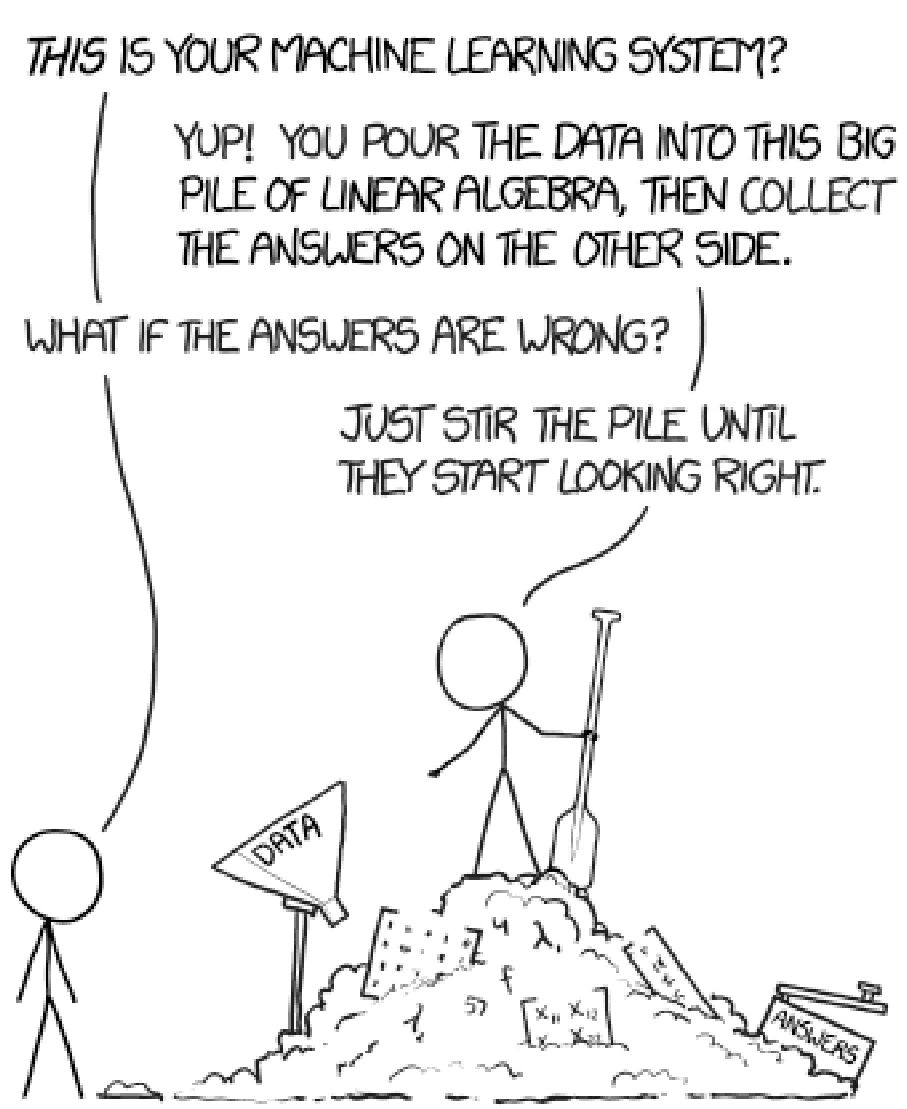
Deep Learning (CS324)

2. Mathematical background*

Prof. Jianguo Zhang
SUSTech

*Source: http://www.deeplearningbook.org/lecture_slides.html (2-4) + a bunch of xkcd.com comics

Linear algebra


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Scalars and vectors

- A scalar is a single number
- Integers, real numbers, rational numbers, etc.
- We denote it with italic font: a, n, x
- A vector is a 1-D array of numbers:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrices and tensors

- A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a 2-D array of numbers

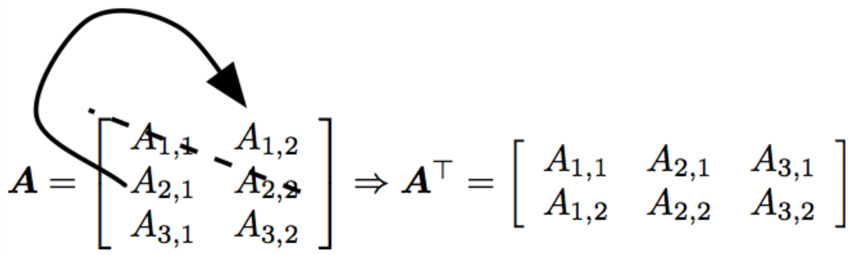
$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

- A tensor is an array of numbers, that may have
  - zero dimensions, and be a scalar
  - one dimension, and be a vector
  - two dimensions, and be a matrix
  - or more dimensions.

Matrix transpose

$$(\boldsymbol{A}^\top)_{i,j} = A_{j,i}.$$

$$(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top$$



$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \boldsymbol{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Figure 2.1: The transpose of the matrix can be thought of as a mirror image across the main diagonal.

Matrix addition and subtraction

- We can add or subtract two matrices **A** and **B** only if these have the same size, and the result (for addition) is a matrix **C** = **A** + **B** with elements $c_{ij} = a_{ij} + b_{ij}$ for each $1 \leq i \leq m$ and $1 \leq j \leq n$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} -1 & 2 \\ 4 & -8 \\ -16 & 32 \end{bmatrix} = \begin{bmatrix} 0 & 4 \\ 7 & -4 \\ -11 & 38 \end{bmatrix} = \mathbf{C}$$

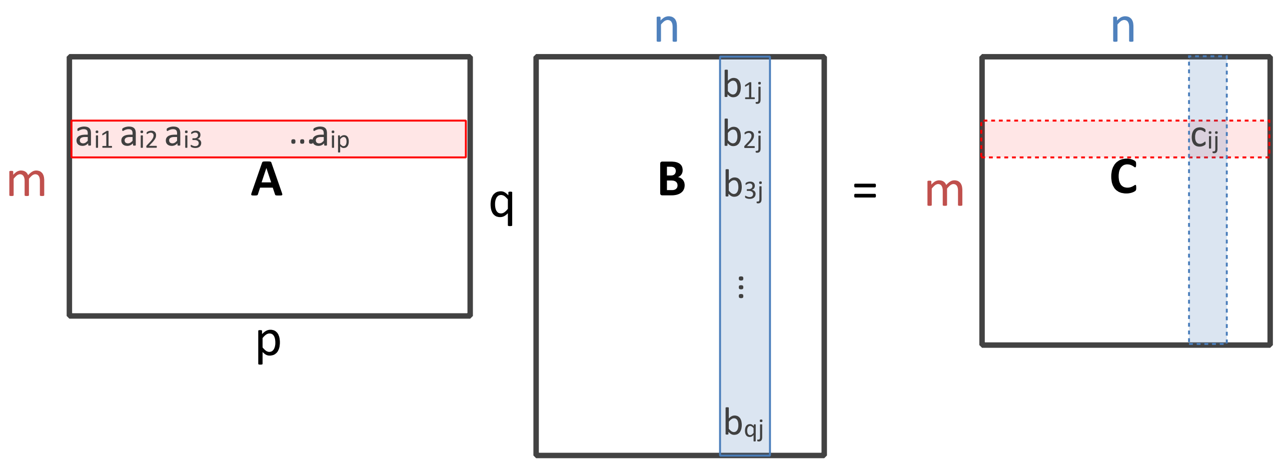
Scalar multiplication

- The multiplication of a matrix by a scalar *t* produces as a result a matrix of the same size whose entries are each multiplied by *t*

$$\mathbf{B} = t\mathbf{A} = t\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 1t & 2t \\ 3t & 3t \\ 5t & 6t \end{bmatrix} = \mathbf{C}$$

Matrix multiplication

- Let **A** be a *m* by *p* matrix and **B** a *q* by *n* matrix: the product **AB** is defined only when *p* = *q*



- The elements of **C** are

$$c_{ij} = \sum_{k=1}^{p} a_{ik}b_{kj} = a_{i1}b_{ij} + a_{i2}b_{2j} + \cdots + a_{ip}b_{pj}$$

Identity matrix

- The identity matrix is the diagonal matrix with ones on the diagonal and zero everywhere else

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 2.2: *Example identity matrix*: This is $\boldsymbol{I}_3$.

- If **A** is *m* x *n* then

$$\mathbf{A}\mathbf{I}_n = \mathbf{A} \qquad \mathbf{I}_m\mathbf{A} = \mathbf{A}$$

Trace of a matrix

- The trace of a matrix is equal to then sum of its diagonal elements

$$\mathrm{Tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i}$$

$$\mathrm{Tr}(\boldsymbol{ABC}) = \mathrm{Tr}(\boldsymbol{CAB}) = \mathrm{Tr}(\boldsymbol{BCA})$$

Linear systems of equations

- A linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can have:
  - No solution
  - Many solutions
  - Exactly one solution: if **A** is invertible

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$$

Linear systems of equations

- A linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can have:
  - No solution
  - Many solutions
  - Exactly one solution: if **A** is invertible

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

$$\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

Linear systems of equations

- A linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ can have:
  - No solution
  - Many solutions
  - Exactly one solution: if **A** is invertible
  - Matrix can't be inverted if…
    - More rows/columns than columns/rows
    - Redundant rows/columns ("linearly dependent", "low rank")

Norms

- Functions that measure how "large" a vector is
- Similar to a distance between zero and the point represented by the vector

- $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \boldsymbol{0}$
- $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (the *triangle inequality*)
- $\forall \alpha \in \mathbb{R}, f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$

