Basic Math

HKUST MSBD 6000B

Instructor: Yu Zhang

Outline

- Linear Algebra
- Probability
- Information Theory
- Numerical Computation

Linear Algebra

- Basic knowledge of linear algebra for deep learning

- A useful sheet: Matrix Cookbook

Linear Algebra: Scalars

- A scalar is just a single number
- In lower-case variables

Linear Algebra: Vectors

- A vector is an array of numbers.
- The numbers are arranged in order.
- We give vectors lower case names written in bold typeface
  - $\boldsymbol{x}$
- We can identify each individual number by its index.
  - The first element of $\boldsymbol{x}$ is x_1, the second element is x_2 and so on
- Denote a vector $\boldsymbol{x}$ having n numbers by $\boldsymbol{x} \in \mathbb{R}^{\mathbf{n}}$

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

Linear Algebra: Vectors

- Need to index a set of elements of a vector.
- Define a set containing the indices and write the set as a subscript
  - Define a set $S = \{1, 3, 6\}$ and write $\boldsymbol{x}_S$ to access x_1, x_3 and x_6.
- Use the ‘-’sign to index the complement of a set
- $\boldsymbol{x}_{-1}$ is the vector containing all elements of $\boldsymbol{x}$ except for x_1
- $\boldsymbol{x}_{-S}$ is the vector containing all of the elements of $\boldsymbol{x}$ except for x_1, x_3 and x_6

Linear Algebra: Matrices

- A matrix is a 2-D array of numbers
- Each element is identified by two indices
- Give matrices upper-case variable names with bold typeface (e.g., $\boldsymbol{A}$)
- For a real-valued matrix $\boldsymbol{A}$ has a height of m and a width of n, then $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
- $A_{1,1}$ is the upper left entry of $\boldsymbol{A}$
- $A_{m,n}$ is the bottom right entry
- $\boldsymbol{A}_{i,:}$denotes the i-th row of $\boldsymbol{A}$
- $\boldsymbol{A}_{:,j}$ denotes the j-th row of $\boldsymbol{A}$
- By applying the function f to $\boldsymbol{A}$, $f(\boldsymbol{A})_{i,j}$ gives the (i, j)-th element of the matrix computed.

Linear Algebra: Tensors

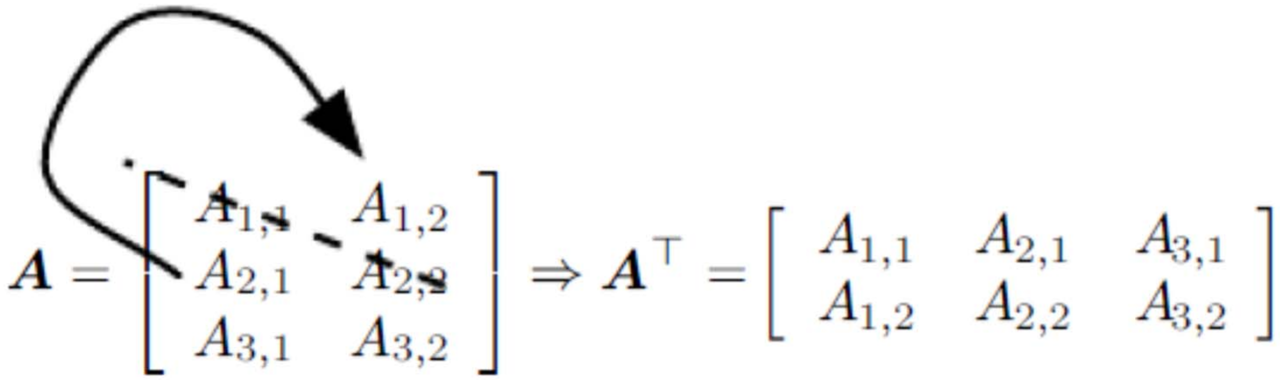
- An array of numbers arranged on a regular grid with a variable number of axes is known as a tensor

- We denote a tensor named “A” by **A**.

- For a tensor **A** with three axes**,** the value at coordinates (i, j, k) is denoted by $A_{i,j,k}$.

Linear Algebra: Transpose

- The transpose of a matrix is the mirror image of the matrix across a diagonal line, called the **main diagonal**, running down and to the right, starting from its upper left corner.



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

- The transpose of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^T$ which satisfies $(\boldsymbol{A}^T)_{i,j} = A_{j,i}$

Linear Algebra: Transpose

- Vectors can be thought of as matrices that contain only one column.
- The transpose of a vector is therefore a matrix with only one row.
- A vector can be the transpose of a row vector $\boldsymbol{x} = [x_1, x_2, x_3]^\top$
- A scalar can be thought of as a matrix with only a single entry.
- A scalar is its own transpose: $a = a^\top$.

Linear Algebra: Addition

- For two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with the same size, we can add them: $\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{B}$, where $C_{i,j}=A_{i,j}+\mathrm{B}_{i,j}$

- Add a scalar to a matrix or multiply a matrix by a scalar: $\boldsymbol{D}=a\boldsymbol{B}+c$, where $D_{i,j}=a*B_{i,j}+c$

- Allow the addition of matrix and a vector, yielding another matrix: $\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{b}$, where $C_{i,j}=A_{i,j}+\mathrm{b}_j$
  - the vector b is added to each row of the matrix $\boldsymbol{A}$
  - This implicit copying of b to many locations is called **broadcasting**.

Linear Algebra: Multiplication

- The **matrix product** of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is a third matrix $\boldsymbol{C}$.
- In order for this product to be defined, $\boldsymbol{A}$ must have the same number of columns as $\boldsymbol{B}$ has rows.
- If $\boldsymbol{A}$ is of shape $m \times n$ and $\boldsymbol{B}$ is of shape $n \times p$, then $\boldsymbol{C}$ is of shape $m \times p$.
- $\boldsymbol{C} = \boldsymbol{AB}$ $$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$
- The dot product between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same dimensionality is the matrix product $\boldsymbol{x}^\top \boldsymbol{y}$.
- the matrix product $\boldsymbol{C} = \boldsymbol{AB}$ is to computing $C_{i,j}$ as the dot product between row i of $\boldsymbol{A}$ and column j of $\boldsymbol{B}$.

