

# Deep Learning (CS324)

## 2. Mathematical background\*

Prof. Jianguo Zhang  
SUSTech

# 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.



# 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:

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

# Matrices and tensors

- A matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$  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

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

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$$

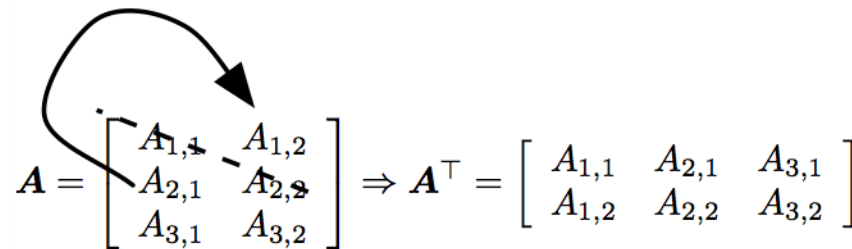

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \mathbf{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

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

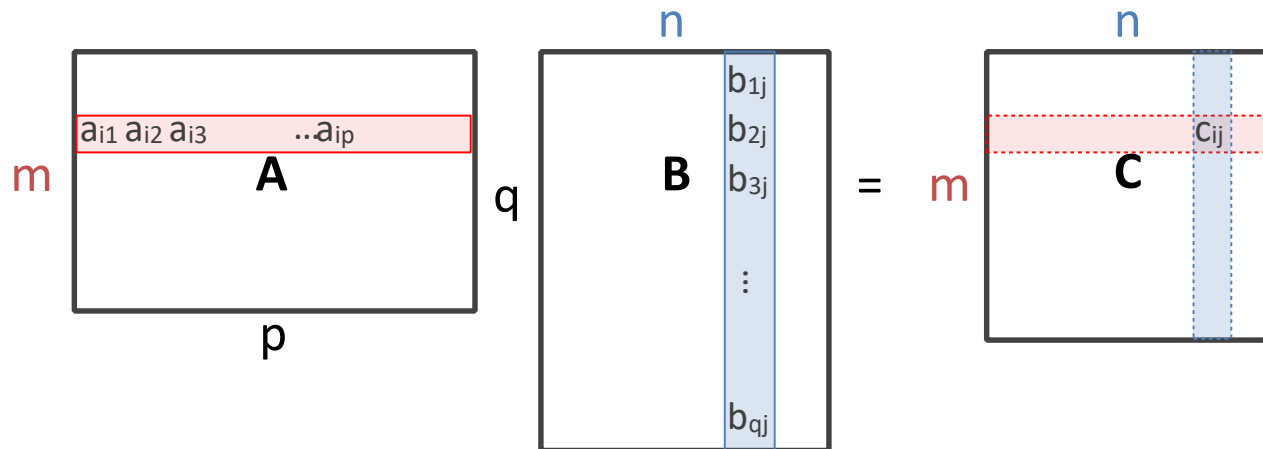
- The **multiplication** of a matrix **by a scalar**  $t$  produces as a result a matrix of the same size whose entries are each multiplied by  $t$

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



# 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

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

# 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  $\mathbf{I}_3$ .

- If  $\mathbf{A}$  is  $m \times n$  then

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

# Trace of a matrix

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

$$\text{Tr}(\mathbf{A}) = \sum_i \mathbf{A}_{i,i}$$

$$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA})$$

# Linear systems of equations

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

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

# Linear systems of equations

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

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$$

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

# Linear systems of equations

- A linear system of equations  $\mathbf{A}\mathbf{x} = \mathbf{b}$  can have:
  - No solution
  - Many solutions
  - Exactly one solution: if  $\mathbf{A}$  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
  - $f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = \mathbf{0}$
  - $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$  (the *triangle inequality*)
  - $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha|f(\mathbf{x})$

# Norms

- $L_p$  norms

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

- Most popular norm:  $L_2$  norm
- $L_1$  norm:

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

- Max norm, infinite  $p$ :

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



# 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  $\mathbf{A}$

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

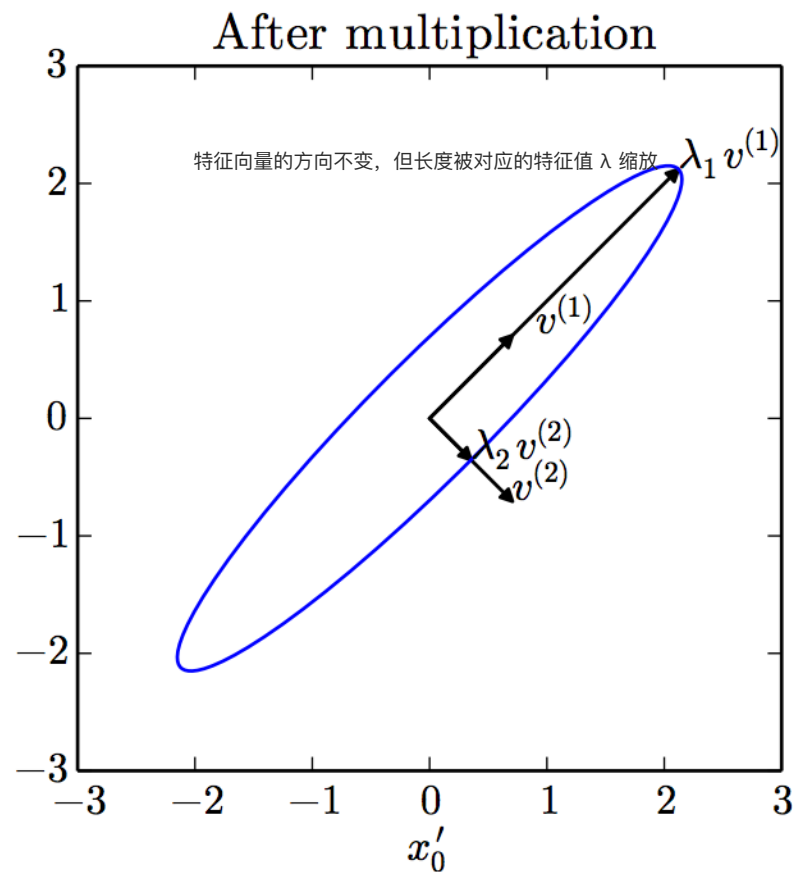
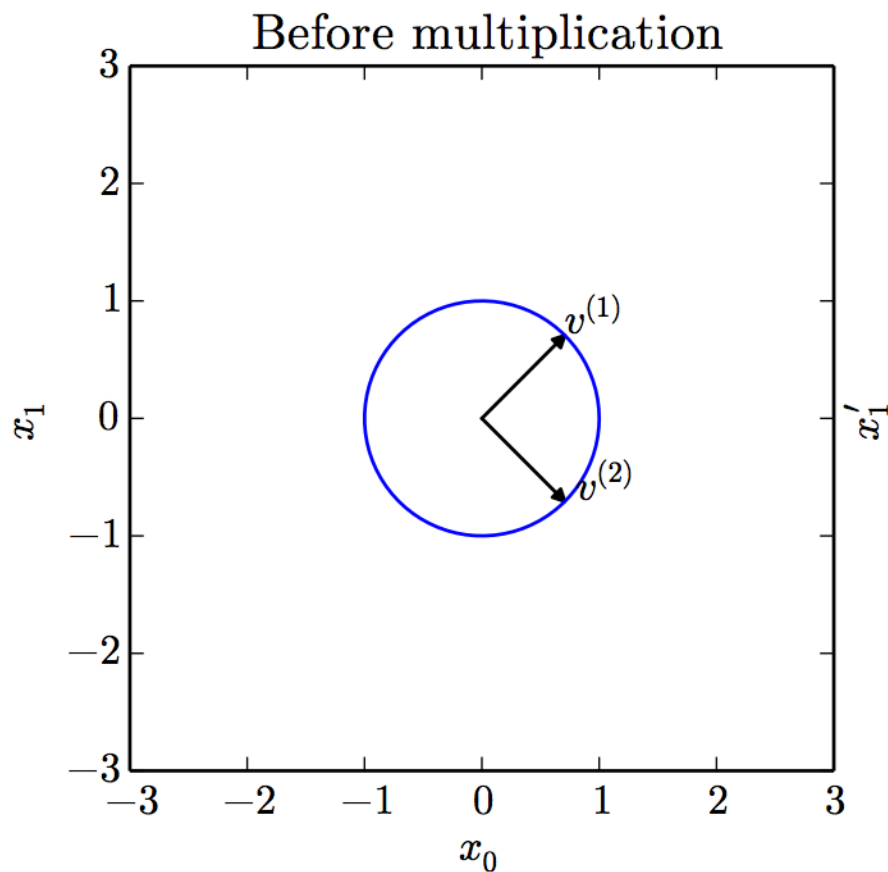
- Eigendecomposition of a diagonalisable matrix