Norms

- *L_p* norms

$$||\boldsymbol{x}||_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

- Most popular norm: *L_2* norm
- *L_1* norm:

$$||\boldsymbol{x}||_1 = \sum_i |x_i|$$

- Max norm, infinite *p:*

$$||\boldsymbol{x}||_\infty = \max_i |x_i|$$

Special vectors and matrices

- Unit vector:

$$||\boldsymbol{x}||_2 = 1$$

- Symmetric Matrix:

$$\boldsymbol{A} = \boldsymbol{A}^\top$$

- Orthogonal matrix:

$$\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{I}$$

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^\top$$

Matrix eigendecomposition

- Eigenvector and eigenvalue of a square matrix **A**

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$$

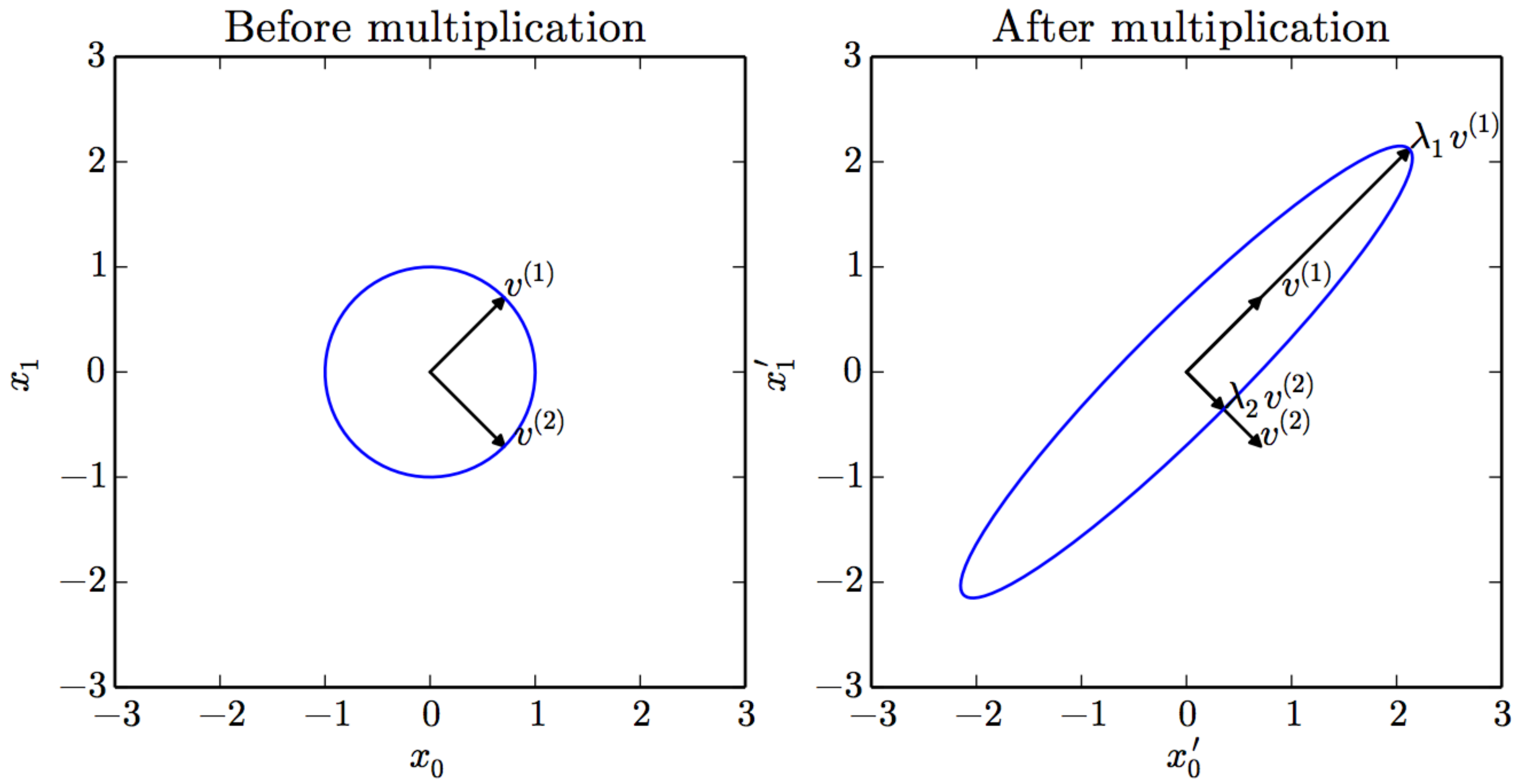
- Eigendecomposition of a diagonalisable matrix

$$\boldsymbol{A} = \boldsymbol{V} \operatorname{diag}(\boldsymbol{\lambda}) \boldsymbol{V}^{-1}$$

- Every real symmetric matrix has a real, orthogonal eigendecomposition

$$\boldsymbol{A} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^{\top}$$

Eigenvalues interpretation



Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}$$

m x *m* orthogonal matrix (left-singular vectors - eigenvectors of $\mathbf{AA}^{\top}$)

- More general; matrix need not be square

Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

n x *n* orthogonal matrix (right-singular vectors - eigenvectors of $\mathbf{A}^\top\mathbf{A}$)

- More general; matrix need not be square

Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

m x *n* diagonal matrix (singular values - square roots of eigenvalues of $\mathbf{A}^\top\mathbf{A}$)

- More general; matrix need not be square

Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

- More general; matrix need not be square
- Allows to compute Moore-Penrose pseudoinverse

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{D}^+\boldsymbol{U}^\top$$

Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

- More general; matrix need not be square
- Allows to compute Moore-Penrose pseudoinverse

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{D}^+\boldsymbol{U}^\top$$

Reciprocal of nonzero elements of **D** and then transpose

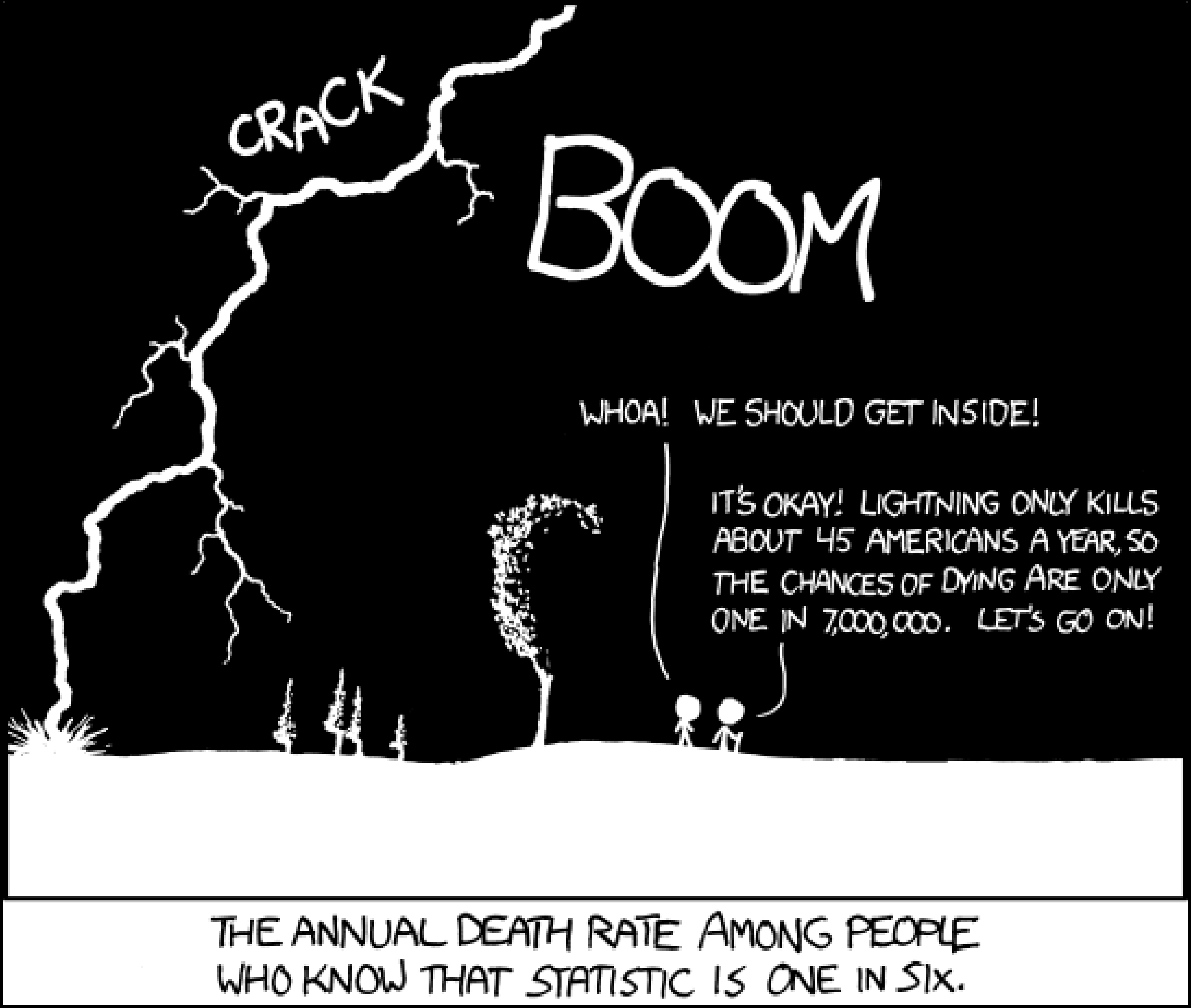
Singular value decomposition

- What if the matrix **A** is not square?

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$$

- More general; matrix need not be square
- Allows to compute Moore-Penrose pseudoinverse
- If **A** has more rows than cols $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{y}$ gives solution with min $||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2$.