Linear Algebra: Multiplication

- Matrix multiplication is distributive: $\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{AB} + \boldsymbol{AC}$
- Matrix multiplication is associative: $\boldsymbol{A}(\boldsymbol{BC}) = (\boldsymbol{AB})\boldsymbol{C}$
- Matrix multiplication is *not* commutative: $\boldsymbol{AB} = \boldsymbol{BA}$ does not always hold
- The dot product between two vectors is commutative: $\boldsymbol{x}^T\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{x}$.
- The transpose of a matrix product satisfies: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$

Linear Algebra: Linear System

- A system of linear equations: $\boldsymbol{Ax} = \boldsymbol{b}$
- $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a known matrix, $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{m}}$ is a known vector, and $\boldsymbol{x} \in \mathbb{R}^{\mathrm{n}}$ is a vector of unknown variables

$$\begin{aligned} \boldsymbol{A}_{1,:}\boldsymbol{x} &= b_1 \\ \boldsymbol{A}_{2,:}\boldsymbol{x} &= b_2 \\ &\cdots \\ \boldsymbol{A}_{m,:}\boldsymbol{x} &= b_m \end{aligned}$$

$$\begin{aligned} \boldsymbol{A}_{1,1}x_1 + \boldsymbol{A}_{1,2}x_2 + \cdots + \boldsymbol{A}_{1,n}x_n &= b_1 \\ \boldsymbol{A}_{2,1}x_1 + \boldsymbol{A}_{2,2}x_2 + \cdots + \boldsymbol{A}_{2,n}x_n &= b_2 \\ &\cdots \\ \boldsymbol{A}_{m,1}x_1 + \boldsymbol{A}_{m,2}x_2 + \cdots + \boldsymbol{A}_{m,n}x_n &= b_m \end{aligned}$$

Linear Algebra: Identity Matrix

- An identity matrix is a matrix that does not change any vector when we multiply that vector by that matrix.
- The identity matrix that preserves *n*-dimensional vectors is denoted by $\boldsymbol{I}_n$
- $\boldsymbol{I_n} \in \mathbb{R}^{n\times n}$, and $\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{I_n x} = \boldsymbol{x}$.
- All of the entries along the main diagonal are 1, while all of the other entries are zero.

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

Linear Algebra: Inverse Matrix

- The **matrix inverse** of $\boldsymbol{A}$ is denoted as $\boldsymbol{A}^{-1}$.
- $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$
- We can solve the linear system as

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

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

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

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

Linear Dependence and Span

- linear combination: $\boldsymbol{A}\boldsymbol{x} = \sum_i x_i \boldsymbol{A}_{:,i}$

- A linear combination of some set of vectors $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}\}$ is given by multiplying each vector $\boldsymbol{v}^{(i)}$ by a corresponding scalar coefficient and adding the results:

$$\sum_i c_i \boldsymbol{v}^{(i)}$$

- The **span** of a set of vectors is the set of all points obtainable by linear combination of the original vectors.

Linear Dependence and Span

- Determining whether $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution thus amounts to testing whether $\boldsymbol{b}$ is in the span of the columns of $\boldsymbol{A}$.
- This particular span is known as the column space or the range of A.
- In order for the system $\boldsymbol{Ax} = \boldsymbol{b}$ to have a solution for all values of $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{m}}$, we require that the column space of A be all of $\mathbb{R}^{\boldsymbol{m}}$.
- Implies that $\boldsymbol{A}$ must have at least m columns, i.e., $n \geqslant m$.
  - a necessary condition for every point to have a solution.
  - It is not a sufficient condition, because it is possible for some of the columns to be redundant.
  - this kind of redundancy is known as **linear dependence**

Linear Dependence and Span

- A set of vectors is **linearly independent** if no vector in the set is a linear combination of the other vectors.
- If we add a vector to a set, the new vector does not add any points to the set's span.
- This means that for the column space of the matrix to encompass all of $\mathbb{R}^{\boldsymbol{m}}$
- The matrix must contain at least one set of *m* linearly independent columns.
- This condition is both necessary and sufficient for linear system $\boldsymbol{Ax}=\boldsymbol{b}$ to have a solution for every value of $\boldsymbol{b}$.
- No set of *m*-dimensional vectors can have more than *m* mutually linearly independent columns

Linear Dependence and Span

- In order for the matrix to have an inverse, this means that the matrix must be **square**.
- We require that $m = n$ and that all of the columns must be linearly independent.
- A square matrix with linearly dependent columns is known as **singular**.
- If $\boldsymbol{A}$ is not square or is square but singular, it can still be possible to solve the equation.
- However, we can not use the method of matrix inversion to find the
- Matrix inverses are multiplied on the left.
- It is also possible to define an inverse that is multiplied on the right: $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$.
- For square matrices, the left inverse and right inverse are equal.

Linear Algebra: Norms

- Sometimes we need to measure the size of a vector.
- In machine learning, we usually measure the size of vectors using a function called a **norm**.
- The L^p norm is given by $$||\boldsymbol{x}||_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$
- $p \geq 1$
- Norms, including the L^p norm, are functions mapping vectors to non-negative values.
- The norm of a vector ***x*** measures the distance from the origin to the point ***x***.

Linear Algebra: Norms

- a norm is any function f that satisfies the following properties:
  - $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \mathbf{0}$
  - $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (the **triangle inequality**)
  - $\forall \alpha \in \mathbb{R}, f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$

- The L^2 norm, with $p = 2$, is known as the **Euclidean norm**.
- It is simply the Euclidean distance from the origin to the point identified by $\boldsymbol{x}$.
- The L^2 norm is used so frequently in machine learning that it is often denoted simply as $||\boldsymbol{x}||$, with the subscript 2 omitted.
- It is common to measure the size of a vector using the squared L^2 norm, which can be calculated simply as $\boldsymbol{x}^T\boldsymbol{x}$.

Linear Algebra: Norms

- The squared L^2 norm is more convenient to work with mathematically and computationally than the L^2 norm itself.
  - For example, the derivatives of the squared L^2 norm with respect to each element of x each depend only on the corresponding element of x,
  - while all of the derivatives of the L^2 norm depend on the entire vector.