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

- Every real symmetric matrix has a real, orthogonal eigendecomposition

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

# Eigenvalues interpretation



# Singular value decomposition

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

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

# Singular value decomposition

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

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

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

- More general; matrix need not be square

# Singular value decomposition

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

$$\mathbf{A} = \mathbf{U} \mathbf{D} \mathbf{V}^T$$

$n \times n$  orthogonal matrix (right-singular vectors - eigenvectors of  $\mathbf{A}^T \mathbf{A}$ )

- More general; matrix need not be square

# Singular value decomposition

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

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

*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?

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

SVD 的一个重要应用是计算 Moore–Penrose 伪逆

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

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



# Singular value decomposition

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

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

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

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

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

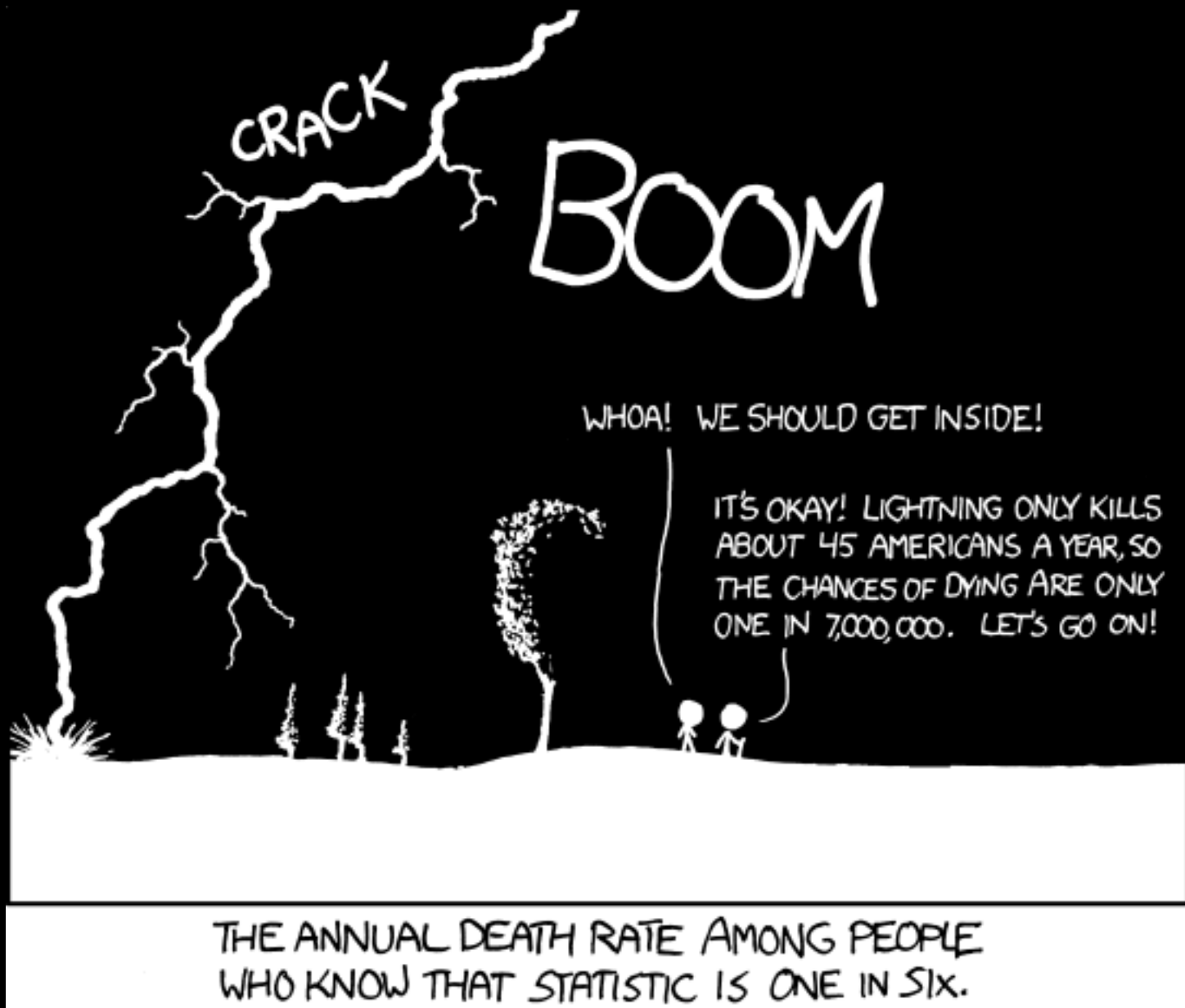
# Singular value decomposition

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

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

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

# Probability & information theory



# Probability mass function

- The domain of  $P$  must be the set of all possible states of  $\mathbf{x}$ .
- $\forall x \in \mathbf{x}, 0 \leq P(x) \leq 1$ . 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.
- $\sum_{x \in \mathbf{x}} P(x) = 1$ . 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.

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

Example: uniform distribution

# Probability density function

- The domain of  $p$  must be the set of all possible states of  $\mathbf{x}$ .
- $\forall x \in \mathbf{x}, p(x) \geq 0$ . 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)

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

- Conditional probability

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

- Chain rule

$$P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}) = P(\mathbf{x}^{(1)}) \prod_{i=2}^n P(\mathbf{x}^{(i)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(i-1)})$$