Probability & information theory


CRACK
BOOM
WHOA! WE SHOULD GET INSIDE!
IT'S OKAY! LIGHTNING ONLY KILLS ABOUT 45 AMERICANS A YEAR, SO THE CHANCES OF DYING ARE ONLY ONE IN 7,000,000. LET'S GO ON!
THE ANNUAL DEATH RATE AMONG PEOPLE WHO KNOW THAT STATISTIC IS ONE IN SIX.

Probability mass function

- The domain of P must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$. An impossible event has probability 0 and no state can be less probable than that. Likewise, an event that is guaranteed to happen has probability 1, and no state can have a greater chance of occurring.
- $\sum_{x \in \mathrm{x}} P(x) = 1$. We refer to this property as being **normalized**. Without this property, we could obtain probabilities greater than one by computing the probability of one of many events occurring.

$$P(\mathrm{x} = x_i) = \frac{1}{k}$$

Example: uniform distribution

Probability density function

- The domain of p must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.
- $\int p(x)dx = 1$.

$$u(x; a, b) = \frac{1}{b-a}.$$

Example: uniform distribution

A few important rules…

- Marginal probabilities (discrete and continuous)

$$\forall x \in \mathrm{x}, P(\mathrm{x}=x) = \sum_y P(\mathrm{x}=x, \mathrm{y}=y) \qquad p(x) = \int p(x,y)dy$$

- Conditional probability

$$P(\mathrm{y}=y \mid \mathrm{x}=x) = \frac{P(\mathrm{y}=y, \mathrm{x}=x)}{P(\mathrm{x}=x)}$$

- Chain rule

$$P(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(n)}) = P(\mathrm{x}^{(1)})\Pi_{i=2}^{n} P(\mathrm{x}^{(i)} \mid \mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(i-1)})$$

A few important rules…

- Independence

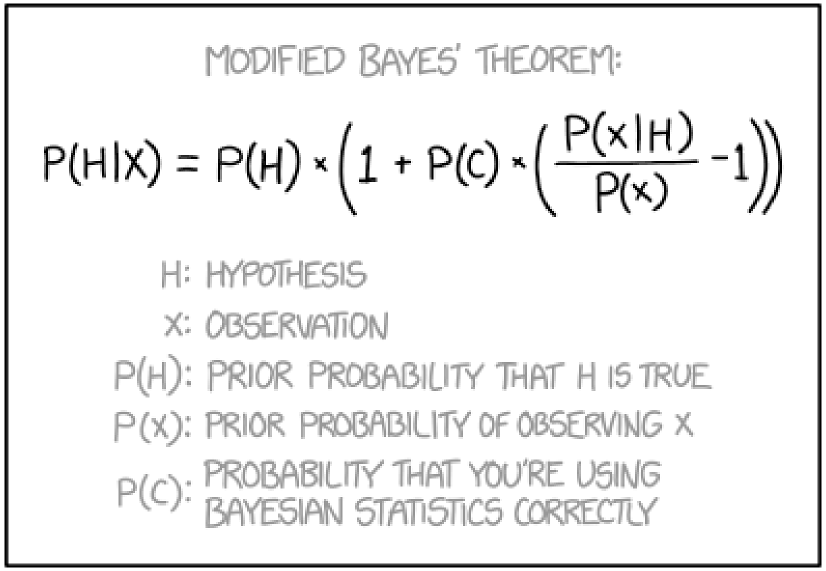
$$\forall x \in \mathrm{x}, y \in \mathrm{y},\ p(\mathrm{x}=x, \mathrm{y}=y) = p(\mathrm{x}=x)p(\mathrm{y}=y)$$

- Conditional independence

$$\forall x \in \mathrm{x}, y \in \mathrm{y}, z \in \mathrm{z},\ p(\mathrm{x}=x, \mathrm{y}=y \mid \mathrm{z}=z) = p(\mathrm{x}=x \mid \mathrm{z}=z)p(\mathrm{y}=y \mid \mathrm{z}=z)$$

Bayes' Rule

$$P(\mathrm{x} \mid \mathrm{y}) = \frac{P(\mathrm{x})P(\mathrm{y} \mid \mathrm{x})}{P(\mathrm{y})}$$



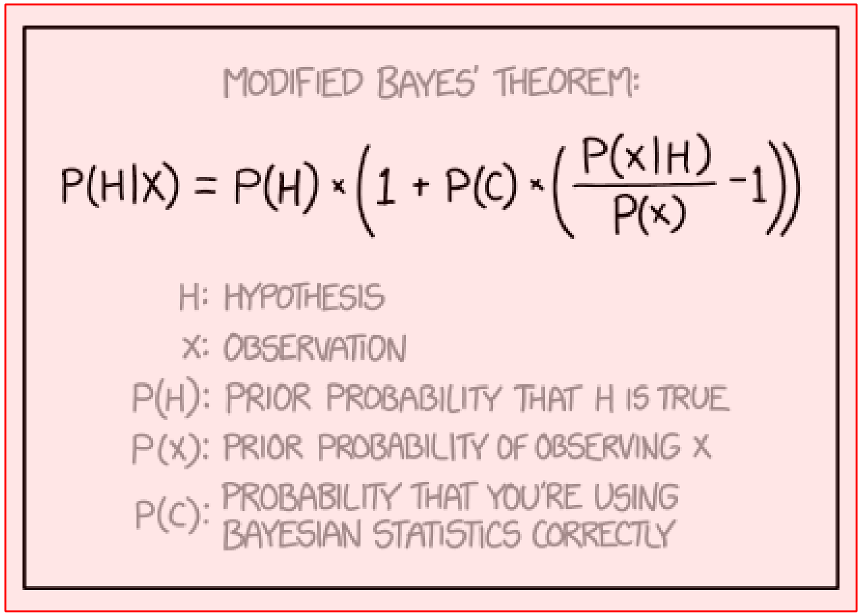
MODIFIED BAYES' THEOREM:

$$P(H|X) = P(H) \times \left(1 + P(C) \times \left(\frac{P(x|H)}{P(x)} - 1\right)\right)$$

H:	HYPOTHESIS
X:	OBSERVATION
P(H):	PRIOR PROBABILITY THAT H IS TRUE
P(X):	PRIOR PROBABILITY OF OBSERVING X
P(C):	PROBABILITY THAT YOU'RE USING BAYESIAN STATISTICS CORRECTLY

Bayes' Rule

$$P(\mathrm{x} \mid \mathrm{y}) = \frac{P(\mathrm{x})P(\mathrm{y} \mid \mathrm{x})}{P(\mathrm{y})}$$



MODIFIED BAYES' THEOREM:

$$P(H|X) = P(H) \times \left(1 + P(C) \times \left(\frac{P(x|H)}{P(x)} - 1\right)\right)$$

H: HYPOTHESIS

X: OBSERVATION

P(H): PRIOR PROBABILITY THAT H IS TRUE

P(X): PRIOR PROBABILITY OF OBSERVING X

P(C): PROBABILITY THAT YOU'RE USING BAYESIAN STATISTICS CORRECTLY

NB: I hope there's no need to point this out but this is just a joke…

Expectation

- of a discrete random variable

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \sum_x P(x)f(x)$$

- of a continuous random variable

$$\mathbb{E}_{\mathrm{x}\sim p}[f(x)] = \int p(x)f(x)dx$$

- Linearity of expectations

$$\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha\mathbb{E}_{\mathrm{x}}[f(x)] + \beta\mathbb{E}_{\mathrm{x}}[g(x)]$$

Variance and covariance

- Variance

$$\mathrm{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

- Covariance

$$\mathrm{Cov}(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}\left[f(x)\right])\left(g(y) - \mathbb{E}\left[g(y)\right]\right)\right]$$

- Covariance matrix given a random vector **x**

$$\mathrm{Cov}(\mathbf{x})_{i,j} = \mathrm{Cov}(\mathrm{x}_i, \mathrm{x}_j)$$

Bernoulli distribution

$$P(\mathrm{x} = 1) = \phi$$

$$P(\mathrm{x} = 0) = 1 - \phi$$

$$P(\mathrm{x} = x) = \phi^x (1 - \phi)^{1-x}$$

$$\mathbb{E}_{\mathrm{x}}[\mathrm{x}] = \phi$$

$$\mathrm{Var}_{\mathrm{x}}(\mathrm{x}) = \phi(1 - \phi)$$

Gaussian distribution

- Parametrised by variance

$$\mathcal{N}(x;\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

- Parametrised by precision

$$\mathcal{N}(x;\mu,\beta^{-1}) = \sqrt{\frac{\beta}{2\pi}}\exp\left(-\frac{1}{2}\beta(x-\mu)^2\right)$$