- In many contexts, the squared L^2 norm may be undesirable because it increases very slowly near the origin.

Linear Algebra: Norms

- In several machine learning applications, it is important to discriminate between elements that are exactly zero and elements that are small but nonzero.
- In these cases, we turn to the L^1 norm. $$||\boldsymbol{x}||_1 = \sum_i |x_i|$$
- The L^1 norm is commonly used in machine learning when the difference between zero and nonzero elements is very important.
- Every time an element of $\boldsymbol{x}$ moves away from 0 by ϵ, the L^1 norm increases by ϵ.

Linear Algebra: Norms

- We sometimes measure the size of the vector by counting its number of nonzero elements.
- Some authors refer to this function as the "L^0 norm"
- This is an incorrect terminology.
- The number of non-zero entries in a vector is not a norm
  - Because scaling the vector by α does not change the number of nonzero entries.
- The L^1 norm is often used as a substitute for the number of nonzero entries.

Linear Algebra: Norms

- One other norm that commonly arises in machine learning is the L^∞ norm, also known as the **max norm**.

- This norm simplifies to the absolute value of the element with the largest magnitude in the vector

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

Linear Algebra: Norms

- Sometimes we may also wish to measure the size of a matrix.
- In the context of deep learning, the most common way to do this is with the **Frobenius norm**:

$$||A||_F = \sqrt{\sum_{i,j} A_{i,j}^2}$$

- It is analogous to the L^2 norm of a vector

Linear Algebra: Diagonal Matrices

- **Diagonal** matrices consist mostly of zeros and have non-zero entries only along the main diagonal.
- Formally, a matrix $\boldsymbol{D}$ is diagonal if and only if $D_{i,j} = 0$ for all $i \neq j$.
- The identity matrix is a diagonal matrix, where all of the diagonal entries are 1.
- We write diag($\boldsymbol{v}$) to denote a square diagonal matrix whose diagonal entries are given by the entries of the vector $\boldsymbol{v}$.

Linear Algebra: Diagonal Matrices

- Diagonal matrices are of interest in part because multiplying by a diagonal matrix is very computationally efficient.
- To compute diag($\boldsymbol{v}$)$\boldsymbol{x}$, we only need to scale each element x_i by v_i. In other words, diag($\boldsymbol{v}$)$\boldsymbol{x}$ = $\boldsymbol{v} \odot \boldsymbol{x}$.
- Inverting a square diagonal matrix is also efficient. The inverse exists only if every diagonal entry is nonzero, and in that case, diag($\boldsymbol{v}$)$^{-1}$ = diag($[1/v_1, \ldots, 1/v_n]^\top$).
- Not all diagonal matrices need be square.
- It is possible to construct a rectangular diagonal matrix.
- Non-square diagonal matrices do not have inverses but it is still possible to multiply by them cheaply.

Linear Algebra: Symmetric Matrices

- A **symmetric** matrix is any matrix that is equal to its own transpose: $\boldsymbol{A} = \boldsymbol{A}^{\top}$.

- Symmetric matrices often arise when the entries are generated by some function of two arguments that does not depend on the order of the arguments.
  - If $\boldsymbol{A}$ is a matrix of distance measurements, with $A_{i,j}$ giving the distance from point i to point j, then $A_{i,j} = A_{j,i}$ because distance functions are symmetric.

Linear Algebra: Special Vectors

- A **unit vector** is a vector with **unit norm**: $||\boldsymbol{x}||_2 = 1$.
- A vector $\boldsymbol{x}$ and a vector $\boldsymbol{y}$ are **orthogonal** to each other if $\boldsymbol{x}^\top \boldsymbol{y} = 0$.
- If both vectors have nonzero norm, this means that they are at a 90 degree angle to each other.
- In $\mathbb{R}^n$, at most n vectors may be mutually orthogonal with nonzero norm.
- If the vectors are not only orthogonal but also have unit norm, we call them **orthonormal**.

Linear Algebra: Orthogonal Matrix

- An **orthogonal matrix** is a square matrix whose rows are mutually orthonormal and whose columns are mutually orthonormal
- $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{I}$.
- $\boldsymbol{A}^{-1} = \boldsymbol{A}^\top$
- Orthogonal matrices are of interest because their inverse is very cheap to compute.
- Their rows are not merely orthogonal but fully orthonormal.

Linear Algebra: Eigendecomposition

- Integers can be decomposed into prime factors
- We can decompose matrices in ways that show us information about their functional properties that is not obvious from the representation of the matrix as an array of elements.
- One of the most widely used kinds of matrix decomposition is called **eigendecomposition**,
- We decompose a matrix into a set of eigenvectors and eigenvalues.
- An **eigenvector** of a square matrix $\boldsymbol{A}$ is a non-zero vector $\boldsymbol{v}$ such that multiplication by $\boldsymbol{A}$ alters only the scale of $\boldsymbol{v}$: $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$.
- The scalar λ is known as the **eigenvalue** corresponding to this eigenvector.

Linear Algebra: Eigendecomposition

- One can also find a **left eigenvector** such that $\boldsymbol{v}^\top \boldsymbol{A} = \lambda \boldsymbol{v}^\top$, but we are usually concerned with right eigenvectors
- If $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{A}$, then so is any rescaled vector $s\boldsymbol{v}$ for a non-zero scalar s.
- $s\boldsymbol{v}$ still has the same eigenvalue. We usually only look for unit eigenvectors.
- Suppose that a matrix $\boldsymbol{A}$ has n linearly independent eigenvectors, $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}\}$, with corresponding eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$.
- Concatenate all of the eigenvectors to form a matrix $\boldsymbol{V}$ with one eigenvector per column: $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}]$
- concatenate the eigenvalues to form a vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^\top$.
- The **eigendecomposition** of $\boldsymbol{A}$ is then given by $\boldsymbol{A} = \boldsymbol{V}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$.

