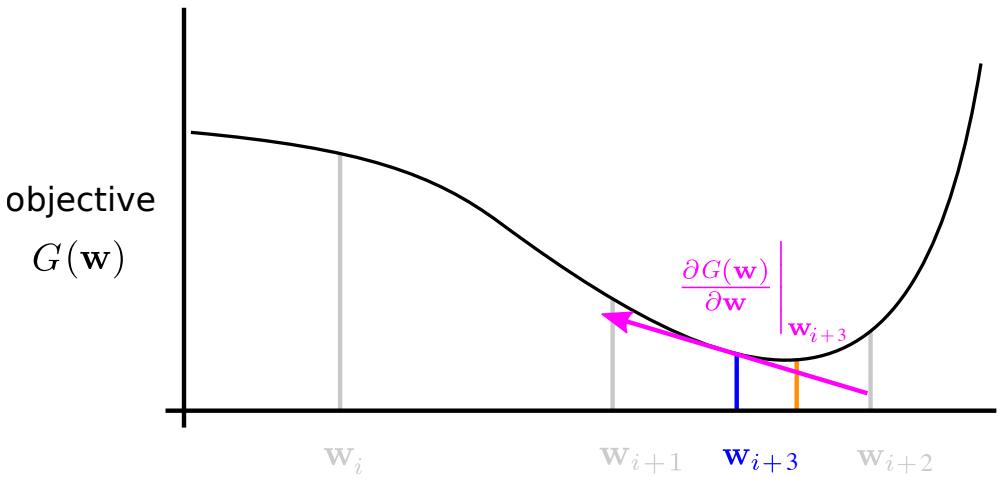
Training using gradient descent



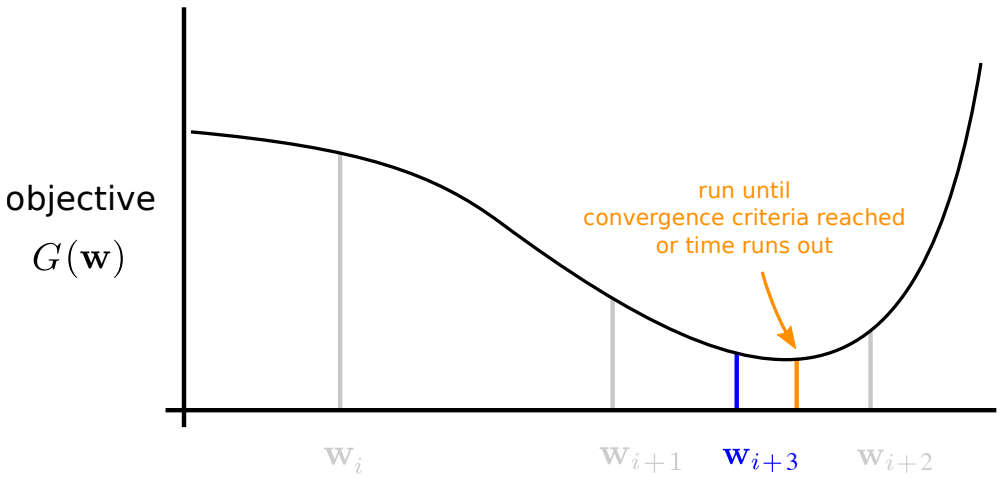
Training using gradient descent



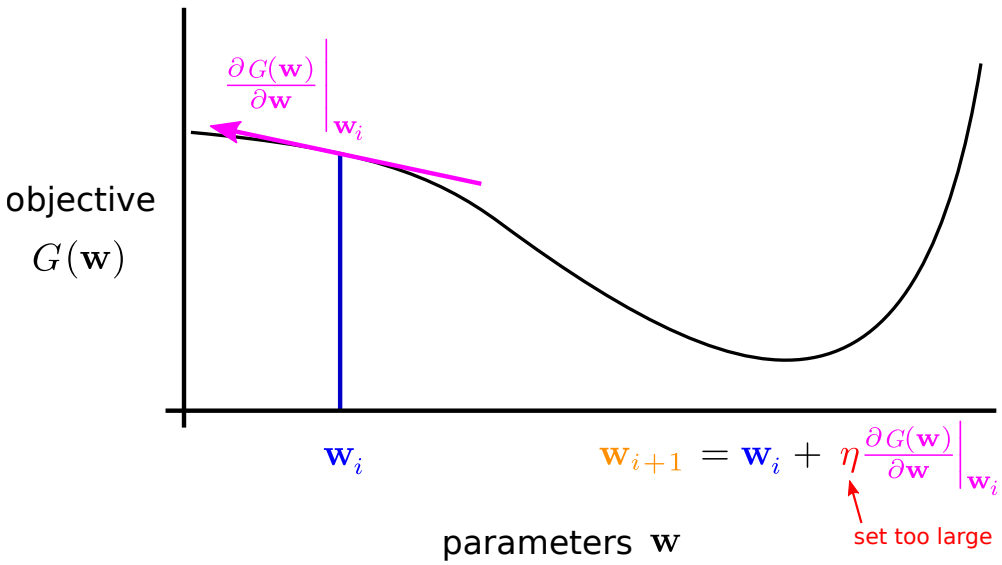
Training using gradient descent



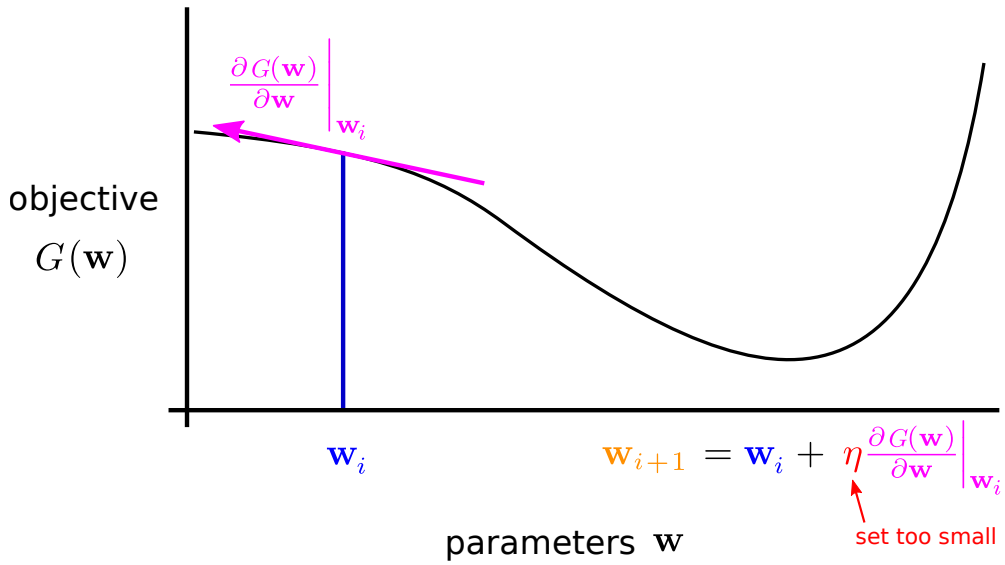
Training using gradient descent



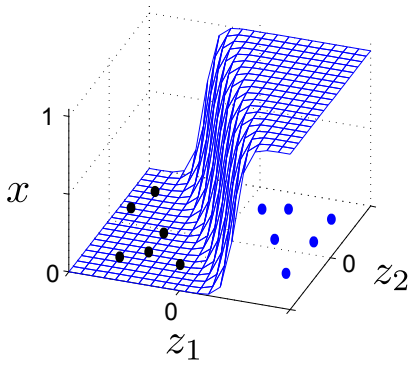
What happens if the learning rate η is set too large?



What happens if the learning rate η is set too small?



Training a single neuron: a solution



training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ (inputs) $\{y^{(n)}\}_{n=1}^{N}$ (class labels)

objective function:

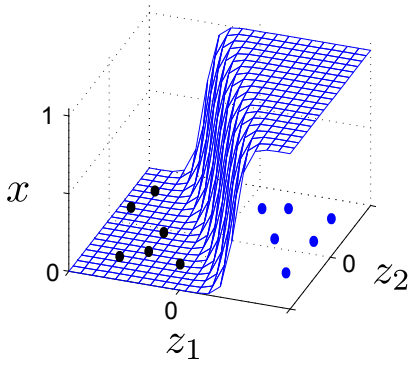