# A few important rules...

- Independence

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

- Conditional independence

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



# Bayes' Rule

$$P(x | y) = \frac{P(x)P(y | x)}{P(y)}$$

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

$$P(x | y) = \frac{P(x)P(y | x)}{P(y)}$$

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

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

- of a continuous random variable

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

- Linearity of expectations

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

# Variance and covariance

- Variance

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

- Covariance

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

- Covariance matrix given a random vector  $\mathbf{x}$

$$\text{Cov}(\mathbf{x})_{i,j} = \text{Cov}(x_i, x_j)$$

# Bernoulli distribution

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

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

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

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

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

# 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

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

$$\beta = \frac{1}{\sigma^2}$$

# Gaussian distribution

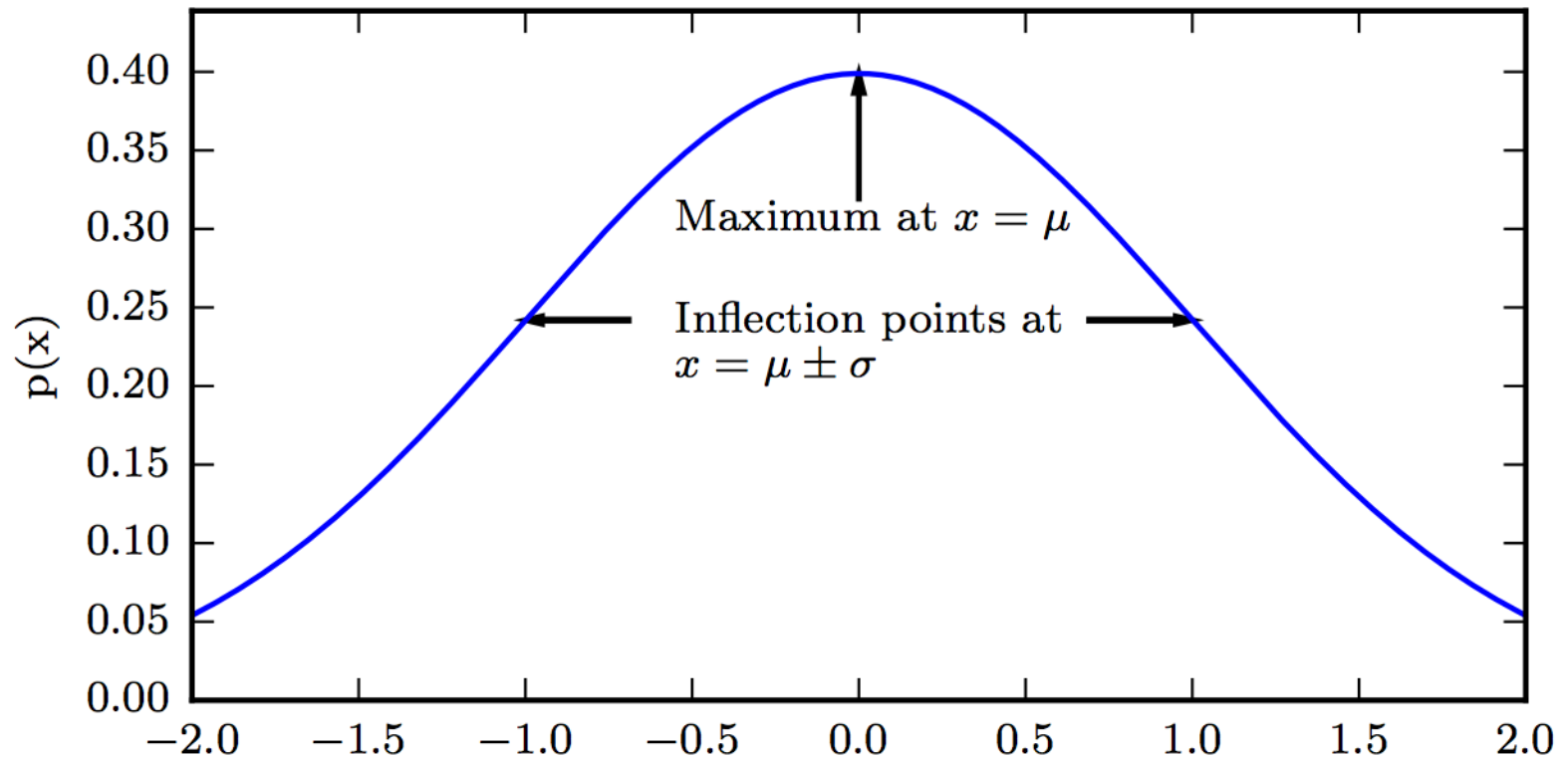


Figure 3.1

# Multivariate Gaussian

- Parametrised by variance

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

- Parametrised by precision

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



# More distributions

- Exponential

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

- 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

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

- Laplacian Assign probability zero to all negative values of  $x$

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

- Dirac

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

# Mixture distributions

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

Gaussian mixture with three components

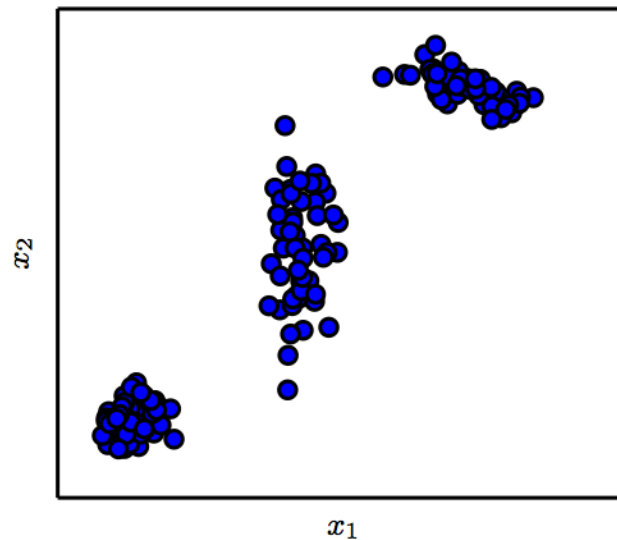


Figure 3.2

# Logistic sigmoid

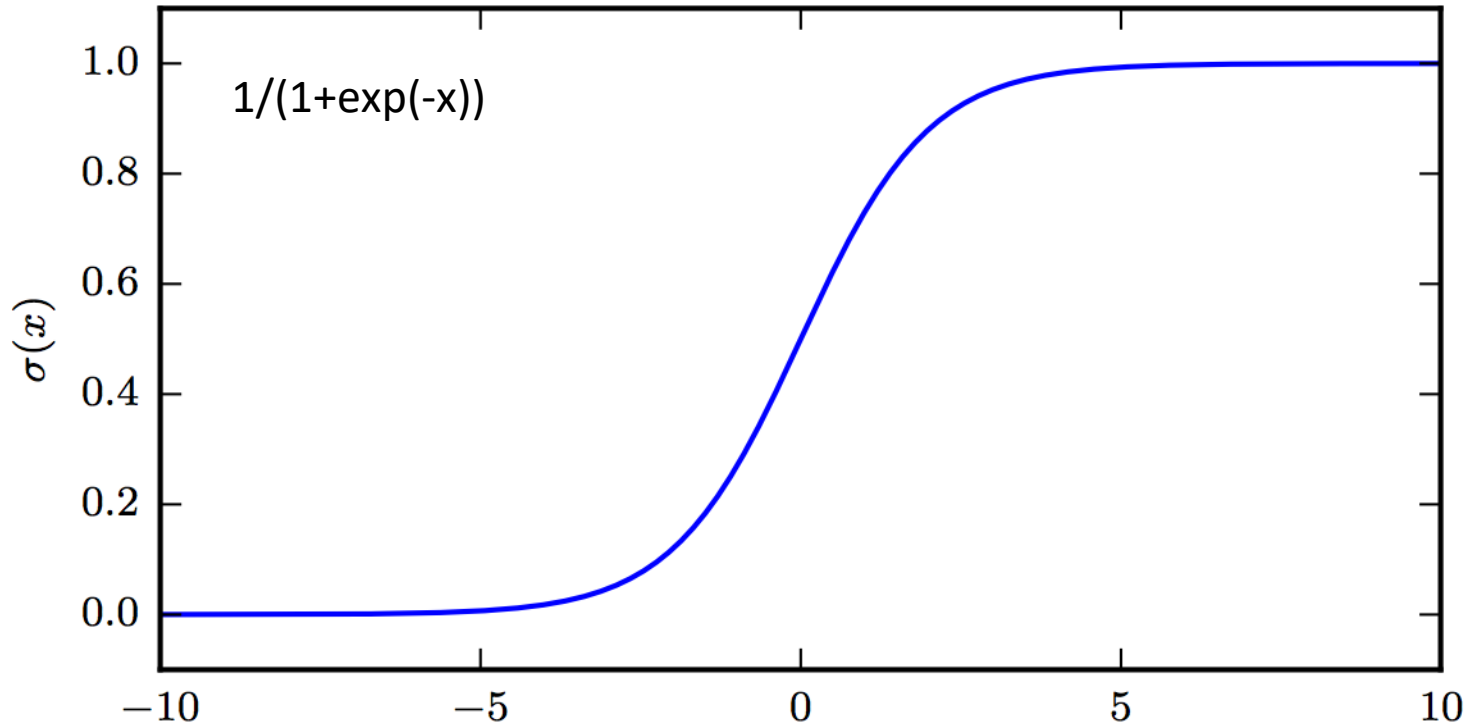


Figure 3.3: The logistic sigmoid function.