Linear Algebra: Eigendecomposition

- Not every matrix can be decomposed into eigenvalues and eigenvectors.
- In some cases, the decomposition exists, but may involve complex rather than real numbers.
- We usually need to decompose only a specific class of matrices that have a simple decomposition.
- Every real symmetric matrix can be decomposed into an expression using only real-valued eigenvectors and eigenvalues: $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}$
- $\boldsymbol{Q}$ is an orthogonal matrix composed of eigenvectors of $\boldsymbol{A}$
- $\boldsymbol{\Lambda}$ is a diagonal matrix. The eigenvalue $\boldsymbol{\Lambda}_{i,i}$ is associated with the eigenvector in column i of $\boldsymbol{Q}$

Linear Algebra: Eigendecomposition

- While any real symmetric matrix $\boldsymbol{A}$ is guaranteed to have an eigendecomposition, the eigendecomposition may not be unique.
- If any two or more eigenvectors share the same eigenvalue, then any set of orthogonal vectors lying in their span are also eigenvectors with that eigenvalue,
- By convention, we usually sort the entries of $\boldsymbol{\Lambda}$ in descending order.
- Under this convention, the eigendecomposition is unique only if all of the eigenvalues are unique.

Linear Algebra: Eigendecomposition

- The eigendecomposition of a matrix tells us many useful facts about the matrix.
- The matrix is singular if and only if any of the eigenvalues are zero.
- The eigendecomposition of a real symmetric matrix can also be used to optimize quadratic expressions of the form $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$ subject to $||\boldsymbol{x}||_2 = 1$.
- Whenever $\boldsymbol{x}$ is equal to an eigenvector of $\boldsymbol{A}$, f takes on the value of the corresponding eigenvalue.
- The maximum value of f within the constraint region is the maximum eigenvalue
- Its minimum value within the constraint region is the minimum eigenvalue.

Linear Algebra: Eigendecomposition

- A matrix whose eigenvalues are all positive is called **positive definite**
- A matrix whose eigenvalues are all positive or zero-valued is called **positive semidefinite**
- If all eigenvalues are negative, the matrix is **negative definite**
- If all eigenvalues are negative or zero-valued, it is **negative semidefinite**.
- Positive semidefinite matrices are interesting because they guarantee that $\forall \boldsymbol{x}, \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} \geqslant 0$.
- Positive definite matrices additionally guarantee that $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = 0 \Rightarrow \boldsymbol{x} = \mathbf{0}$.

Linear Algebra: Singular Value Decomposition

- The **singular value decomposition** (SVD) provides another way to factorize a matrix, into **singular vectors** and **singular values**.
- The SVD allows us to discover some of the same kind of information as the eigendecomposition.
- The SVD is more generally applicable. Every real matrix has a singular value decomposition
  - If a matrix is not square, the eigendecomposition is not defined, and we must use a singular value decomposition instead.

Linear Algebra: Singular Value Decomposition

- The singular value decomposition is to write $\boldsymbol{A}$ as a product of three matrices: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}$.
- Suppose that $\boldsymbol{A}$ is an $m \times n$ matrix. Then $\boldsymbol{U}$ is defined to be an $m \times m$ matrix, $\boldsymbol{D}$ to be an $m \times n$ matrix, and $\boldsymbol{V}$ to be an $n \times n$ matrix.
- The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are both defined to be orthogonal matrices. The matrix $\boldsymbol{D}$ is defined to be a diagonal matrix.
  - Note that $\boldsymbol{D}$ is not necessarily square.
- The elements along the diagonal of $\boldsymbol{D}$ are known as the **singular values** of the matrix $\boldsymbol{A}$.
- The columns of $\boldsymbol{U}$ are known as the **left-singular vectors**.
- The columns of $\boldsymbol{V}$ are known as as the **right-singular vectors**.

Linear Algebra: Singular Value Decomposition

- We can actually interpret the singular value decomposition of $\boldsymbol{A}$ in terms of the eigendecomposition of functions of $\boldsymbol{A}$.
- The left-singular vectors of $\boldsymbol{A}$ are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$.
- The right-singular vectors of $\boldsymbol{A}$ are the eigenvectors of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$.
- The non-zero singular values of $\boldsymbol{A}$ are the square roots of the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ or $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$.

Linear Algebra: Pseudoinverse

- Matrix inversion is not defined for matrices that are not square.
- Suppose we want to make a left-inverse ***B*** of a matrix ***A***
- $\boldsymbol{Ax} = \boldsymbol{y}$ becomes $\boldsymbol{x} = \boldsymbol{By}$.
- The **Moore-Penrose pseudoinverse** allows us to make some headway in these cases.
- The pseudoinverse of *A* is defined as: $\boldsymbol{A}^{+} = \lim_{\alpha \searrow 0} (\boldsymbol{A}^{\top}\boldsymbol{A} + \alpha \boldsymbol{I})^{-1}\boldsymbol{A}^{\top}$
- Practical algorithms for computing the pseudoinverse are based on the formula: $\boldsymbol{A}^{+} = \boldsymbol{V}\boldsymbol{D}^{+}\boldsymbol{U}^{\top}$
  - ***U***, ***D*** and ***V*** are the singular value decomposition of ***A***
  - The pseudoinverse $\boldsymbol{D}^{+}$ of a diagonal matrix ***D*** is obtained by taking the reciprocal of its non-zero elements then taking the transpose of the resulting matrix.

Linear Algebra: Pseudoinverse

- When $\boldsymbol{A}$ has more columns than rows, then solving a linear equation using the pseudoinverse provides one of the many possible solutions.
- Specifically, it provides the solution $\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{y}$ with minimal Euclidean norm $||\boldsymbol{x}||_2$ among all possible solutions.

- When $\boldsymbol{A}$ has more rows than columns, it is possible for there to be no solution.
- In this case, using the pseudoinverse gives us the $\boldsymbol{x}$ for which $\boldsymbol{Ax}$ is as close as possible to $\boldsymbol{y}$ in terms of Euclidean norm $||\boldsymbol{Ax} - \boldsymbol{y}||_2$.