$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$$ choose weights that minimise network's surprise about training data

There is no analytic solution to this optimisation problem: what should we do?

Answer: use gradient descent

Training a single neuron



training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ (inputs) $\{y^{(n)}\}_{n=1}^{N}$ (class labels)

objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$ choose weights that minimise network's surprise about training data

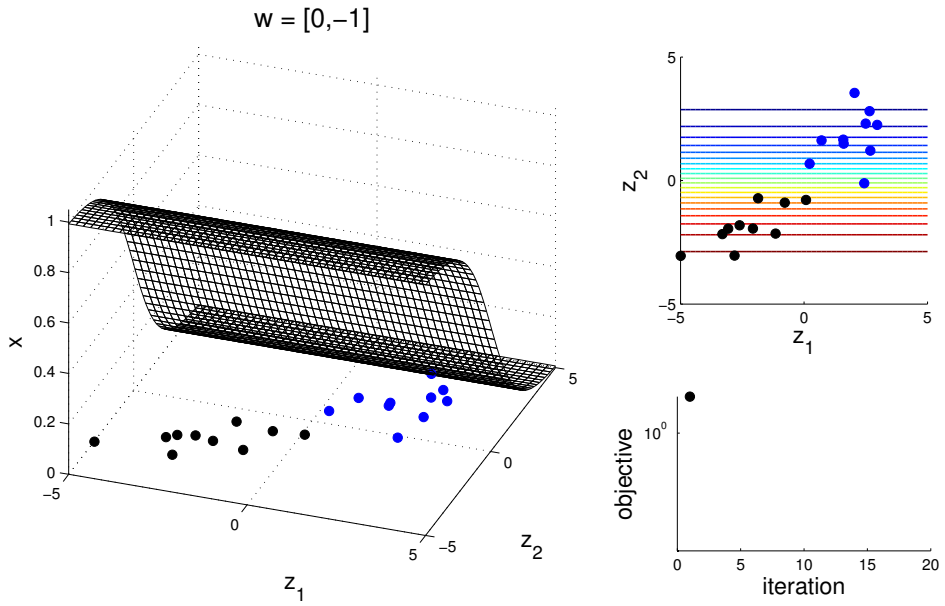
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w}) = \sum_n \frac{\mathrm{d}G(\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}\boldsymbol{w}} = -\sum_n (y^{(n)} - x^{(n)}) \boldsymbol{z}^{(n)}$$