Commonly used to parametrise Bernoulli distributions

# Softplus function

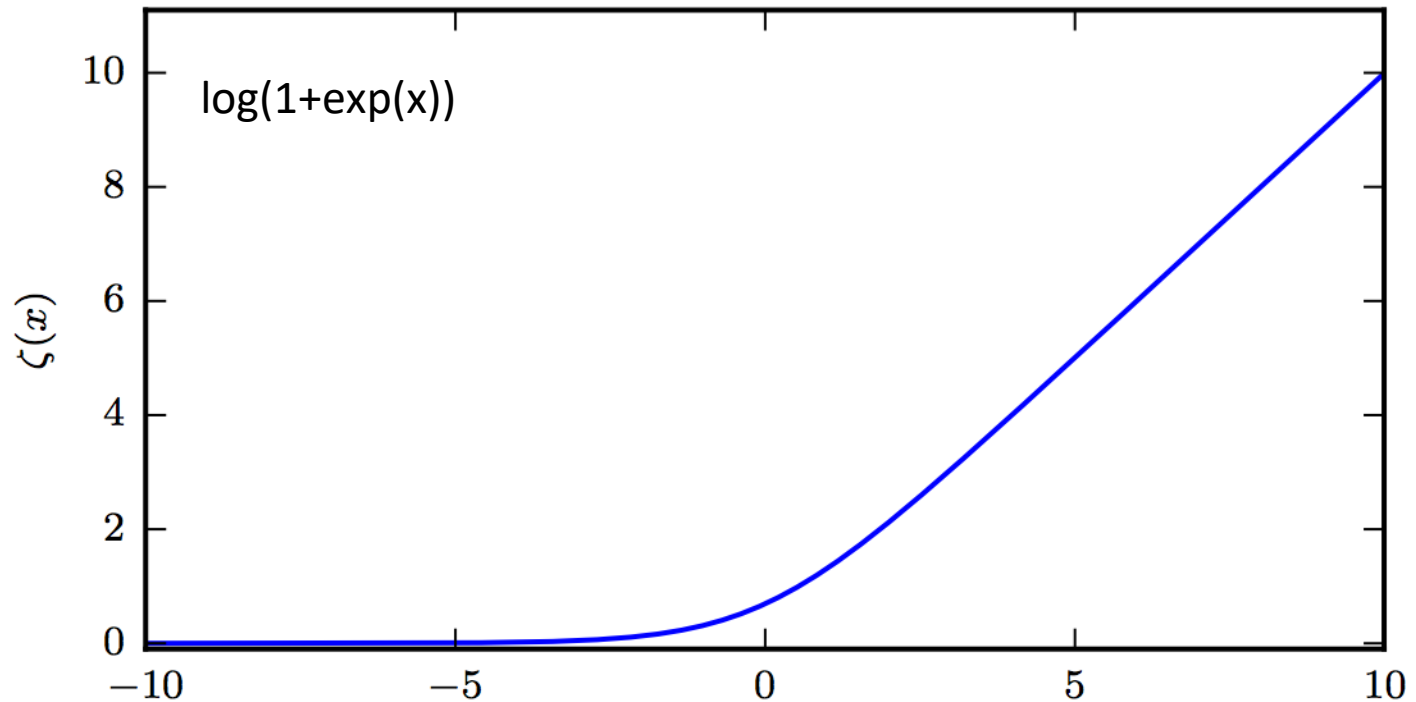


Figure 3.4: The softplus function.

Commonly used to parametrise normal distribution (beta)

# Information theory

- Information

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

- Entropy

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

- KL divergence

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

# Entropy of a Bernoulli variable

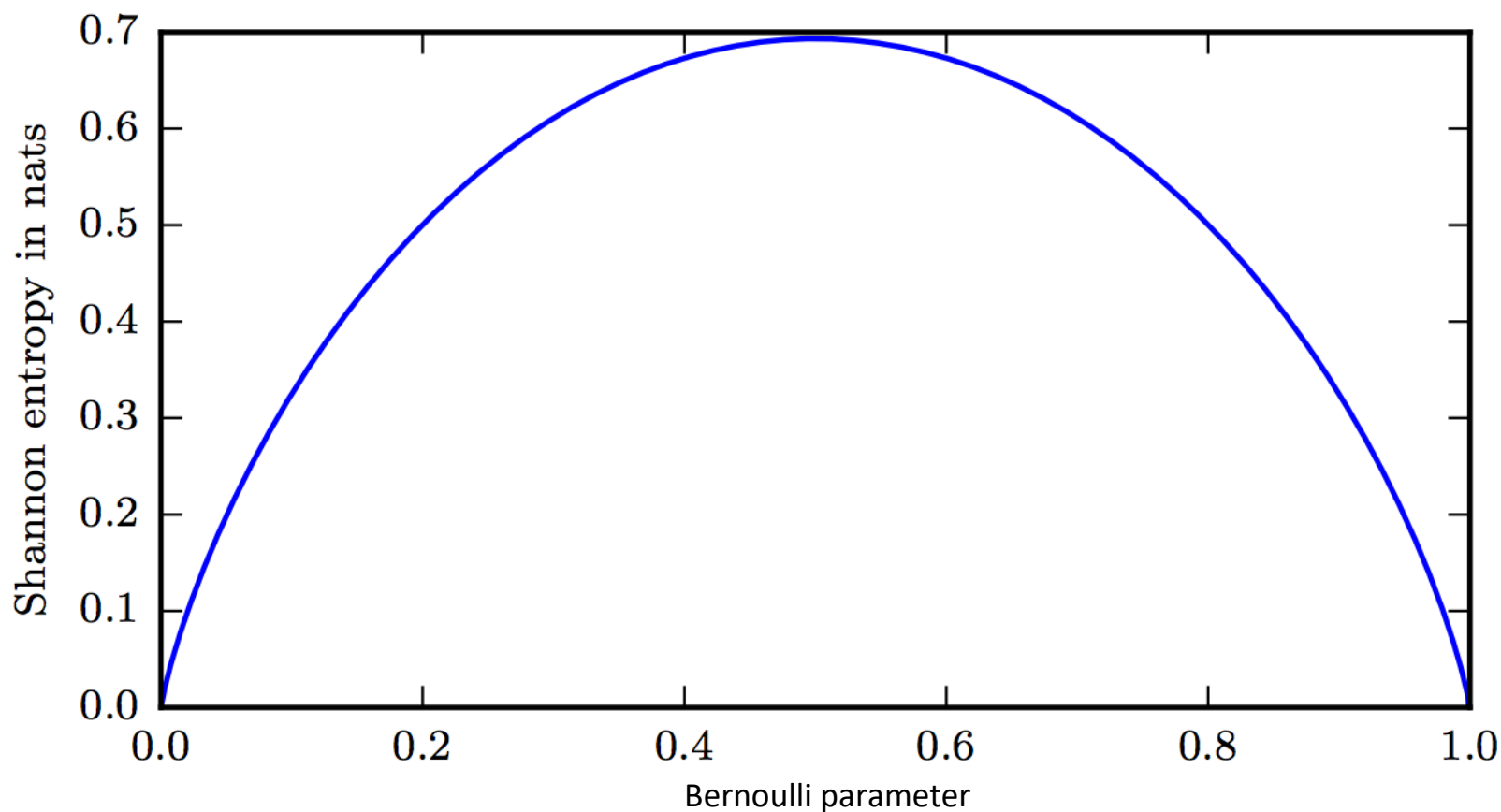
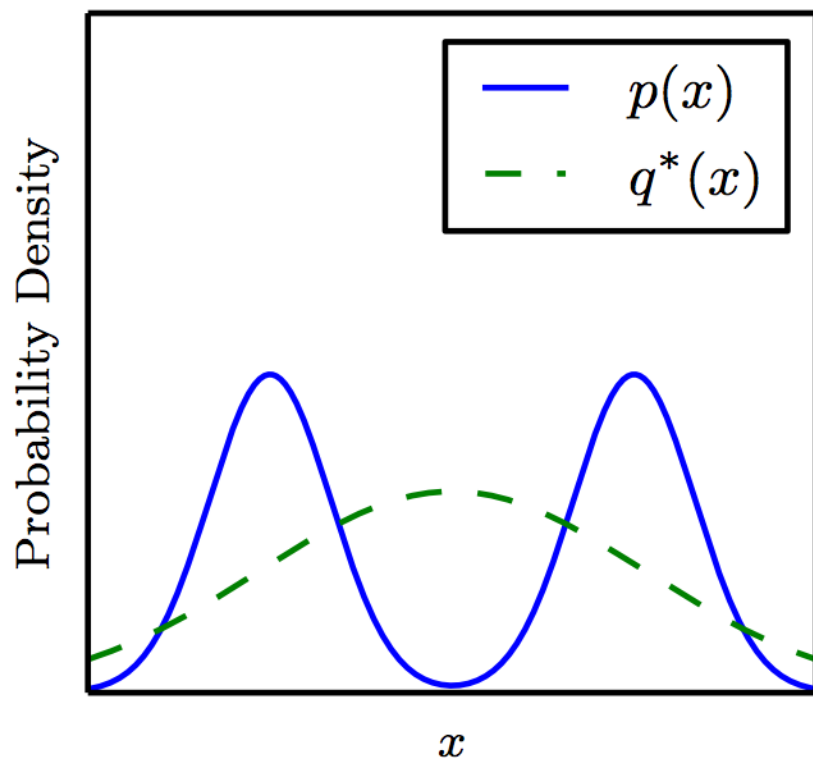