Gaussian distribution

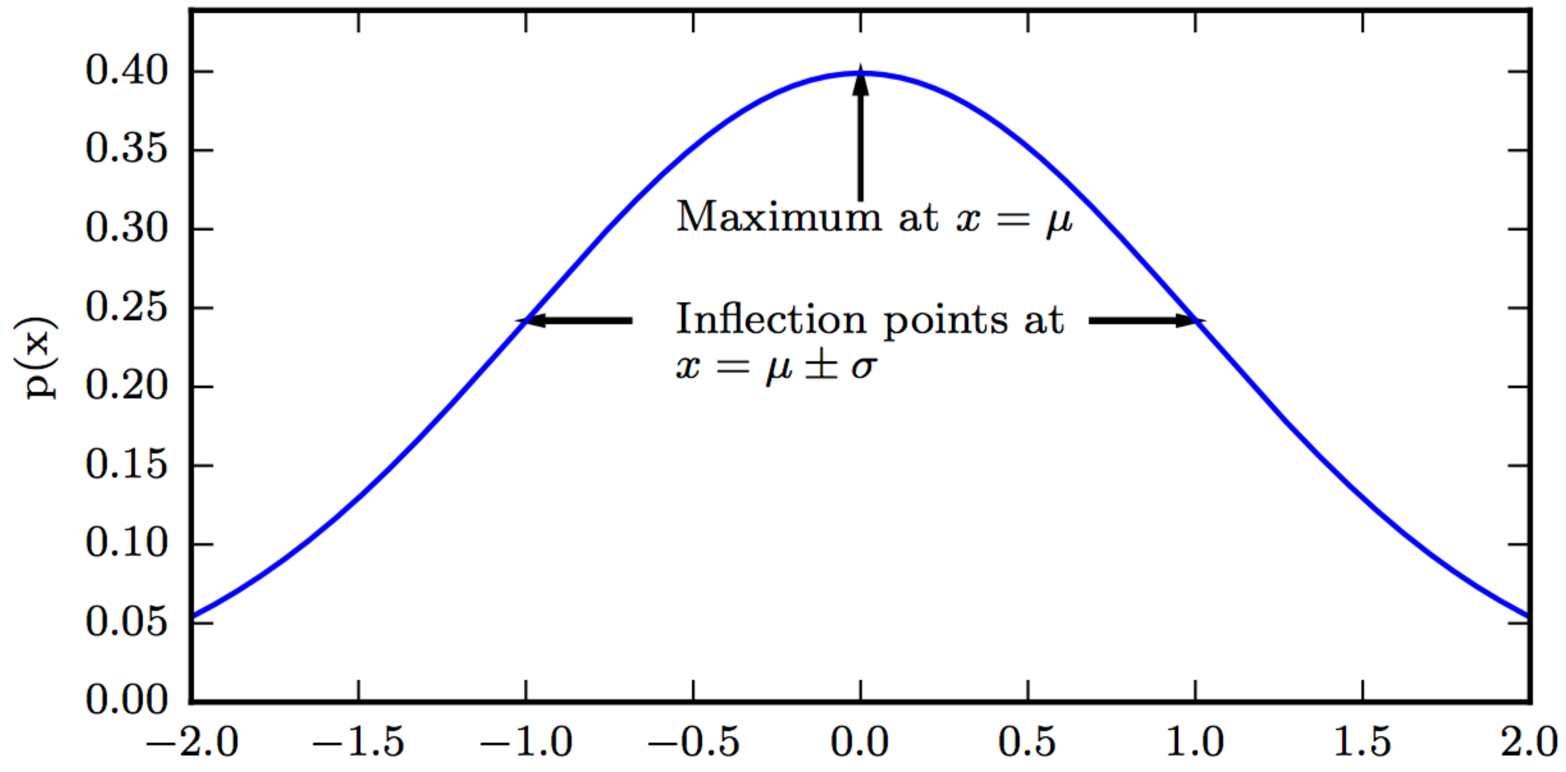


Figure 3.1

Multivariate Gaussian

- Parametrised by variance

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sqrt{\frac{1}{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Parametrised by precision

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\beta}^{-1}) = \sqrt{\frac{\det(\boldsymbol{\beta})}{(2\pi)^n}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\beta}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

More distributions

- Exponential

$$p(x;\lambda) = \lambda \mathbf{1}_{x \geq 0} \exp\left(-\lambda x\right)$$

- Laplacian

$$\text{Laplace}(x;\mu,\gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x-\mu|}{\gamma}\right)$$

- Dirac

$$p(x) = \delta(x-\mu)$$

More distributions

- Exponential

$$p(x;\lambda) = \lambda \mathbf{1}_{x\geq 0} \exp(-\lambda x)$$

Assign probability zero to all negative values of *x*

- Laplacian

$$\text{Laplace}(x;\mu,\gamma) = \frac{1}{2\gamma}\exp\left(-\frac{|x-\mu|}{\gamma}\right)$$

- Dirac

$$p(x) = \delta(x-\mu)$$

Mixture distributions

$$P(\mathrm{x}) = \sum_i P(\mathrm{c}=i)P(\mathrm{x} \mid \mathrm{c}=i)$$

Gaussian mixture with three components

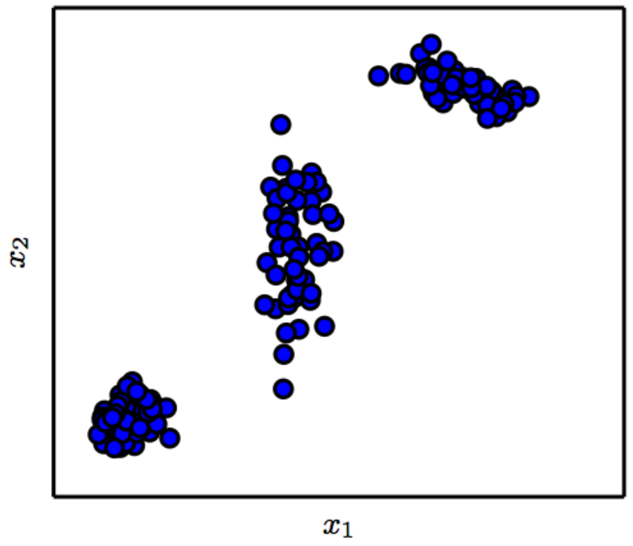


Figure 3.2

Logistic sigmoid

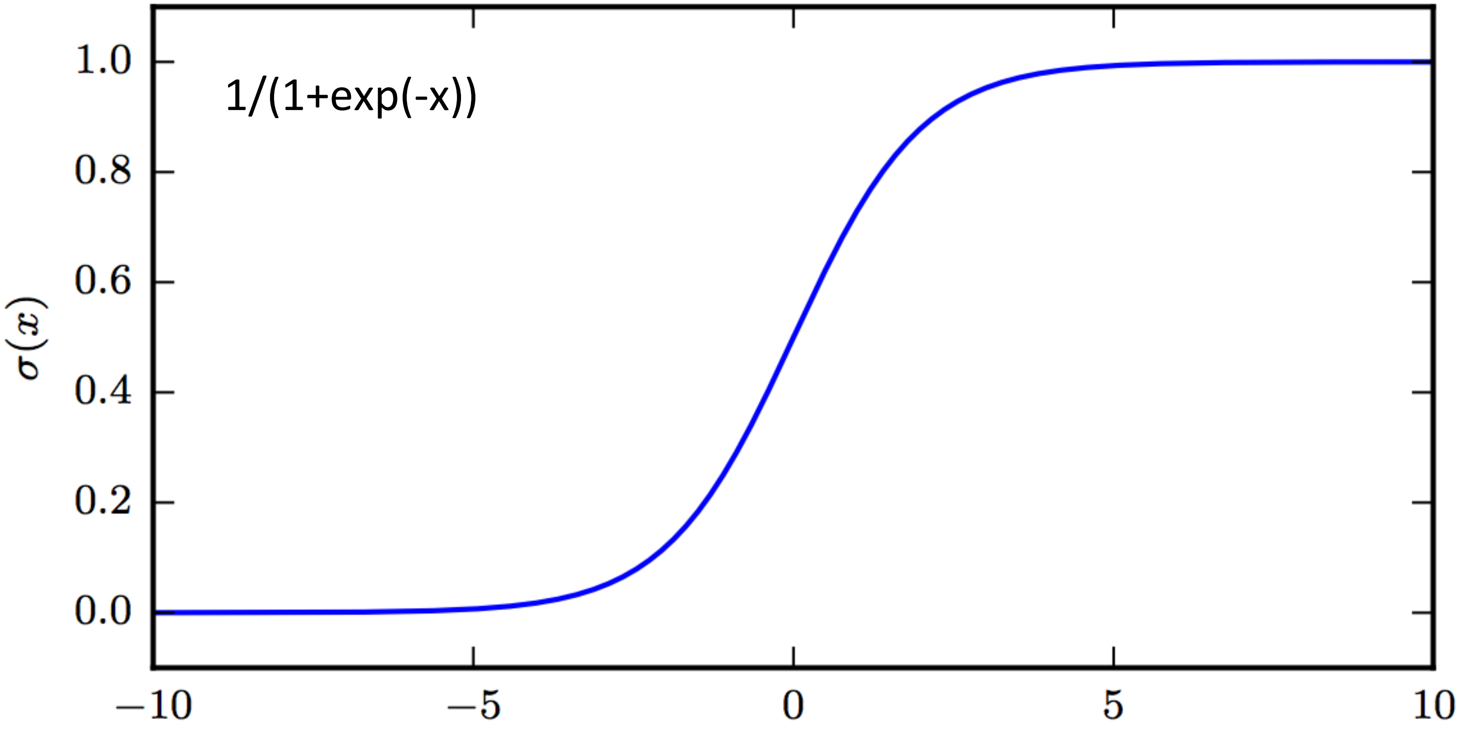


Figure 3.3: The logistic sigmoid function.