feature

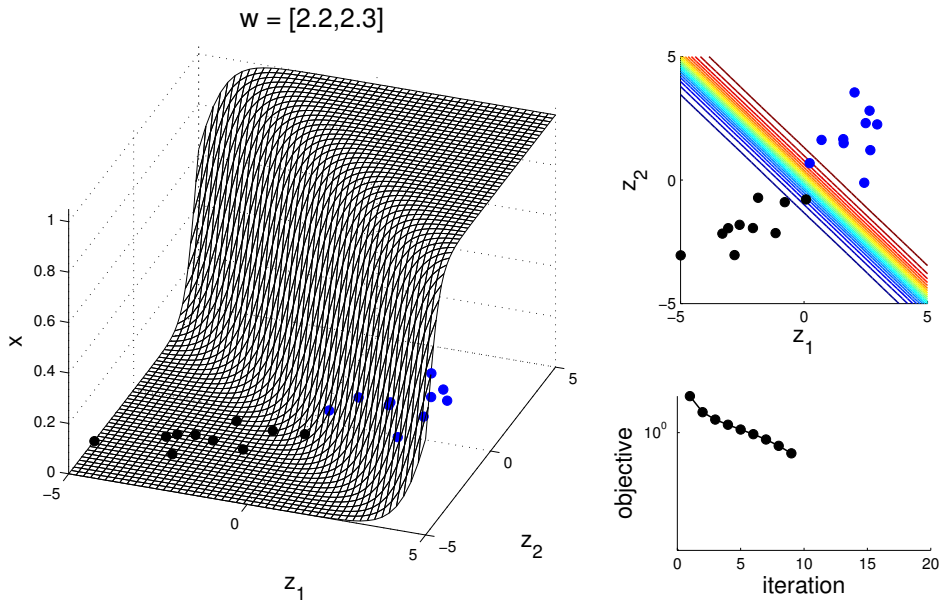
prediction error

$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w})$ iteratively step down the objective (gradient points up hill)

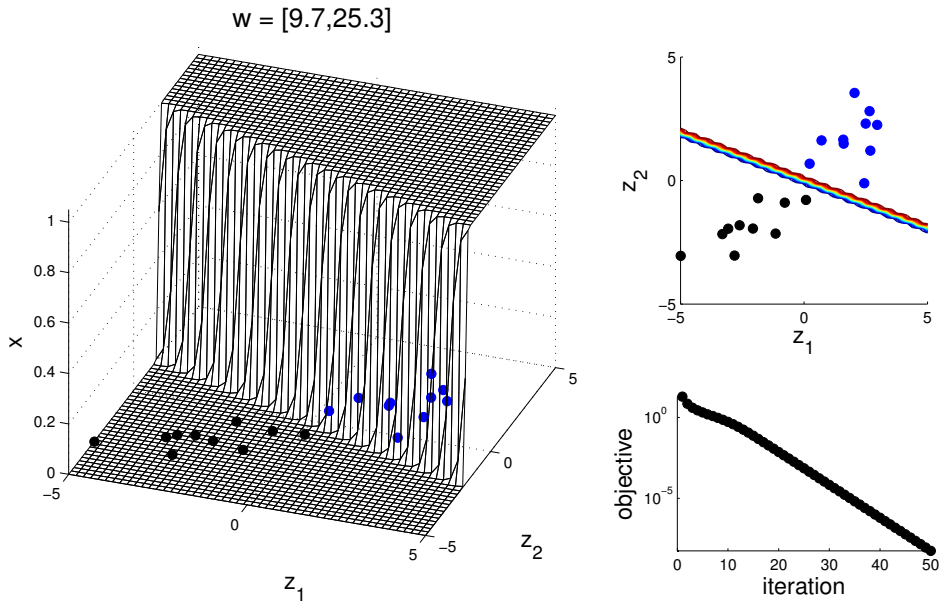
Training a single neuron



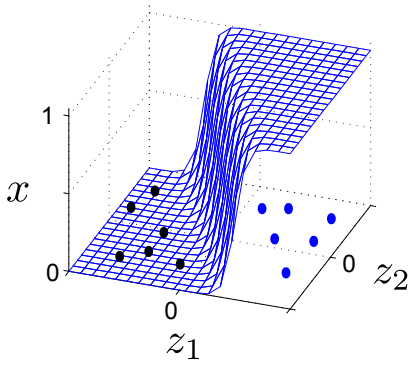
Training a single neuron



Training a single neuron



Over-fitting and weight decay



inputs class labels

training data $\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{y^{(n)}\}_{n=1}^{N}$

objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - y^{(n)}) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right) \right] \geq 0$$