Linear Algebra: Trace

- The trace operator gives the sum of all of the diagonal entries of a matrix

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

- Some operations that are difficult to specify without resorting to summation notation can be specified using matrix products and the trace operator.
- The trace operator provides an alternative way of writing the Frobenius norm of a matrix:

$$||A||_F = \sqrt{\mathrm{Tr}(\boldsymbol{A}\boldsymbol{A}^\top)}$$

Linear Algebra: Trace

- Writing an expression in terms of the trace operator opens up opportunities to manipulate the expression using many useful identities.
- The trace operator is invariant to the transpose operator: Tr($\boldsymbol{A}$) = Tr($\boldsymbol{A}^\top$).
- The trace of a square matrix composed of many factors is also invariant to moving the last factor into the first position: Tr($\boldsymbol{ABC}$) = Tr($\boldsymbol{CAB}$) = Tr($\boldsymbol{BCA}$)

$$\mathrm{Tr}(\prod_{i=1}^{n} \boldsymbol{F}^{(i)}) = \mathrm{Tr}(\boldsymbol{F}^{(n)} \prod_{i=1}^{n-1} \boldsymbol{F}^{(i)})$$

- This invariance to cyclic permutation holds even if the resulting product has a different shape.
- For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, we have Tr($\boldsymbol{AB}$) = Tr($\boldsymbol{BA}$).
- A scalar is its own trace: a = Tr(a).

Linear Algebra: Determinant

- The determinant of a square matrix, denoted det(***A***), is a function mapping matrices to real scalars.
- The determinant is equal to the product of all the eigenvalues of the matrix.
- The absolute value of the determinant can be thought of as a measure of how much multiplication by the matrix expands or contracts space.
- If the determinant is 0, then space is contracted completely along at least one dimension, causing it to lose all of its volume.
- If the determinant is 1, then the transformation preserves volume.

Algebra: Principal Components Analysis

- Suppose we have a collection of m points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ in $\mathbb{R}^n$.
- Suppose we would like to apply lossy compression to these points.
- Lossy compression means storing the points in a way that requires less memory but may lose some precision.
- We would like to lose as little precision as possible.
- One way we can encode these points is to represent a lower-dimensional version of them.
- For each point $\boldsymbol{x}^{(i)}$, we will find a corresponding code vector $\boldsymbol{c}^{(i)} \in \mathbb{R}^l$.
- If l is smaller than n, it will take less memory to store the code points than the original data.
- We will want to find some encoding function that produces the code for an input, $f(\boldsymbol{x}) = \boldsymbol{c}$, and a decoding function that produces the reconstructed input given its code, $\boldsymbol{x} \approx g(f(\boldsymbol{x}))$.

Algebra: Principal Components Analysis

- PCA is defined by our choice of the decoding function.
- Specifically, to make the decoder very simple, we choose to use matrix multiplication to map the code back into $\mathbb{R}^n$.
- Let $g(\boldsymbol{c}) = \boldsymbol{Dc}$, where $\boldsymbol{D} \in \mathbb{R}^{n \times l}$ is the matrix defining the decoding.
- To keep the encoding problem easy, PCA constrains the columns of $\boldsymbol{D}$ to be orthogonal to each other and all of the columns of $\boldsymbol{D}$ to have unit norm.
- The first thing we need to do is figure out how to generate the optimal code point $\boldsymbol{c}^*$ for each input point $\boldsymbol{x}$.

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} ||\boldsymbol{x} - g(\boldsymbol{c})||_2^2$$

Algebra: Principal Components Analysis

$$
\begin{aligned}
& ||\boldsymbol{x} - g(\boldsymbol{c})||_2^2 \\
&= (\boldsymbol{x} - g(\boldsymbol{c}))^\top (\boldsymbol{x} - g(\boldsymbol{c})) \\
&= \boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top g(\boldsymbol{c}) - g(\boldsymbol{c})^\top \boldsymbol{x} + g(\boldsymbol{c})^\top g(\boldsymbol{c}) \\
&= \boldsymbol{x}^\top \boldsymbol{x} - 2\boldsymbol{x}^\top g(\boldsymbol{c}) + g(\boldsymbol{c})^\top g(\boldsymbol{c})
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{c}^* &= \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top g(\boldsymbol{c}) + g(\boldsymbol{c})^\top g(\boldsymbol{c}) \\
&= \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{D}^\top \boldsymbol{D}\boldsymbol{c} \\
&= \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{I}_l \boldsymbol{c} \\
&= \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{c}
\end{aligned}
$$

$$\nabla_{\boldsymbol{c}}(-2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{c}) = \boldsymbol{0}$$

$$-2\boldsymbol{D}^\top \boldsymbol{x} + 2\boldsymbol{c} = \boldsymbol{0}$$

$$\boldsymbol{c} = \boldsymbol{D}^\top \boldsymbol{x}.$$

Algebra: Principal Components Analysis

- To encode a vector, we apply the encoder function: $f(\boldsymbol{x}) = \boldsymbol{D}^\top \boldsymbol{x}$.
- We can define the PCA reconstruction operation: $r(\boldsymbol{x}) = g(f(\boldsymbol{x})) = \boldsymbol{D}\boldsymbol{D}^\top \boldsymbol{x}$.
- Since we will use the same matrix $\boldsymbol{D}$ to decode all of the points, we must minimize the Frobenius norm of the matrix of errors computed over all dimensions and all points

$$\boldsymbol{D}^* = \arg\min_{\boldsymbol{D}} \sqrt{\sum_{i,j} \left(x_j^{(i)} - r(\boldsymbol{x}^{(i)})_j \right)^2} \text{ subject to } \boldsymbol{D}^\top \boldsymbol{D} = \boldsymbol{I}_l$$

- To derive the algorithm for finding $\boldsymbol{D}^*$, we will start by considering the case where $l = 1$.