Commonly used to parametrise Bernoulli distributions

Softplus function

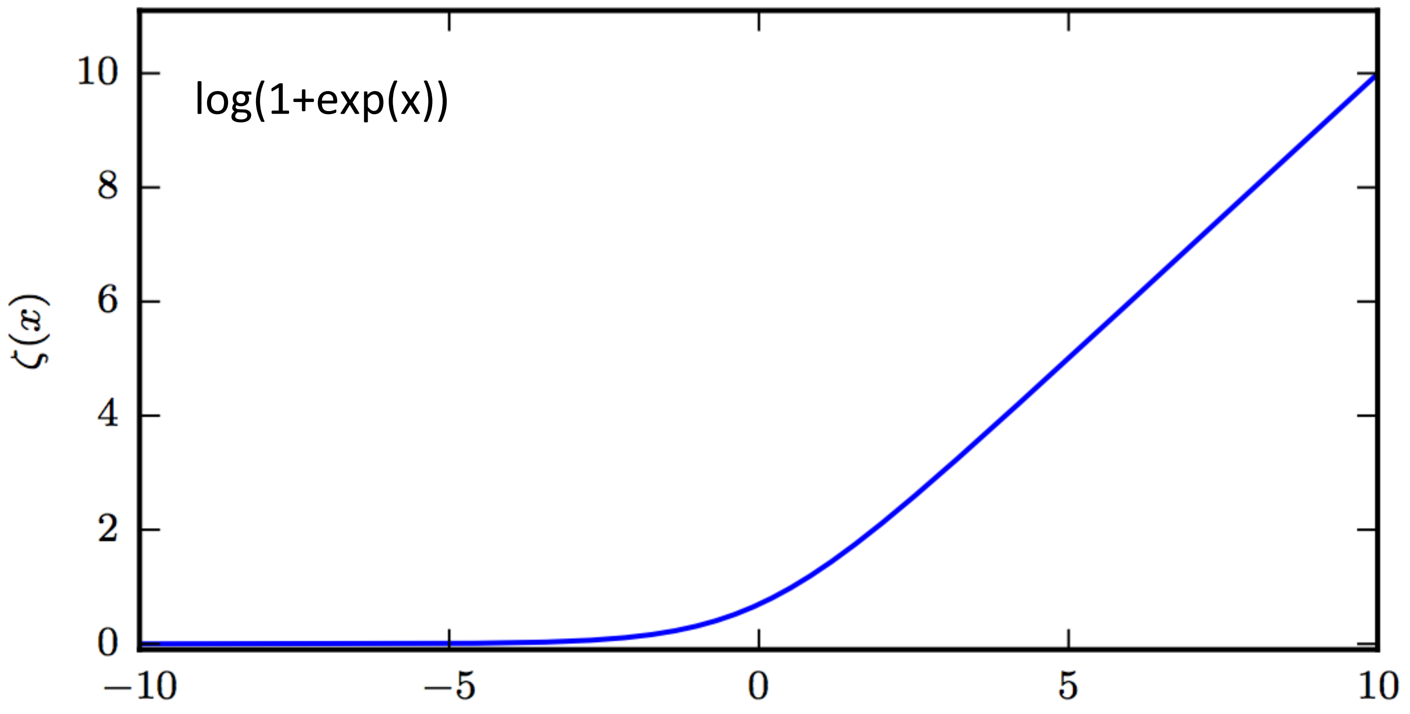


Figure 3.4: The softplus function.

Commonly used to parametrise normal distribution (beta)

Information theory

- Information

$$I(x) = -\log P(x)$$

- Entropy

$$H(\mathrm{x}) = \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)]$$

- KL divergence

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]$$

Entropy of a Bernoulli variable

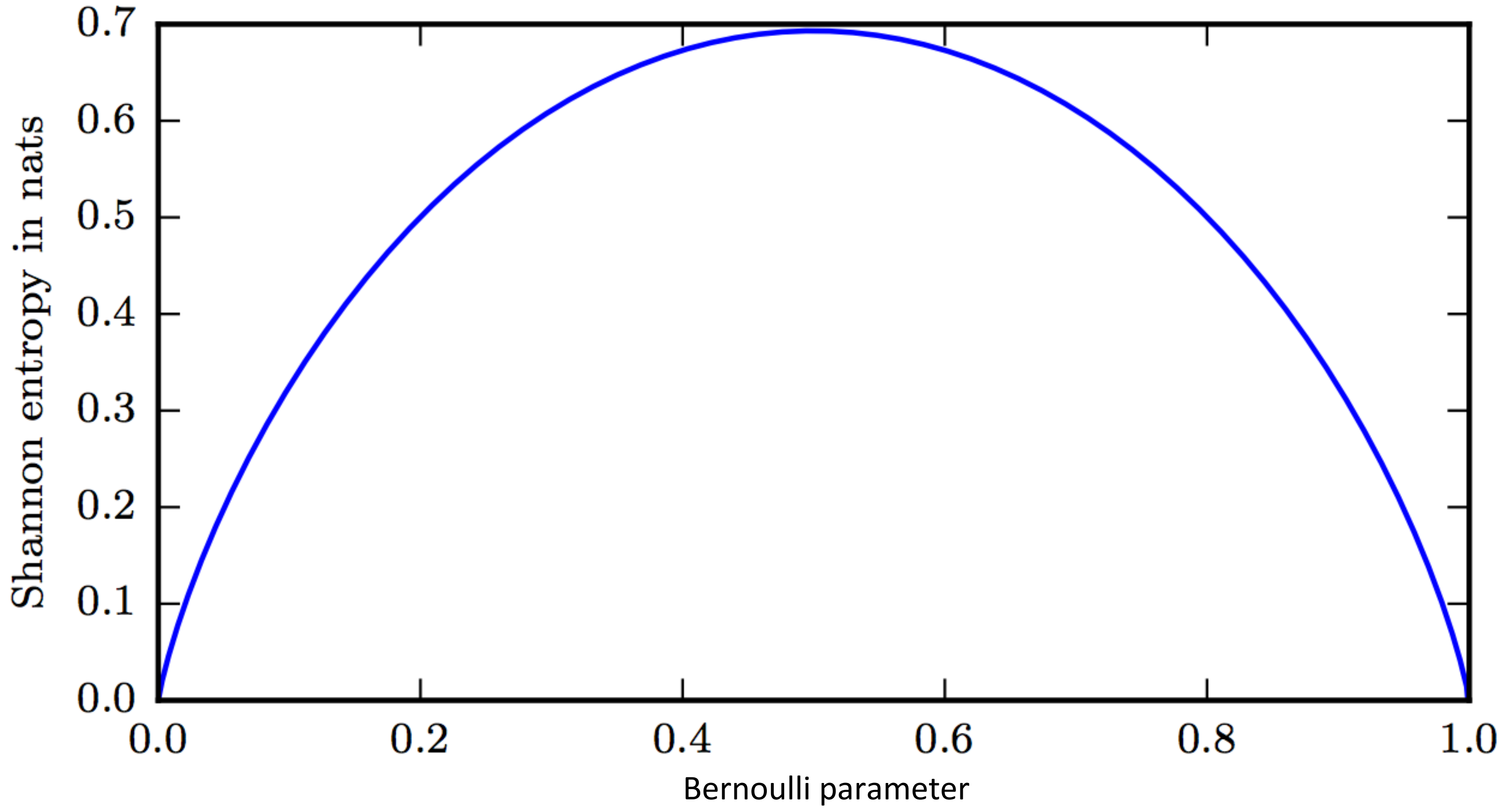


Figure 3.5

The KL divergence is asymmetric

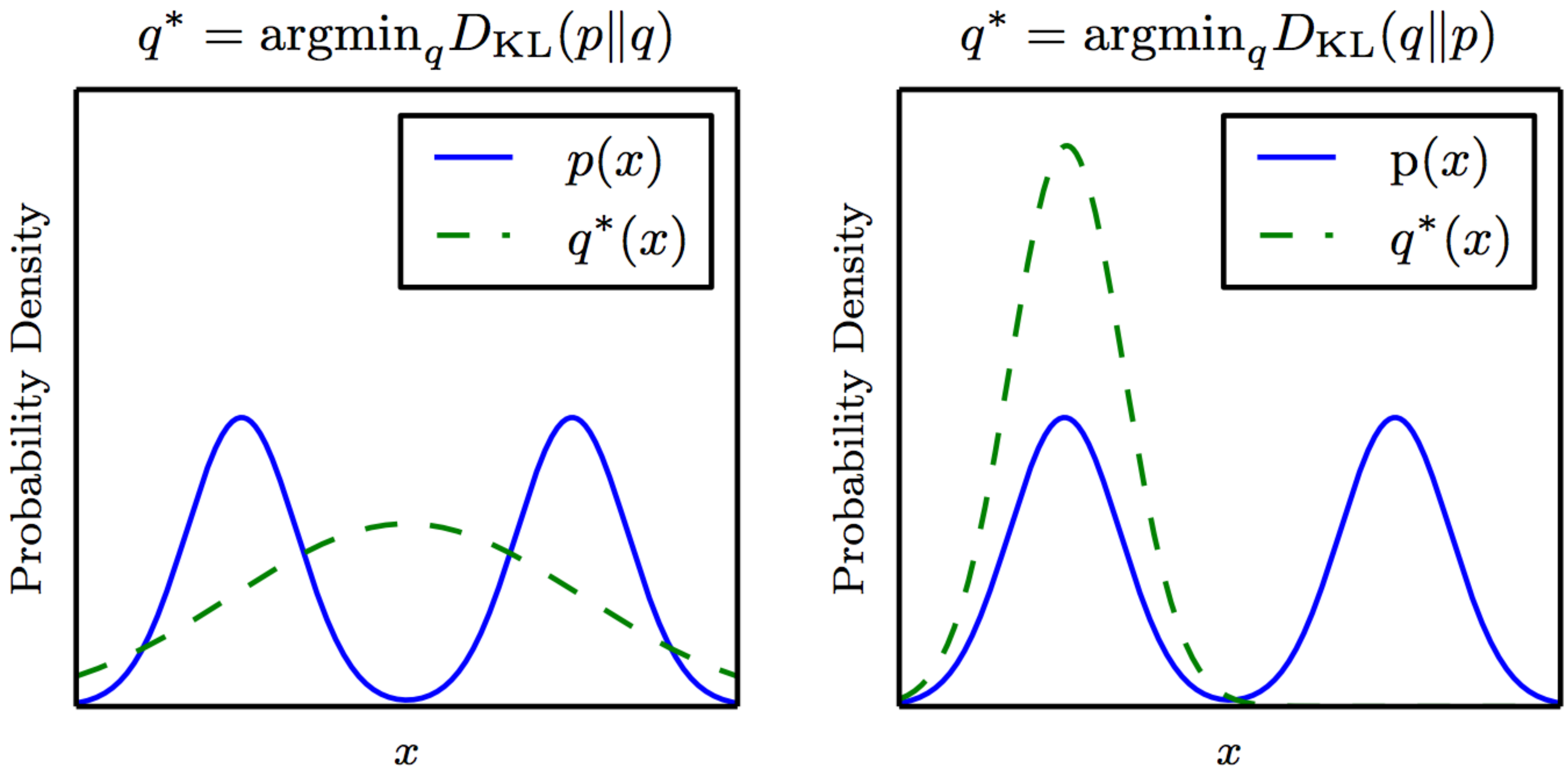
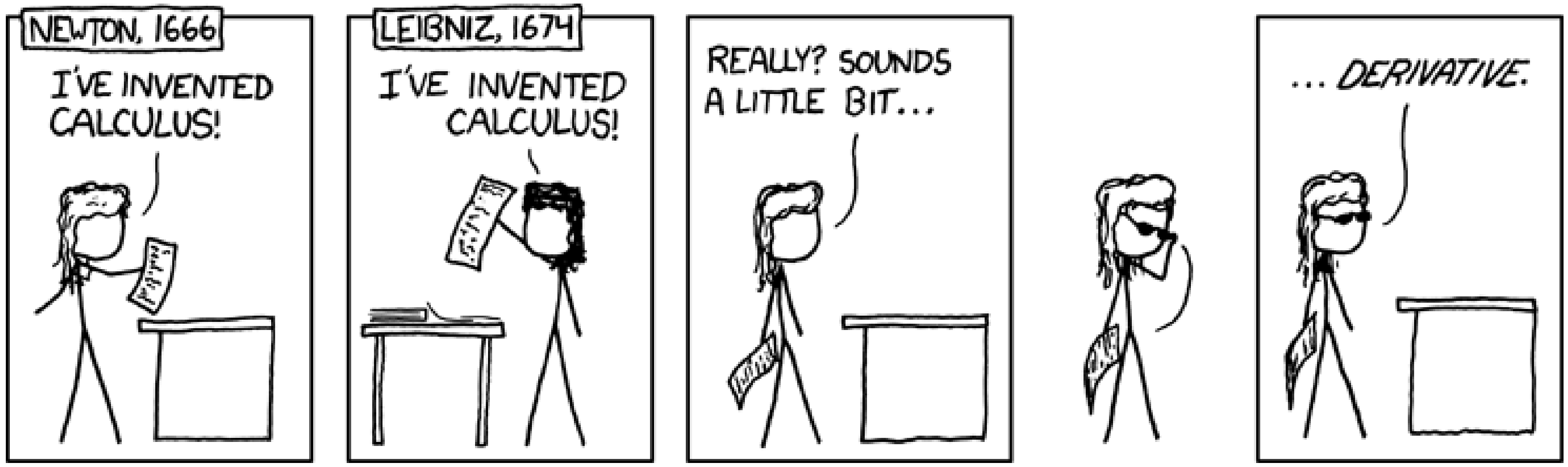