$$E(\boldsymbol{w}) = \tfrac{1}{2} \sum_i w_i^2$$

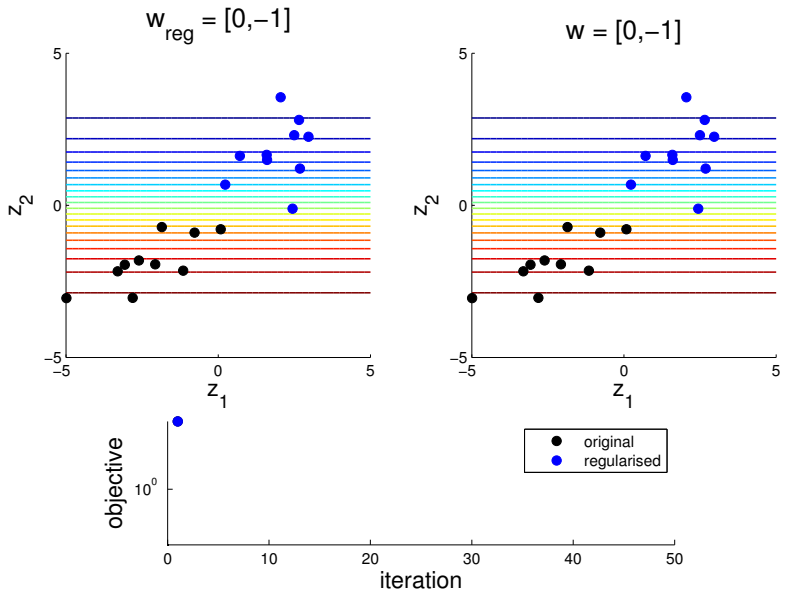
"regulariser": penalise magnitude of the weights and discourages the network from using extreme weights

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} M(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[G(\boldsymbol{w}) + \alpha E(\boldsymbol{w}) \right]$$

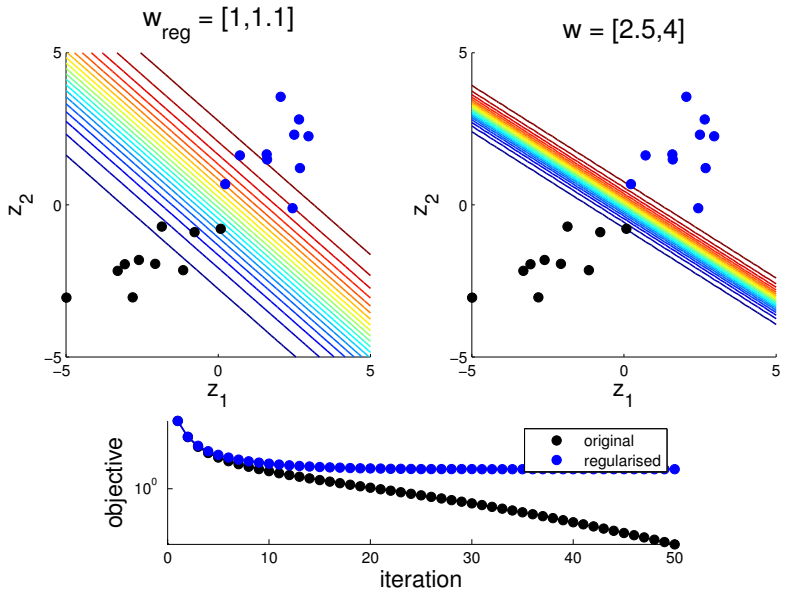
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} M(\boldsymbol{w}) = -\sum_n (y^{(n)} - x^{(n)}) \boldsymbol{z}^{(n)} + \alpha \boldsymbol{w}$$

weight decay - shrinks weights towards zero

Training a single neuron



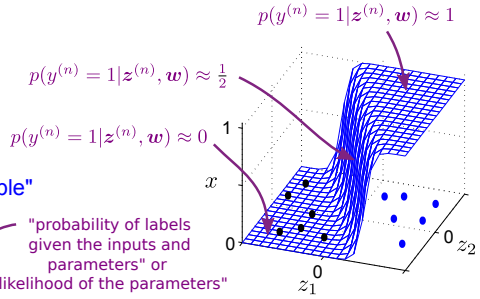
Training a single neuron



Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)}=1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 1: maximum-likelihood learning

"choose setting of w that makes observed data most probable"

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} \; p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N,\boldsymbol{w})$$

"probability of labels given the inputs and parameters" or "likelihood of the parameters"

$$p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N,\boldsymbol{w}) = \prod_{n=1}^N p(y^{(n)}|\boldsymbol{z}^{(n)},\boldsymbol{w})$$

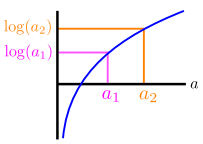
$$p(y^{(n)}|\boldsymbol{z}^{(n)},\boldsymbol{w}) = \begin{cases} \mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) & \text{if } y^{(n)}=1 \\ 1-\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) & \text{if } y^{(n)}=0 \end{cases}$$

$$= \prod_{n=1}^N \mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w})^{y^{(n)}}(1-\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}))^{(1-y^{(n)})}$$

computationally and mathematically simpler to work with log-likelihoods

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} \; \log p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N,\boldsymbol{w})$$

monotonicity of logarithm preserves location of optimum

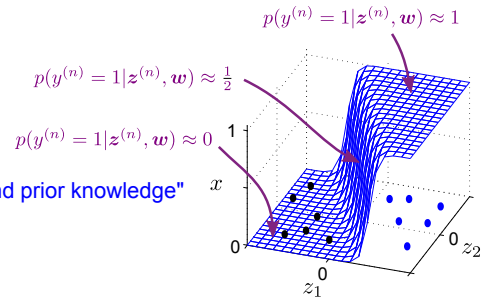


$$\log p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N,\boldsymbol{w}) = \sum_{n=1}^N y^{(n)}\log \mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) + (1-y^{(n)})\log(1-\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w})) = -G(\boldsymbol{w})$$