Figure 3.5

# The KL divergence is asymmetric

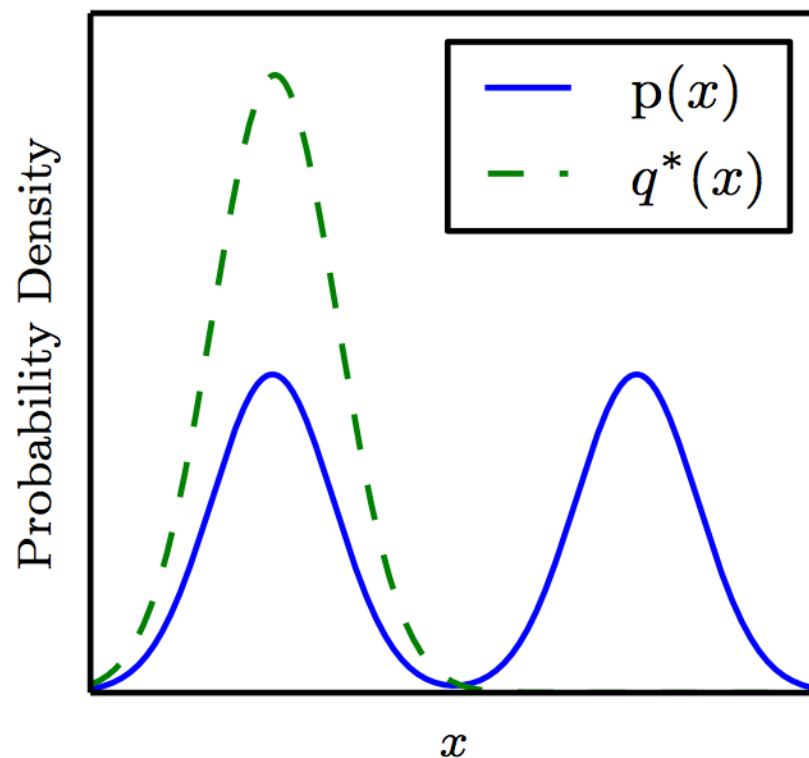
不对称性

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p \| q)$$



倾向于“覆盖”所有模式

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q \| p)$$

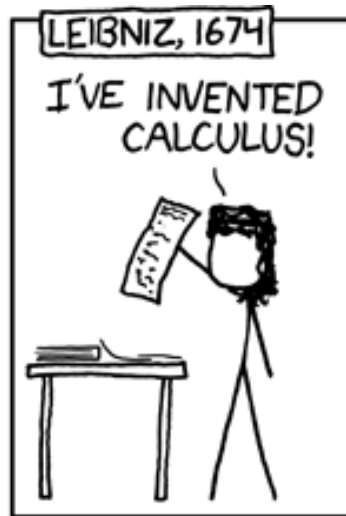
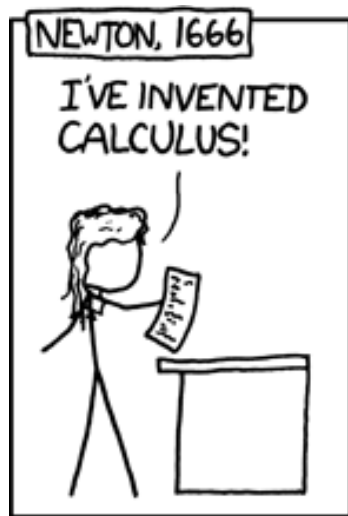