Figure 3.6

Optimisation and numerical computation


NEWTON, 1666
I'VE INVENTED CALCULUS!
LEIBNIZ, 1674
I'VE INVENTED CALCULUS!
REALLY? SOUNDS A LITTLE BIT...
... DERIVATIVE.

First order derivatives

- First order derivatives

$$\frac{dy}{dx}$$

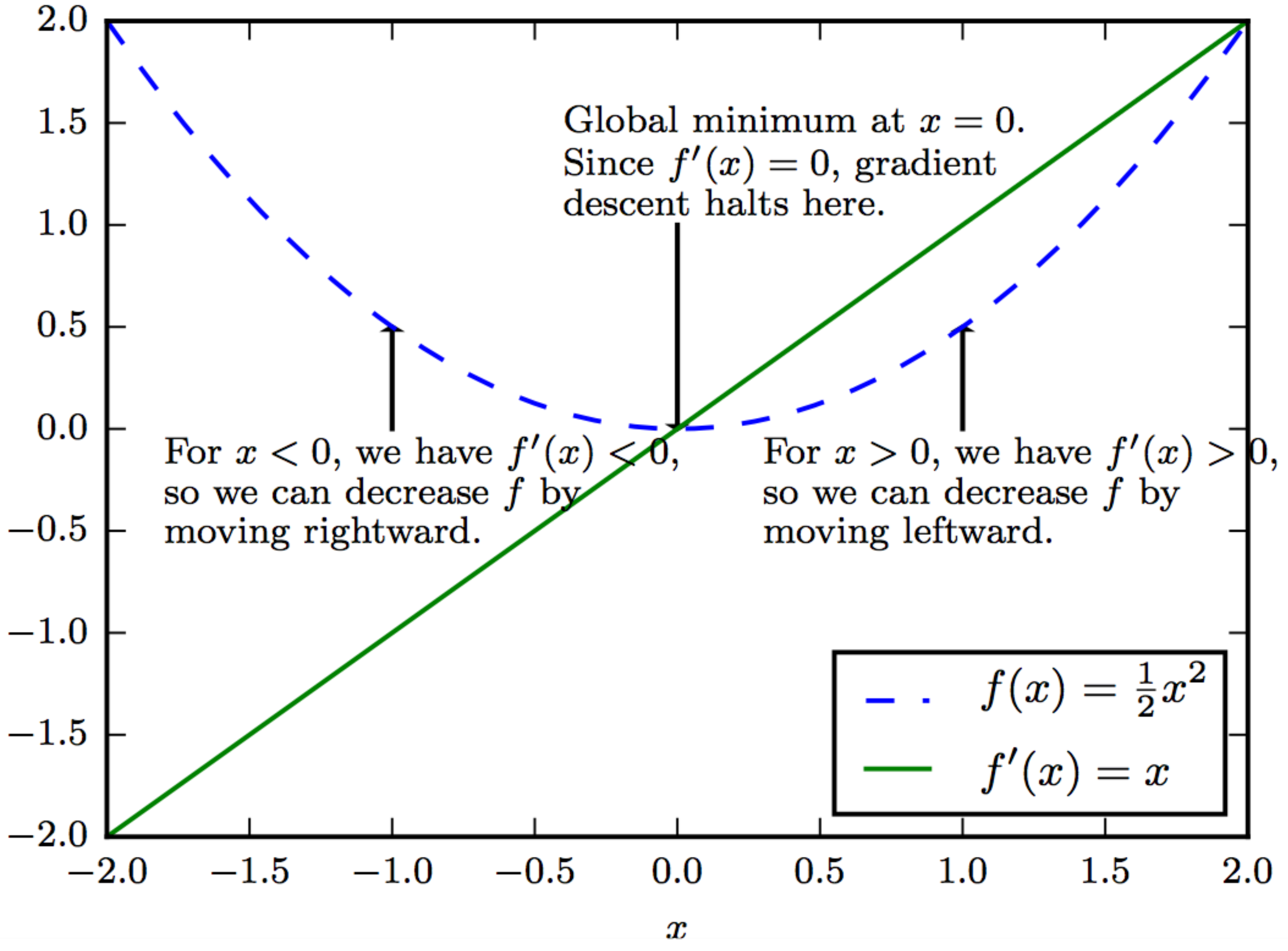
- Partial derivatives and gradient

$$\frac{\partial}{\partial x_i} f(\boldsymbol{x}) \qquad \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