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)}=1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A=\boldsymbol{w}$, $B=\{y^{(n)}\}_{n=1}^N$ and $C=\{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

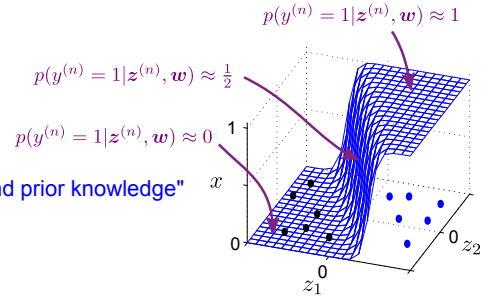
$$= \arg\max_{\boldsymbol{w}} \left[\log p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) + \log p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w}) - \log p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)\right]$$

does not depend on w

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)}=1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A=\boldsymbol{w}$, $B=\{y^{(n)}\}_{n=1}^N$ and $C=\{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

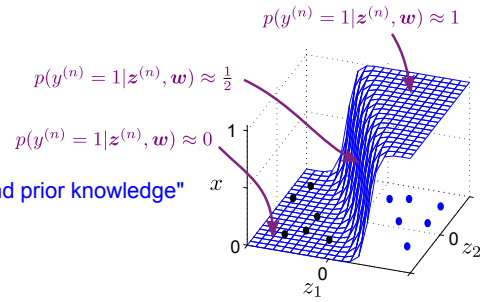
$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

$$= \arg\max_{\boldsymbol{w}} \left[\log p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) + \log p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})\right] = -G(\boldsymbol{w})$$

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)}=1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A=\boldsymbol{w}$, $B=\{y^{(n)}\}_{n=1}^N$ and $C=\{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

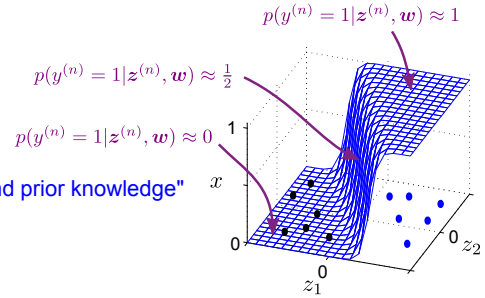
substitute in log prior

$$= \arg\max_{\boldsymbol{w}} \left[\log p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) - G(\boldsymbol{w})\right]$$

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)}=1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A=\boldsymbol{w}$, $B=\{y^{(n)}\}_{n=1}^N$ and $C=\{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

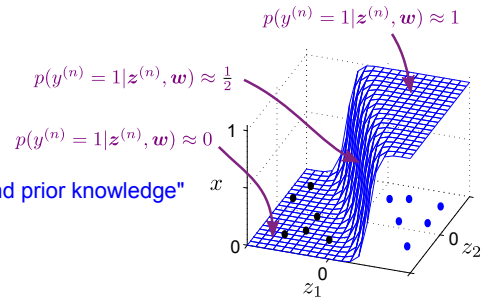
does not depend on w

$$= \arg\max_{\boldsymbol{w}} \left[-\frac{D}{2}\log(2\pi\sigma_w^2) - \frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2 \;-\; G(\boldsymbol{w})\right]$$

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)} = 1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A = \boldsymbol{w}$, $B = \{y^{(n)}\}_{n=1}^N$ and $C = \{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

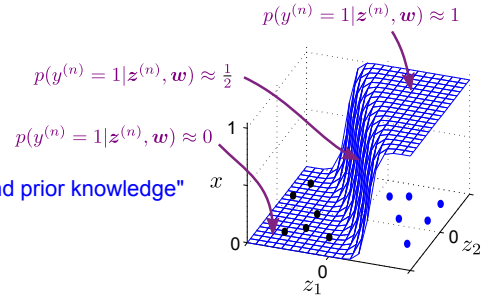
$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

$$= \arg\max_{\boldsymbol{w}} \left[-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2 \; - \; G(\boldsymbol{w}) \right] = \alpha E(\boldsymbol{w})$$

Probabilistic interpretation of the single neuron

Intrepret the output of the network as a probability:

$$\mathrm{x}(\boldsymbol{z}^{(n)};\boldsymbol{w}) = p(y^{(n)} = 1|\boldsymbol{z}^{(n)},\boldsymbol{w})$$



Fitting method 2: maximum a posteriori (MAP)

"choose setting of w that is most probable given the data and prior knowledge"

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

Bayes' rule $p(A|B,C) = \frac{1}{p(B|C)}\, p(A|C)p(B|A,C)$ with $A = \boldsymbol{w}$, $B = \{y^{(n)}\}_{n=1}^N$ and $C = \{\boldsymbol{z}^{(n)}\}_{n=1}^N$

$$p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N) = \frac{1}{p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N)}\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) p(\{y^{(n)}\}_{n=1}^N|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$$

posterior distribution (what we know after seeing data) — normalising constant — prior distribution (what we knew before seeing data) — likelihood of the parameters (what the data tell us)

use a zero mean, independent Gaussian prior $p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N) = \frac{1}{(2\pi\sigma_w^2)^{D/2}} \exp(-\frac{1}{2\sigma_w^2}\sum_{d=1}^D w_d^2)$