倾向于“选择”一个模式

Figure 3.6



# Optimisation and numerical computation



# First order derivatives

- First order derivatives

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

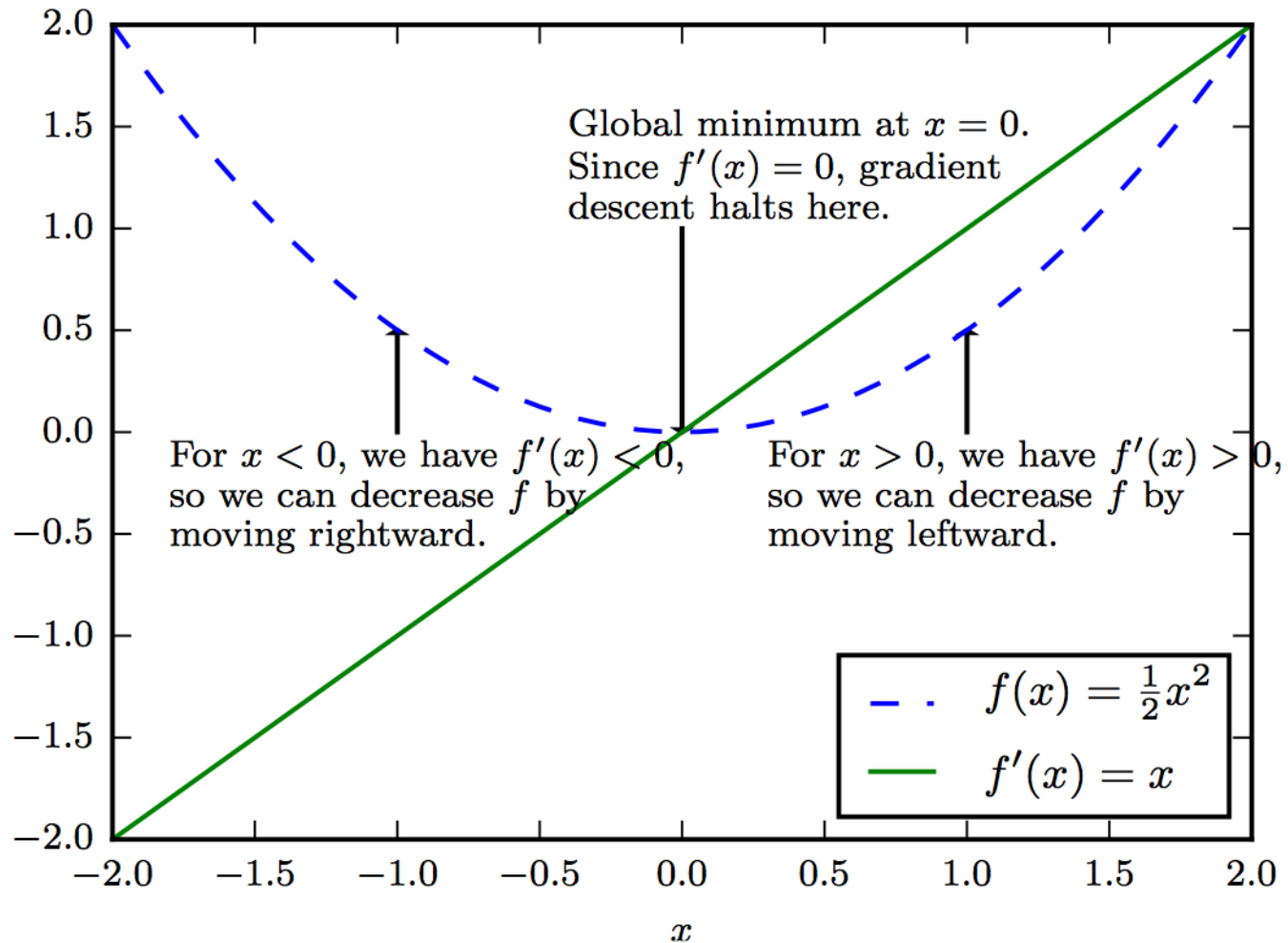
- Partial derivatives and gradient

$$\frac{\partial}{\partial x_i} f(\mathbf{x}) \quad \nabla_{\mathbf{x}} f(\mathbf{x})$$