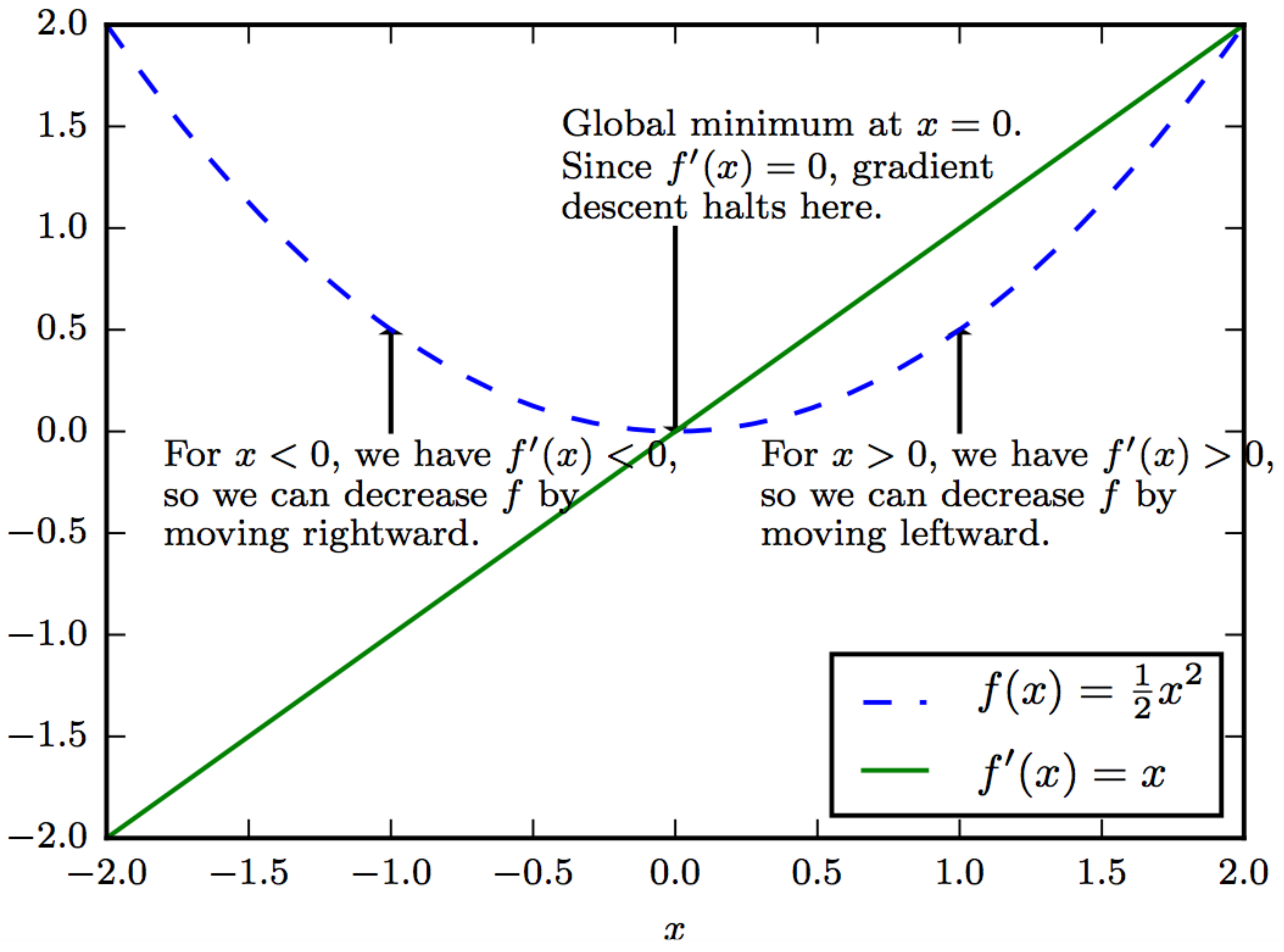
- Jacobian matrix

$$J_{i,j} = \frac{\partial}{\partial x_j} f(\boldsymbol{x})_i$$

Gradient descent

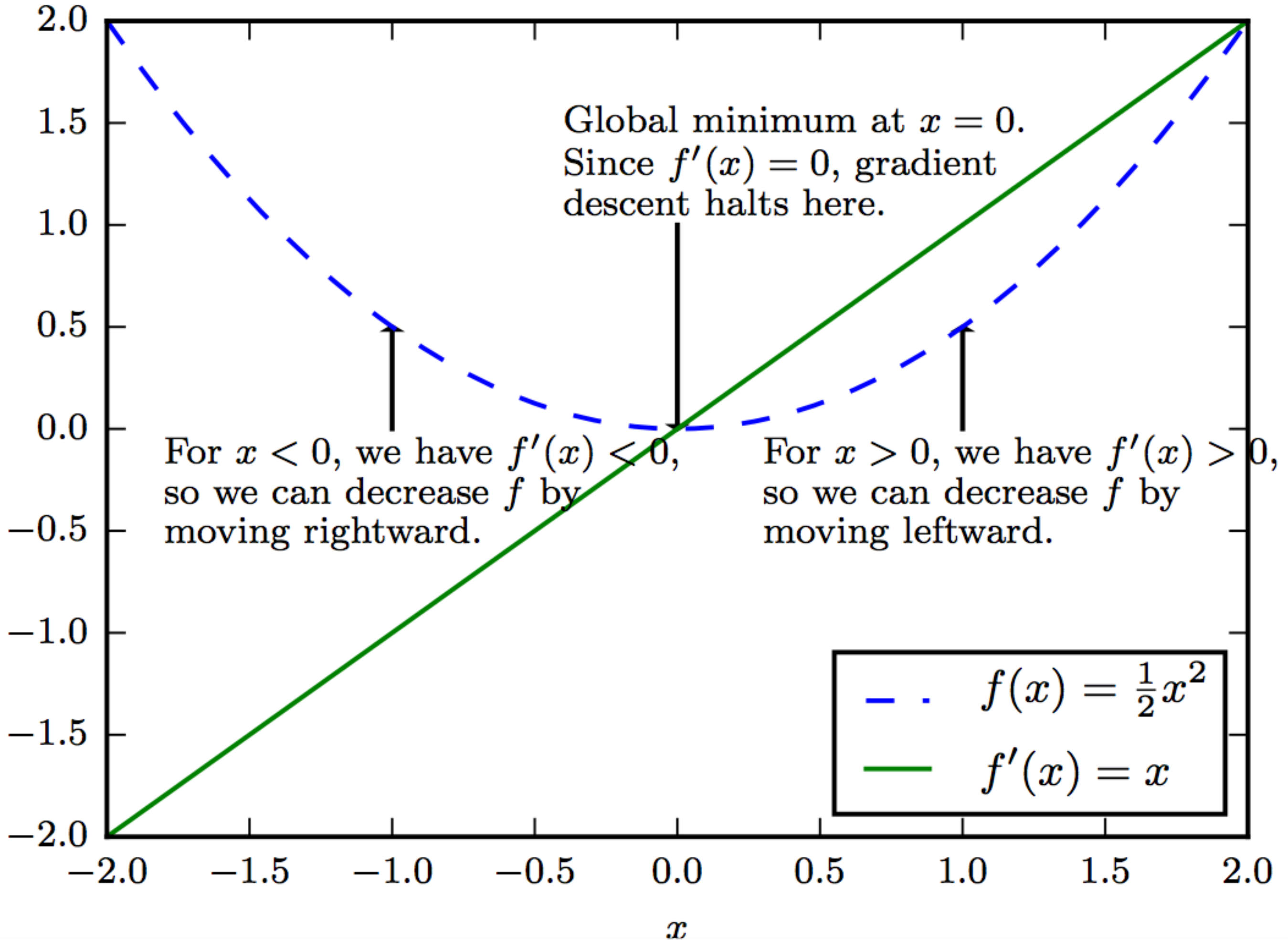


Gradient descent



If **x** is multi-dimensional we follow the gradient, i.e., vector of partial derivatives