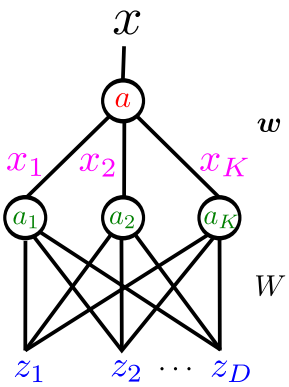
$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log\, p(\boldsymbol{w}|\{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$$

$$= \arg\max_{\boldsymbol{w}} \left[-\alpha E(\boldsymbol{w}) \; - \; G(\boldsymbol{w}) \right] \quad \text{where} \quad \alpha = \frac{1}{\sigma_w^2}$$

Machine learning: rebranding statistics?

Machine Learning terminology	Statistical terminology
single neuron for classification	logistic regression or logistic classification
$\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$	$p(y^{(n)} = 1 \vert \boldsymbol{z}^{(n)}, \boldsymbol{w})$
relative entropy (data fit term)	log-likelihood function
$G(\boldsymbol{w})$	$\log p(\{y^{(n)}\}_{n=1}^N \vert \{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$
regularisation (weight decay term)	regularisation (log-prior)
$\alpha E(\boldsymbol{w})$	$\log p(\boldsymbol{w} \vert \{\boldsymbol{z}^{(n)}\}_{n=1}^N)$
minimise of relative entropy	maximum likelihood learning
$\arg\min_{\boldsymbol{w}} \; G(\boldsymbol{w})$	$\arg\max_{\boldsymbol{w}} \; \log p(\{y^{(n)}\}_{n=1}^N \vert \{\boldsymbol{z}^{(n)}\}_{n=1}^N, \boldsymbol{w})$
minimise relative entropy + regulariser	maximum a posteriori fitting
$\arg\min_{\boldsymbol{w}} [G(\boldsymbol{w}) + \alpha E(\boldsymbol{w})]$	$\arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \vert \{\boldsymbol{z}^{(n)}\}_{n=1}^N, \{y^{(n)}\}_{n=1}^N)$

Single hidden layer neural networks

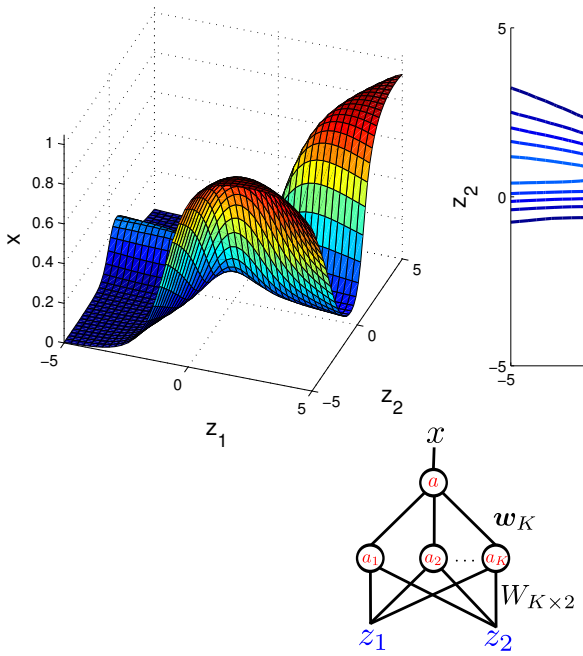


output of network	$x(a) = \frac{1}{1+\exp(-a)}$
activity of output neuron	$a = \sum_{k=1}^K w_k x_k$
hidden neuron outputs	$x(a_k) = \frac{1}{1+\exp(-a_k)} = x_k$
hidden neuron activities	$a_k = \sum_{d=1}^D W_{k,d} z_d$
inputs	

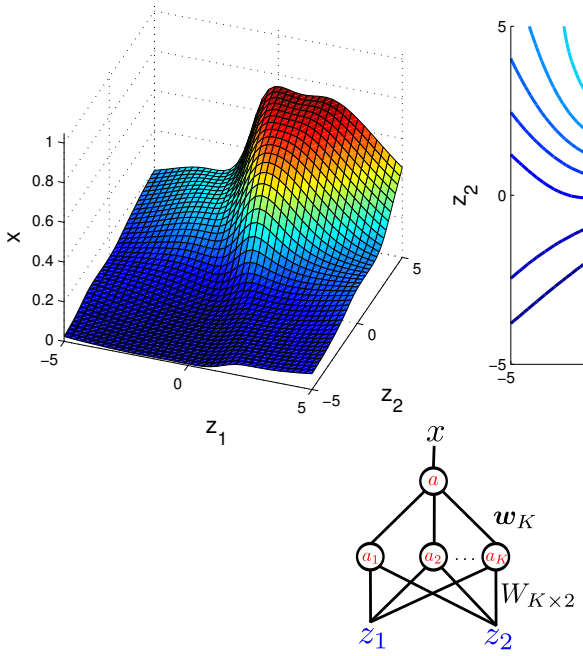
output weights
dimensionality($\boldsymbol{w}$) = K

input weights
dimensionality(W) = $K \times D$

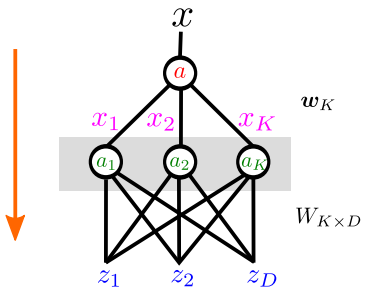
Sampling random neural network classifiers