Algebra: Principal Components Analysis

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i ||\boldsymbol{x}^{(i)} - \boldsymbol{d}\boldsymbol{d}^\top \boldsymbol{x}^{(i)}||_2^2 \text{ subject to } ||\boldsymbol{d}||_2 = 1.$$

$$\Downarrow$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i ||\boldsymbol{x}^{(i)} - \boldsymbol{d}^\top \boldsymbol{x}^{(i)} \boldsymbol{d}||_2^2 \text{ subject to } ||\boldsymbol{d}||_2 = 1$$

$$\Downarrow$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i ||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)\top} \boldsymbol{d}\boldsymbol{d}||_2^2 \text{ subject to } ||\boldsymbol{d}||_2 = 1$$

$$\Downarrow$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} ||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top||_F^2 \text{ subject to } \boldsymbol{d}^\top \boldsymbol{d} = 1.$$

Algebra: Principal Components Analysis

$$
\begin{aligned}
&\arg\min_{\boldsymbol{d}} ||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top||_F^2 \\
&= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\right)^\top \left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\right)\right) \\
&= \arg\min_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top - \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X} + \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \\
&= \arg\min_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}) - \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) - \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \\
&= \arg\min_{\boldsymbol{d}} - \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) - \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \\
&= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \\
&= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{d}\boldsymbol{d}^\top)
\end{aligned}
$$

Algebra: Principal Components Analysis

$$
\begin{aligned}
&\arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1 \\
&= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1 \\
&= \arg\min_{\boldsymbol{d}} -\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1 \\
&= \arg\max_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1 \\
&= \arg\max_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{d}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1
\end{aligned}
$$

- This optimization problem can be solved using eigendecomposition.
- Specifically, the optimal $\boldsymbol{d}$ is given by the eigenvector of $\boldsymbol{X}^\top\boldsymbol{X}$ corresponding to the largest eigenvalue.

Algebra: Principal Components Analysis

- This derivation is specific to the case of $l = 1$ and recovers only the first principal component.

- More generally, when we wish to recover a basis of principal components, the matrix $\boldsymbol{D}$ is given by the l eigenvectors corresponding to the largest eigenvalues.

Probability

- Probability theory is a mathematical framework for representing uncertain statements.
- It provides a means of quantifying uncertainty.
- In artificial intelligence applications, we use probability theory in two major ways.
    - First, the laws of probability tell us how AI systems should reason, so we design our algorithms to compute or approximate various expressions derived using probability theory.
    - Second, we can use probability and statistics to theoretically analyze the behavior of proposed AI systems.

Why Probability?

- Machine learning must always deal with uncertain or stochastic (non-deterministic) quantities.
- There are three possible sources of uncertainty:
  - Inherent stochasticity in the system being modeled
  - Incomplete observability
    - Even deterministic systems can appear stochastic when we cannot observe all of the variables that drive the behavior of the system.
  - Incomplete modeling
    - When we use a model that must discard some of the information we have observed, the discarded information results in uncertainty in the model's predictions.

Random Variables

- A **random variable** is a variable that can take on different values randomly.
- Denote the random variable itself with a lower case letter in plain typeface
- Denote the values it can take on with lower case script letters
    - x_1 and x_2 are both possible values that the random variable x can take on.
- For vector-valued variables, we would write the random variable as **x** and one of its values as ***x***.

Random Variables

- A random variable is just a description of the states that are possible.
- It must be coupled with a probability distribution that specifies how likely each of these states are.
- Random variables may be discrete or continuous.
- A discrete random variable is one that has a finite or countably infinite number of states.
- A continuous random variable is associated with a real value.

Probability Distributions

- A **probability distribution** is a description of how likely a random variable or set of random variables is to take on each of its possible states.

- The way to describe probability distributions depends on whether the variables are discrete or continuous.

Discrete Variables

- A probability distribution over discrete variables can be described using a **probability mass function** (PMF).
- Denote probability mass functions with a capital P.
- The probability mass function maps from a state of a random variable to the probability of that random variable taking on that state.
- The probability that $\mathrm{x} = x$ is denoted as $P(x)$
  - a probability of 1 indicating that $\mathrm{x} = x$ is certain
  - a probability of 0 indicating that $\mathrm{x} = x$ is impossible.
- Define a variable first, then use $\sim$ notation to specify which distribution it follows later: $\mathrm{x} \sim P(\mathrm{x})$.

Discrete Variables

- Probability mass functions can act on many variables at the same time.
- Such a probability distribution over many variables is known as a **joint probability distribution**.
- $P(\mathrm{x} = x, \mathrm{y} = y)$ denotes the probability that $\mathrm{x} = x$ and $\mathrm{y} = y$ simultaneously.
- We may also write $P(x, y)$ for brevity.

Discrete Variables

- A probability mass function P must satisfy the following properties:
  - The domain of P must be the set of all possible states of x.
  - $\forall x \in \mathrm{x}, 0 \leqslant P(x) \leqslant 1.$
  - $\sum_{x\in\mathrm{x}} P(x) = 1$

- Consider a single discrete random variable x with k different states.
- We can place a **uniform distribution** on x
- Each of its states is equally likely: $P(\mathrm{x} = x_i) = 1/k$

Continuous Variables

- Probability distributions of continuous random variables are described using a **probability density function (PDF)**.
- To be a probability density function, a function *p* must satisfy the following properties:
- The domain of *p* must be the set of all possible states of x
- $\forall x \in \mathrm{x}, p(x) \geqslant 0.$
- $\int p(x)dx = 1$

Continuous Variables

- We can integrate the density function to find the actual probability mass of a set of points.
- The probability that *x* lies in some set S is given by the integral of *p* (*x*) over that set.
- In the univariate example, the probability that *x* lies in the interval [*a, b*] is given by $\int_{[a,b]} p(x)dx$

Marginal Probability

- We know the probability distribution over a set of variables and we want to know the probability distribution over just a subset of them.
- The probability distribution over the subset is known as the **marginal probability** distribution.
- Suppose we have discrete random variables x and y, and we know *P* (x*,* y). We can find *P* (x) with the **sum rule**:

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

- For continuous variables, we need to use integration instead of summation:

$$p(x) = \int p(x, y) dy$$

Conditional Probability

- In many cases, we are interested in the probability of some event, given that some other event has happened.
- This is called a **conditional probability**.
- Denote the conditional probability that $\mathrm{y} = y$ given $\mathrm{x} = x$ as $P(\mathrm{y} = y \mid \mathrm{x} = x)$.
- $$P(\mathrm{y} = y \mid \mathrm{x} = x) = \frac{P(\mathrm{y} = y, \mathrm{x} = x)}{P(\mathrm{x} = x)}$$
- The conditional probability is only defined when $P(\mathrm{x} = x) > 0$.