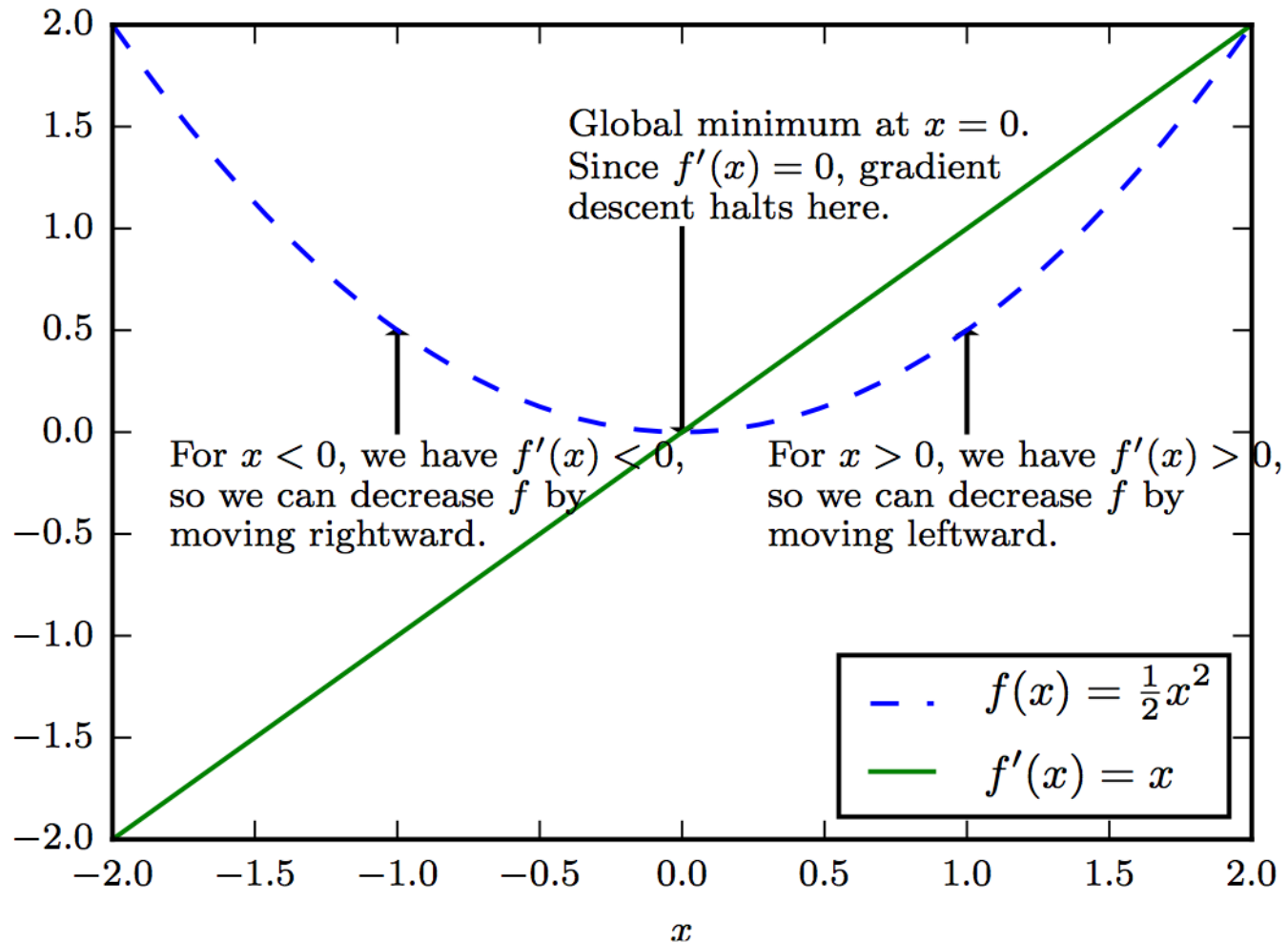
- Jacobian matrix

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

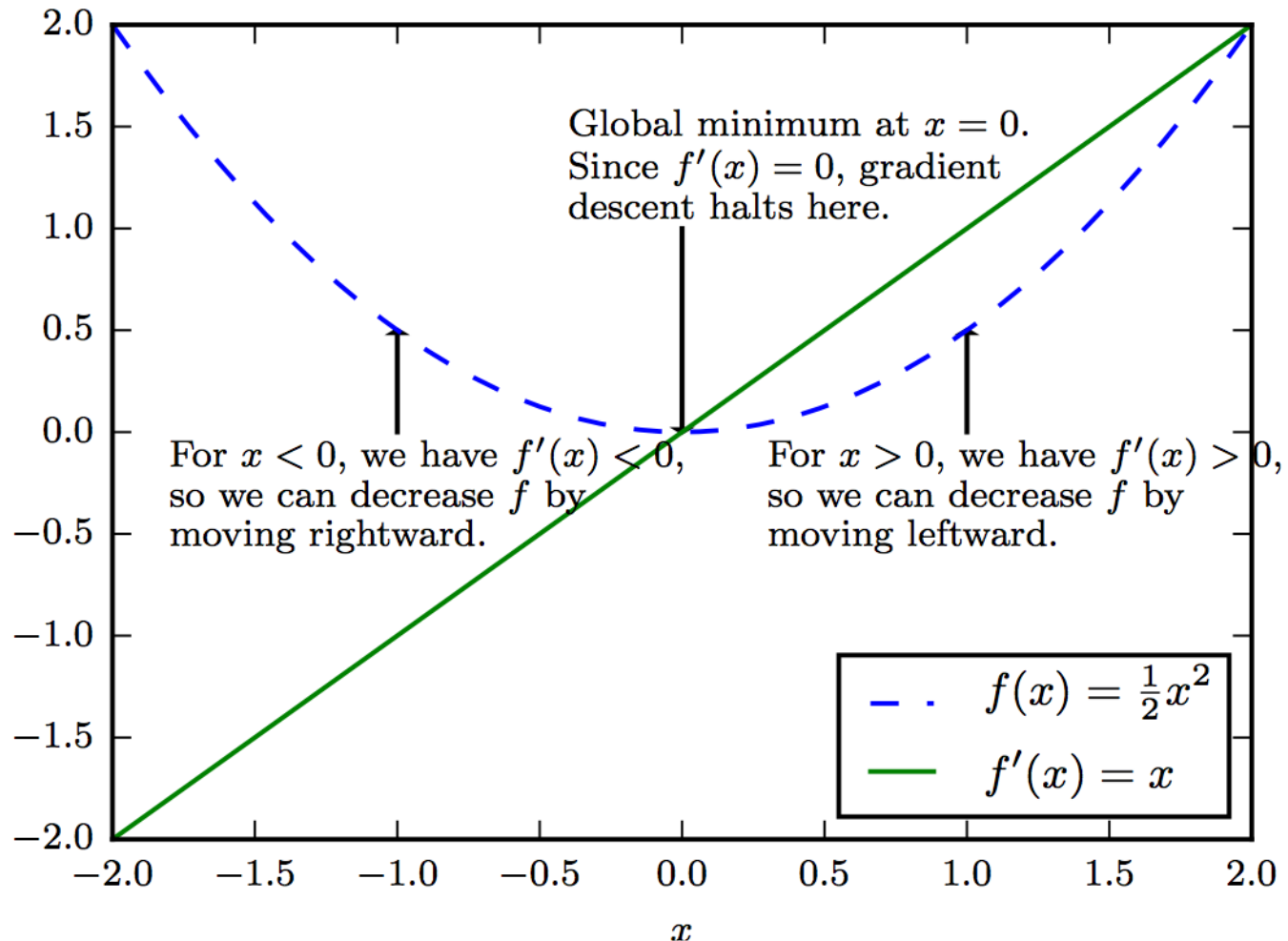
# Gradient descent



# Gradient descent



# Gradient descent



# Approximate optimisation

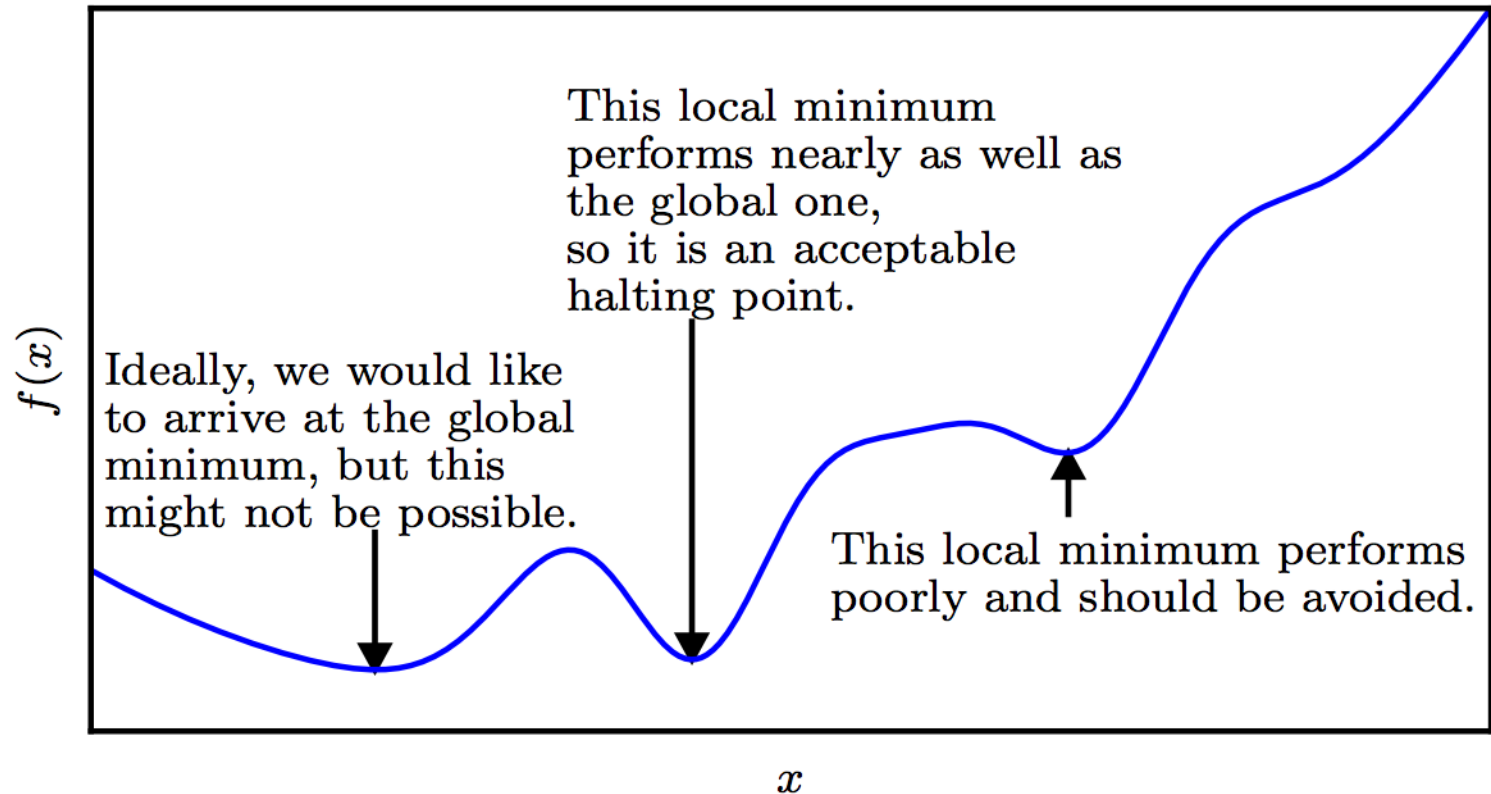
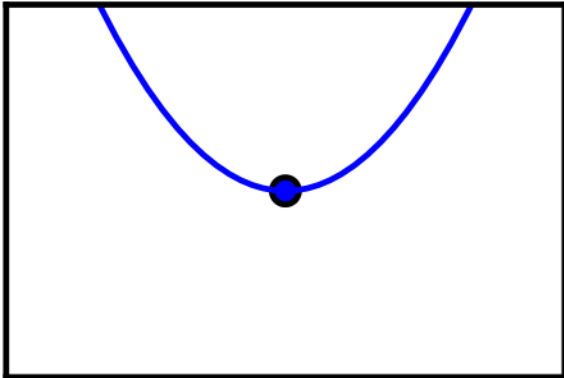