Sampling random neural network classifiers



Backpropagation



output of network $x(a) = \frac{1}{1+\exp(-a)}$

activity of output neuron $a = \sum_{k=1}^{K} w_k x_k$

hidden neuron outputs $x(a_k) = \frac{1}{1+\exp(-a_k)} = x_k$

hidden neuron activities $a_k = \sum_{d=1}^{D} W_{k,d} z_d$

inputs

objective function:

$$x^{(n)} = \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$$

$$G(\boldsymbol{w}) = -\sum_n \left[y^{(n)} \log x^{(n)} + (1 - y^{(n)}) \log \left(1 - x^{(n)}\right) \right]$$ likelihood same as before

$$E(W, \boldsymbol{w}) = \tfrac{1}{2} \sum_i w_i^2 + \tfrac{1}{2} \sum_{ij} W_{ij}^2$$ regulariser discourages extreme weights

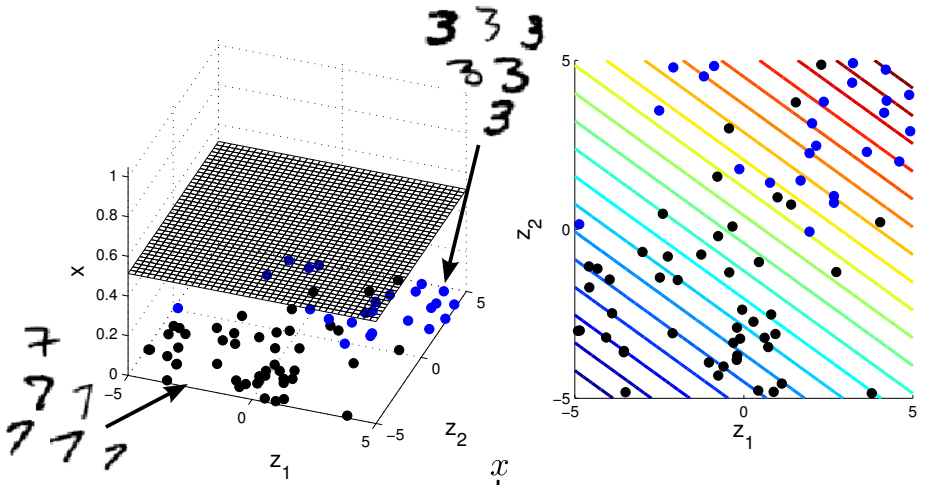
$$\{W, \boldsymbol{w}^*\} = \arg\min_{W,\boldsymbol{w}} M(W, \boldsymbol{w}) = \arg\min_{W,\boldsymbol{w}} \left[G(W, \boldsymbol{w}) + \alpha E(W, \boldsymbol{w}) \right]$$

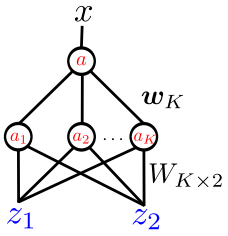
chain rule = backward pass through network = "backpropagation"

$$\frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}W_{ij}}$$

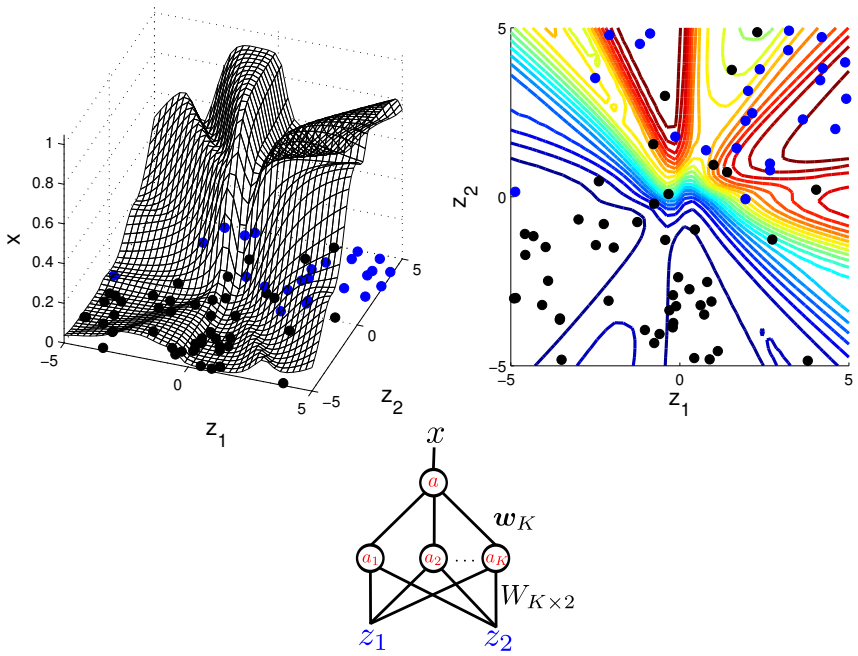
$$= \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}x_i^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}x_i^{(n)}}{\mathrm{d}a_i^{(n)}} \frac{\mathrm{d}a_i^{(n)}}{\mathrm{d}W_{ij}}$$