The Chain Rule of Conditional Probabilities

- Any joint probability distribution over many random variables can be decomposed into conditional distributions over only one variable:

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

- This observation is known as the **chain rule** or **product rule** of probability.
- It follows immediately from the definition of conditional probability
- $P(\mathrm{a}, \mathrm{b}, \mathrm{c}) = P(\mathrm{a} \mid \mathrm{b}, \mathrm{c})P(\mathrm{b}, \mathrm{c})$
- $P(\mathrm{b}, \mathrm{c}) = P(\mathrm{b} \mid \mathrm{c})P(\mathrm{c})$
- $P(\mathrm{a}, \mathrm{b}, \mathrm{c}) = P(\mathrm{a} \mid \mathrm{b}, \mathrm{c})P(\mathrm{b} \mid \mathrm{c})P(\mathrm{c})$

Independence and Conditional Independence

- Two random variables x and y are **independent** if their probability distribution can be expressed as a product of two factors, one involving only x and one involving only y:

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

- Two random variables x and y are **conditionally independent** given a random variable z if the conditional probability distribution over x and y factorizes in this way for every value of z:

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

- $\mathrm{x} \perp \mathrm{y}$ means that x and y are independent
- $\mathrm{x} \perp \mathrm{y} \mid \mathrm{z}$ means that x and y are conditionally independent given z.

Expectation

- The **expectation** or **expected value** of some function *f*(*x*) with respect to a probability distribution *P* (x) is the average or mean value that *f* takes on when *x* is drawn from *P*.
- For discrete variables, $\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \sum_x P(x)f(x)$
- For continuous variables,

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

- we may simply write $\mathbb{E}_{\mathrm{x}}[f(x)]$ or $\mathbb{E}[f(x)]$
- $\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_{\mathrm{x}}[f(x)] + \beta \mathbb{E}_{\mathrm{x}}[g(x)]$

Variance

- The **variance** gives a measure of how much the values of a function of a random variable x vary as we sample different values of *x* from its probability distribution:

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

- When the variance is low, the values of $f(x)$ cluster near their expected value.
- The square root of the variance is known as the **standard deviation**.

Covariance

- The **covariance** gives some sense of how much two values are linearly related to each other as well as the scale of these variables:

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

- High absolute values of the covariance mean that the values change very much and are both far from their respective means at the same time.
- If the sign of the covariance is positive, then both variables tend to take on relatively high values simultaneously.
- If the sign of the covariance is negative, then one variable tends to take on a relatively high value at the times that the other takes on a relatively low value and vice versa.
- Other measures such as **correlation** normalize the contribution of each variable in order to measure only how much the variables are related

Covariance and Dependence

- The notions of covariance and dependence are related, but are in fact distinct concepts.
- Two variables that are independent have zero covariance
- Two variables that have non-zero covariance are dependent.
- For two variables to have zero covariance, there must be no linear dependence between them.
- Independence excludes nonlinear relationships.

Covariance and Dependence

- It is possible for two variables to be dependent but have zero covariance.
  - suppose we first sample a real number x from a uniform distribution over the interval $[-1, 1]$.
  - We next sample a random variable s which has an equal probability (i.e., ½) to be 1 and -1
  - We can then generate a random variable y by assigning $y = sx$.
  - x and y are not independent, because x completely determines the magnitude of y.
  - However, $\mathrm{Cov}(x, y) = 0$.
- The **covariance matrix** of a random vector $\boldsymbol{x} \in \mathbb{R}^n$ is an $n \times n$ matrix such that $\mathrm{Cov}(\mathbf{x})_{i,j} = \mathrm{Cov}(x_i, x_j)$
- The diagonal elements of the covariance give the variance: $\mathrm{Cov}(x_i, x_i) = \mathrm{Var}(x_i)$.

Bernoulli Distribution

- The **Bernoulli** distribution is a distribution over a single binary random variable.
- It is controlled by a single parameter $\phi \in [0, 1]$, which gives the probability of the random variable being equal to 1.
- It has the following properties:

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

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

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

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

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

Multinoulli Distribution

- The **multinoulli** or **categorical** distribution is a distribution over a single discrete variable with k different states, where k is finite
- The multinoulli distribution is parametrized by a vector $\boldsymbol{p} \in [0, 1]^{k-1}$,
  - p_i gives the probability of the i-th state.
- The final, k-th state's probability is given by $1 - 1^{\top} p$.
  - Note that we must constrain $1^{\top} p \leqslant 1$.
- Multinoulli distributions are often used to refer to distributions over categories of objects
- We do not usually assume that state 1 has numerical value 1, etc.
- For this reason, we do not usually need to compute the expectation or variance of multinoulli-distributed random variables.