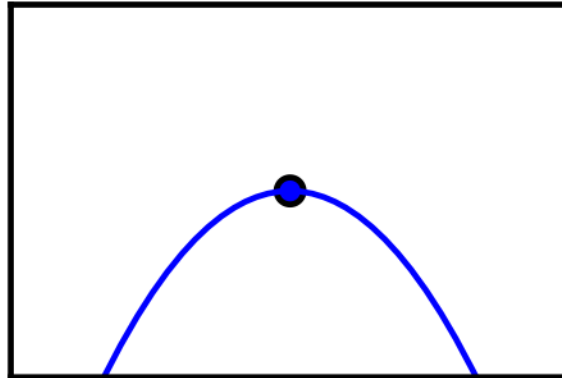
Figure 4.3

# Critical points

Minimum



Maximum



Saddle point

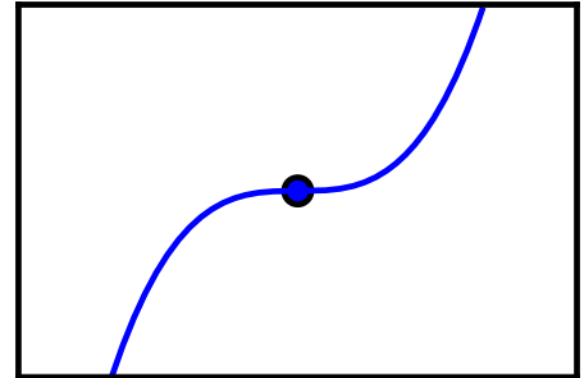


Figure 4.2



# 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

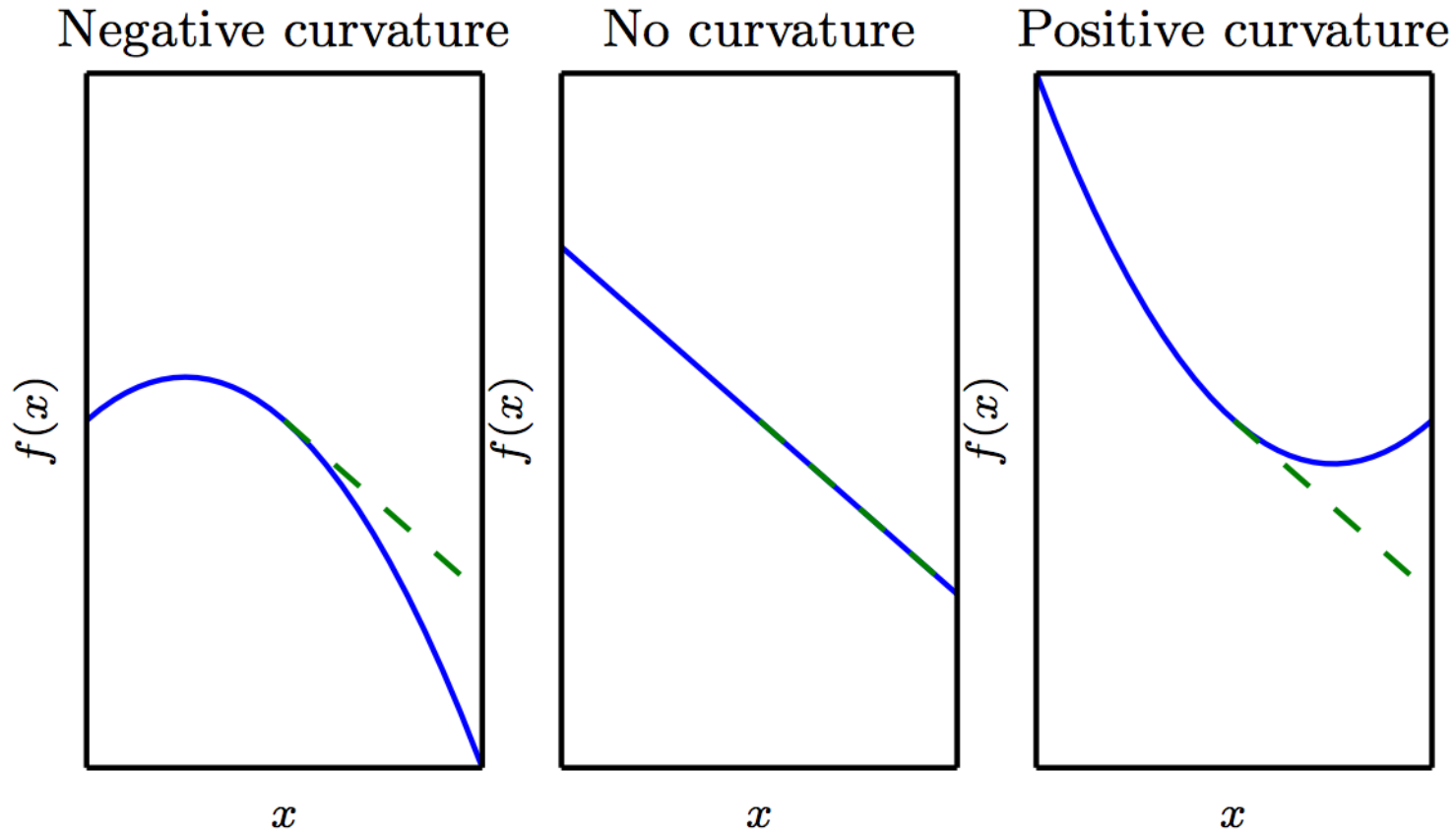


Figure 4.4

# Second order derivatives

- Second order derivatives

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

- Hessian

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

# Second order derivatives

- Second order derivatives

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

- Hessian

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

In deep learning usually  $\mathbf{H}$  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

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

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

# Optimal step size using Taylor series

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

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

Second order Taylor series approximation of  $f(\mathbf{x})$  around  $\mathbf{x}(0)$

# Optimal step size using Taylor series

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

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

Value of the function after gradient descent step

# Optimal step size using Taylor series

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

$$f(\mathbf{x}^{(0)} - \epsilon \mathbf{g}) \approx \underbrace{f(\mathbf{x}^{(0)})}_{\text{Value of the function before the step}} - \epsilon \mathbf{g}^\top \mathbf{g} + \frac{1}{2} \epsilon^2 \mathbf{g}^\top \mathbf{H} \mathbf{g}.$$

Value of the function before the step



# Optimal step size using Taylor series

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

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

梯度方向的下降 带来的改进

这正是 梯度下降 的核心：顺着负梯度方向走，函数值会下降  
( 因为  $\mathbf{g}^\top \mathbf{g} > 0$  )

Improvement due to slope of function

# Optimal step size using Taylor series

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

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

Correction due to curvature

二阶修正项，反映了函数的 曲率 (curvature) 对下降的影响

限制因素 (correction)，提醒我们不要走太大步

# Optimal step size using Taylor series

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

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

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

# Optimal step size using Taylor series

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

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

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

# Optimal step size using Taylor series

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

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

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

Big gradients speed you up

(larger numerator means larger step size)

# Optimal step size using Taylor series

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

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

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

代表函数的曲率 (curvature) 在梯度方向上的大小

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  $x + \text{delta}$  for some small  $\text{delta}$
- 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('float32')
>>> 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  $x - y = \text{inf} - \text{inf} = \text{NaN}$

# `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  $-\text{inf}$  in  $\log$

# log exp

- $\log \exp(x)$  is a common pattern
- Should be simplified to  $x$
- Avoids:
  - Overflow in  $\exp$
  - Underflow in  $\exp$  causing  $-\infty$  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  
梯度被错误地“消掉”了（例如某些地方梯度始终为 0）
- 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