Training a neural network with a single hidden layer

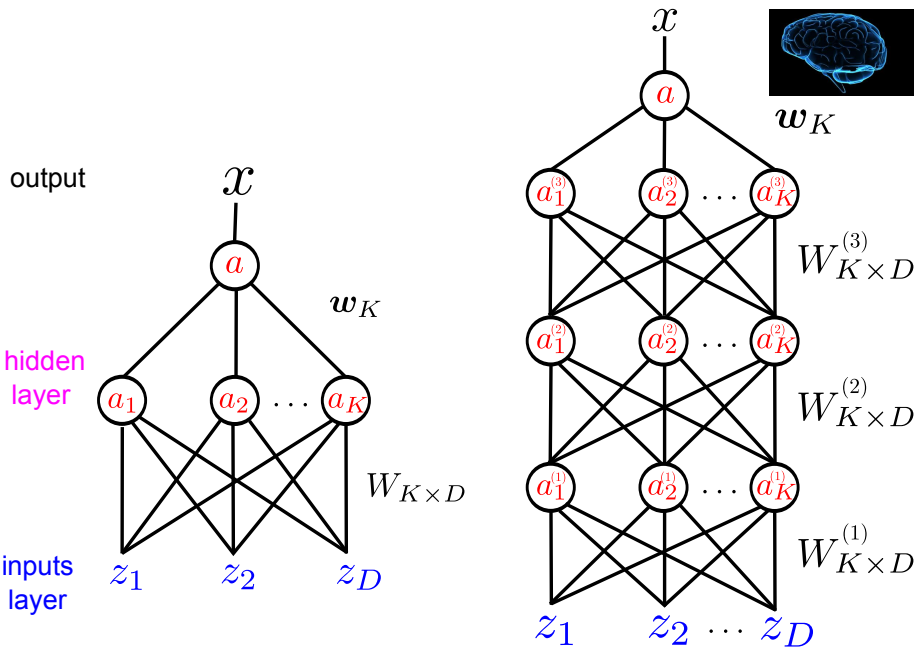




Training a neural network with a single hidden layer



Multi-layer perceptrons: hierarchical models with many hidden layers



A Potted History of Neural Networks

1980s First wave of neural network development

- multi-layer perceptrons, convolutional neural networks, gradient based optimisation etc.
- limited training success caused progress to stall.

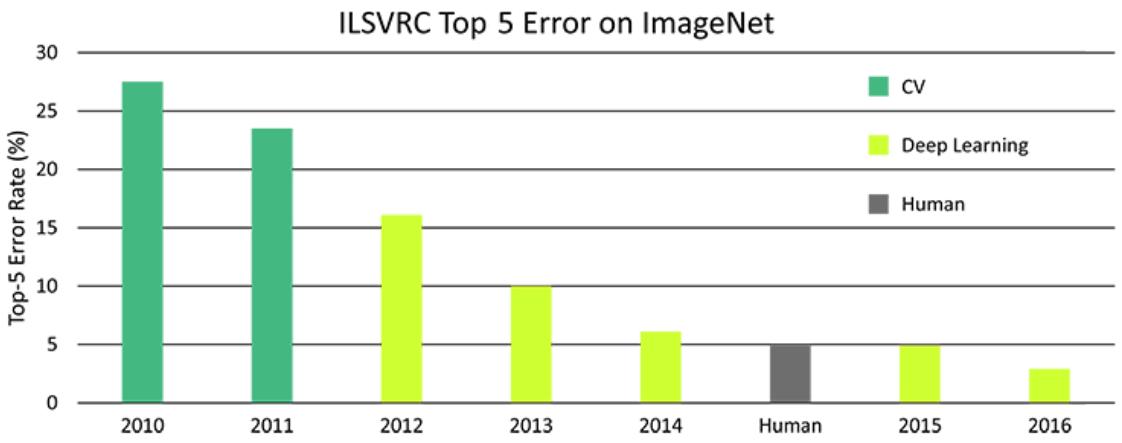
2010s Second wave of neural network development due to

1. more data (e.g. imageNet)
2. more compute (e.g. GPUs)
3. technical advances (e.g. initialisation, neuron non-linearities, model architecture)

2019 Bespoke software for designing & training neural networks

- automatic differentiation, simple deployment to GPUs etc.
- examples include PyTorch and TensorFlow
- software 2.0: neural network design = curating a programme with each layer being a step, now learn precise form from data

Progression of Performance on Object Reconition



Summary of Part I

- **Supervised artificial neural networks** fit a non-linear function x that maps from input features (z) to output targets (y), like a fancy form of curve fitting.
- Networks with several hidden layers are known as **multi-layer perceptrons** and can be fit by optimising an **objective function** by **gradient descent**.
- The objective function comprises a **data fit term** $G(\mathbf{w})$ that ensures the output of the network matches the training data and a **regularisation term** $E(\mathbf{w})$ to stop **over-fitting**. These terms can be related to the **likelihood function** and **prior** of the weights from the field of statistics.
- The gradient computation requires the chain rule and is known as **backpropagation**.
- **Smart initialisation** is required to stop learning falling into poor **local optima**.

Demo: `http://yann.lecun.com/exdb/lenet/`