Gradient descent



See http://www.deeplearningbook.org/contents/numerical.html page 83

Approximate optimisation

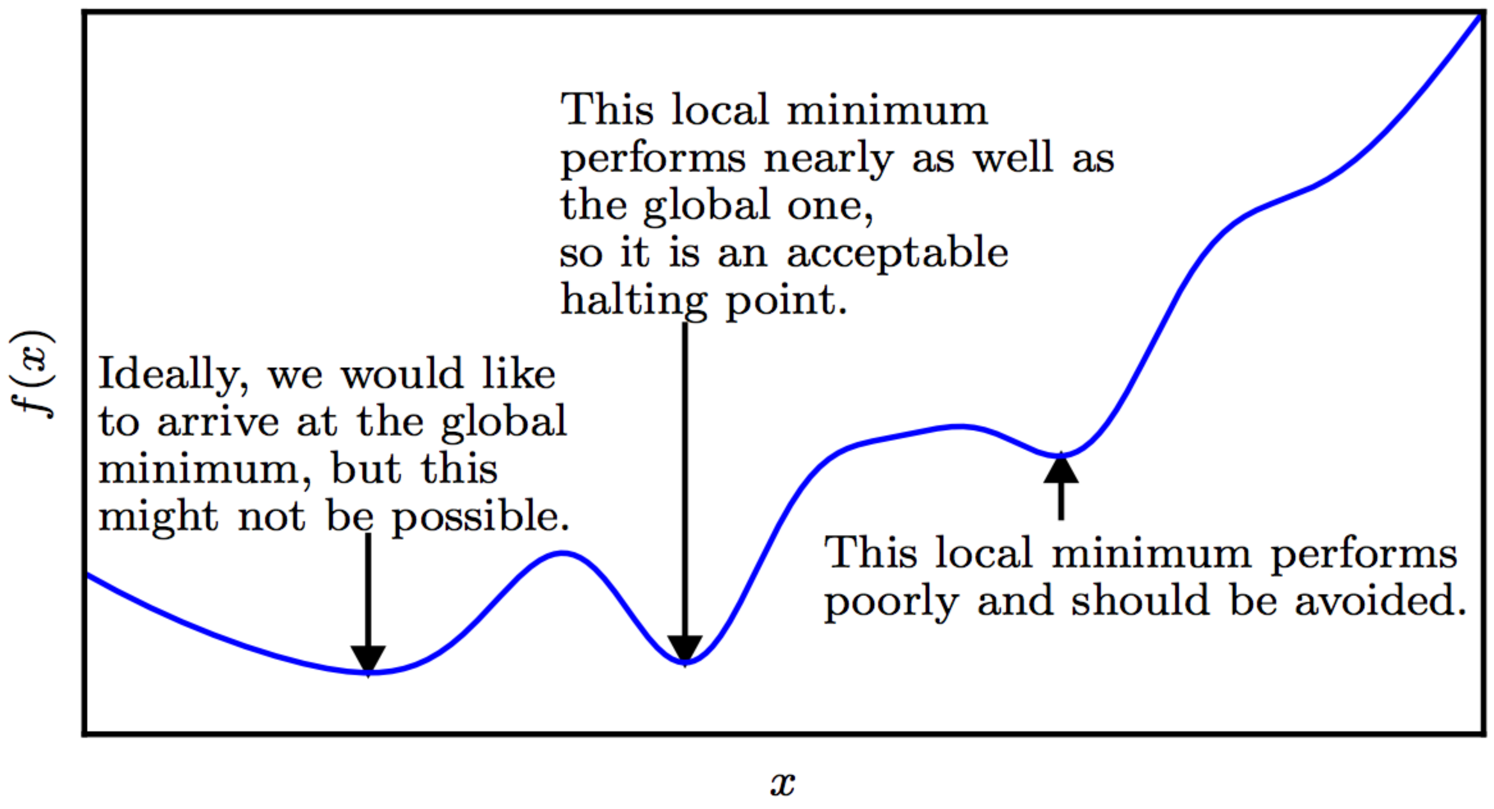


Figure 4.3

Critical points

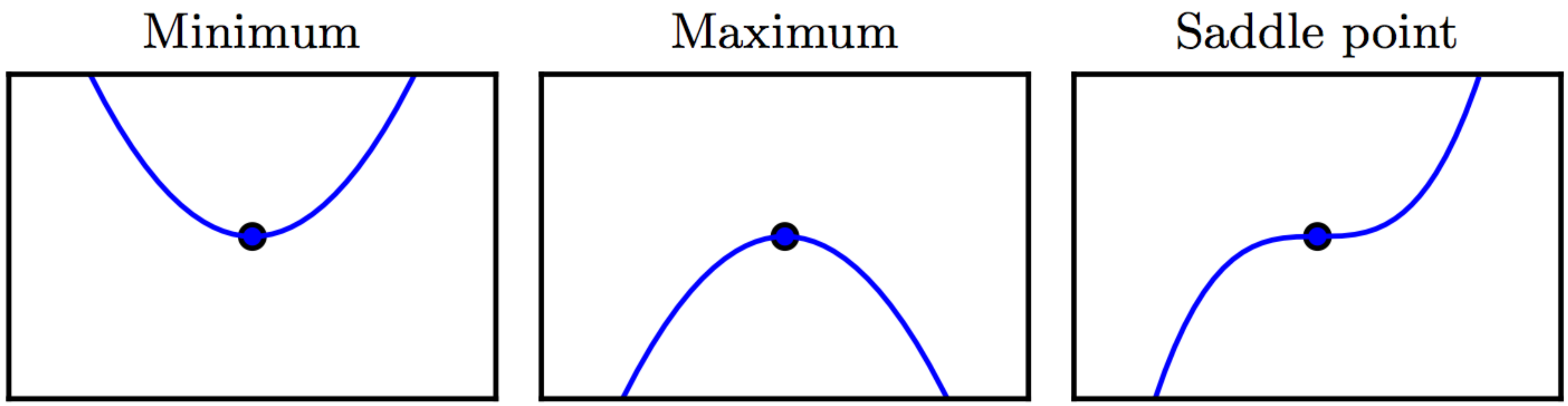


Figure 4.2

Optimising in deep learning

- Pure math way of life:
  - Find literally the smallest value of $f(x)$
  - Or maybe: find some critical point of $f(x)$ where the value is locally smallest
- Deep learning way of life:
  - Decrease the value of $f(x)$ a lot

Curvature

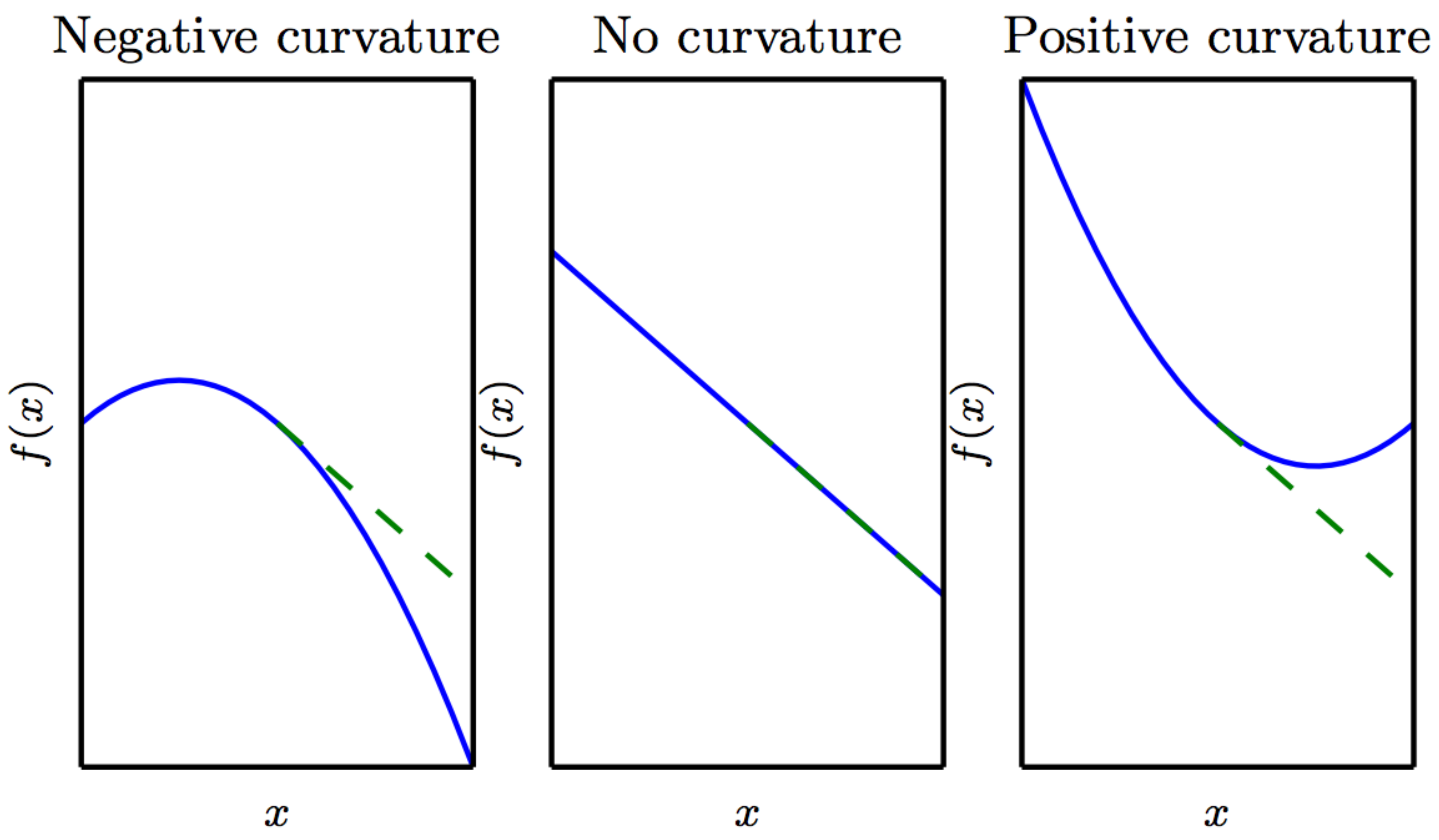


Figure 4.4

Second order derivatives

- Second order derivatives

$$\frac{d^2}{dx^2} f$$

- Hessian

$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$

Second order derivatives

- Second order derivatives

$$\frac{d^2}{dx^2} f$$

- Hessian

$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$

In deep learning usually **H** is a symmetric real-valued matrix,
thus it can be decomposed in a set of eigenvalues and orthogonal eigenvectors

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

Second order Taylor series approximation of f(x) around x(0)

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$\boxed{f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H}\boldsymbol{g}.}$$

Value of the function after gradient descent step

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

Value of the function before the step

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H}\boldsymbol{g}.$$

Improvement due to slope of function

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

Correction due to curvature

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

If this is positive, we can look for the optimal
step size that minimises the Taylor-series approximation

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon\boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon\boldsymbol{g}^\top\boldsymbol{g} + \frac{1}{2}\epsilon^2\boldsymbol{g}^\top\boldsymbol{H}\boldsymbol{g}.$$

$$\epsilon^* = \frac{\boldsymbol{g}^\top\boldsymbol{g}}{\boldsymbol{g}^\top\boldsymbol{H}\boldsymbol{g}}$$

Big gradients speed you up
(larger numerator means larger step size)

Optimal step size using Taylor series

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

Big eigenvalues slow you down if you align with their eigenvectors
(larger denominator means smaller step size)

“Real” issues

- Algorithms often specified in terms of real numbers
  - cannot be implemented with finite number of bits; does the algorithm still work?
- Do small changes in the input to a function cause large changes to an output?
  - Rounding/measurement errors, noise, can cause large changes
  - Iterative search for best input is difficult

Rounding and truncation errors

- In a digital computer, we use `float32` or similar schemes to represent real numbers
- A real number *x* is rounded to `x + delta` for some small delta
- Overflow: large *x* replaced by `inf`
- Underflow: small *x* replaced by `0`

Example

- Adding a very small number to a larger one may have no effect. This can cause large changes downstream:

```
>>> a = np.array([0., 1e-8]).astype('float3
>>> a.argmax()
1
>>> (a + 1).argmax()
0
```

Secondary effects

- Suppose we have code that computes `x-y`
- Suppose `x` overflows to `inf`
- Suppose `y` overflows to `inf`
- Then `x - y = inf - inf = NaN`

`exp`

- `exp(x)` overflows for large `x`
  - Doesn’t need to be very large
  - `float32: 89` overflows
  - Never use large `x`
- `exp(x)` underflows for very negative `x`
  - Possibly not a problem
  - Possibly catastrophic if `exp(x)` is a denominator, an argument to a logarithm, etc.

`log` and `sqrt`

- `log(0) = - inf`
- `log(<negative>)` is imaginary, usually `nan` in software
- `sqrt(0)` is `0`, but its *derivative* has a divide by zero
- Definitely avoid underflow or round-to-negative in the argument!

`log exp`

- `log exp(x)` is a common pattern
- Should be simplified to `x`
- Avoids:
  - Overflow in `exp`
  - Underflow in `exp` causing `-inf` in `log`

`log exp`

- `log exp(x)` is a common pattern
- Should be simplified to `x`
- Avoids:
  - Overflow in `exp`
  - Underflow in `exp` causing `-inf` in `log`

Also see https://blog.feedly.com/tricks-of-the-trade-logsumexp/

Bug hunting strategies

- If you increase your learning rate and the loss *gets stuck*, you are probably rounding your gradient to zero somewhere: maybe computing cross-entropy using probabilities instead of logits
- For correctly implemented loss, too high of learning rate should usually cause *explosion*

Bug hunting strategies

- If you see explosion (NaNs, very large values) immediately suspect:
  - log
  - exp
  - sqrt
  - division
- Always suspect the code that changed most recently