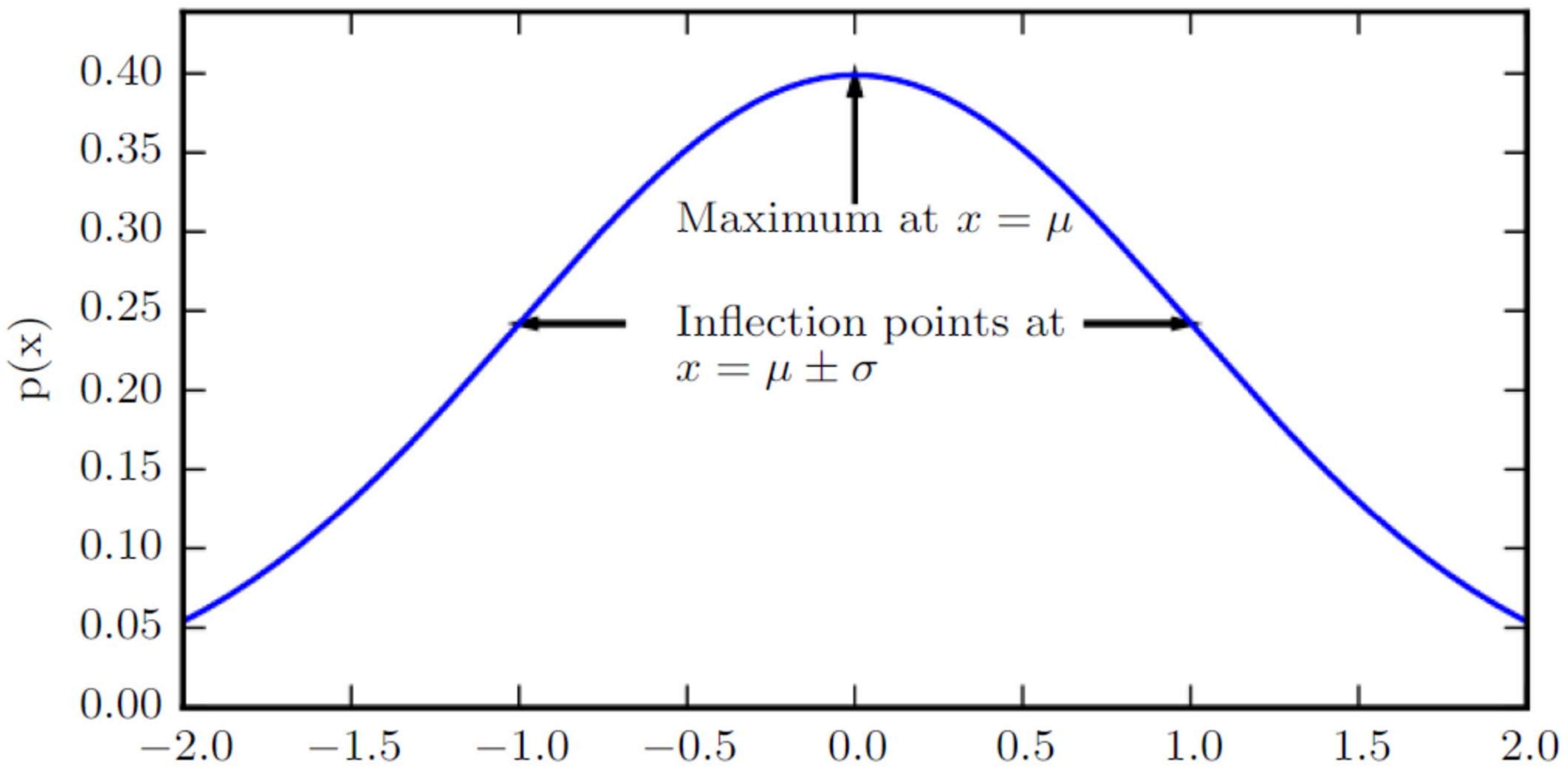
Gaussian Distribution

- The most commonly used distribution over real numbers is the **normal distribution** or **Gaussian distribution**:

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

- The two parameters $\mu \in$ R and $\sigma \in (0, \infty)$ control the normal distribution.
- The parameter μ gives the coordinate of the central peak.
- This is also the mean of the distribution: E[x] = μ.
- The standard deviation of the distribution is given by σ and the variance by σ^2.

Gaussian Distribution



Gaussian Distribution

- When we evaluate the PDF, we need to square and invert σ.
- When we need to frequently evaluate the PDF with different parameter values, a more efficient way of parametrizing the distribution is to use a parameter $\beta \in (0, \infty)$ to control the **precision** or inverse variance of the distribution:

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

Gaussian Distribution

- Normal distributions are a sensible choice for many applications.
- In the absence of prior knowledge, the normal distribution is a good default choice for two major reasons:
  - First, many distributions we wish to model are truly close to being normal distributions.
    - The **central limit theorem** shows that the sum of many independent random variables is approximately normally distributed.
  - Second, out of all possible probability distributions with the same variance, the normal distribution encodes the maximum amount of uncertainty over the real numbers.
    - We can thus think of the normal distribution as being the one that inserts the least amount of prior knowledge into a model.

Gaussian Distribution

- The normal distribution generalizes to $\mathbb{R}^n$, in which case it is known as the **multivariate normal distribution**.
- It can be parametrized with a positive definite symmetric matrix $\boldsymbol{\Sigma}$:

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

- The parameter $\boldsymbol{\mu}$ still gives the mean of the distribution, though now it is vector-valued.
- The parameter $\boldsymbol{\Sigma}$ gives the covariance matrix of the distribution.

Gaussian Distribution

- When we wish to evaluate the PDF several times for many different values of the parameters, the covariance is not a computationally efficient way to parametrize the distribution
- We can instead use a **precision matrix** β:

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

- We often fix the covariance matrix to be a diagonal matrix.
- An even simpler version is the **isotropic** Gaussian distribution, whose covariance matrix is a scalar times the identity matrix.

Exponential and Laplace Distributions

- In the context of deep learning, we often want to have a probability distribution with a sharp point at $x = 0$.
- We can use the **exponential distribution**: $p(x; \lambda) = \lambda \mathbf{1}_{x \geq 0} \exp(-\lambda x)$
  - the indicator function $\mathbf{1}_{x \geqslant 0}$ is used to assign probability zero to all negative values of x.
- A closely related probability distribution that allows us to place a sharp peak of probability mass at an arbitrary point μ is the **Laplace distribution**

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

The Dirac Distribution

- In some cases, we wish to specify that all of the mass in a probability distribution clusters around a single point.
- This can be accomplished by defining a PDF using the Dirac delta function, $\delta(x)$: $p(x) = \delta(x - \mu)$.
- The Dirac delta function is defined such that it is zero-valued everywhere except 0, yet integrates to 1.

The Empirical Distribution

- A common use of the Dirac delta distribution is as a component of an **empirical distribution**

$$\hat{p}(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} \delta(\boldsymbol{x} - \boldsymbol{x}^{(i)})$$

- It puts probability mass $1/m$ on each of the m points $x^{(1)}, \ldots, x^{(m)}$ forming a given dataset or collection of samples.

Mixtures of Distributions

- It is common to define probability distributions by combining other simpler probability distributions.
- One common way of combining distributions is to construct a **mixture distribution**.
- The choice of which component distribution generates the sample is determined by sampling a component identity from a multinoulli distribution:

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

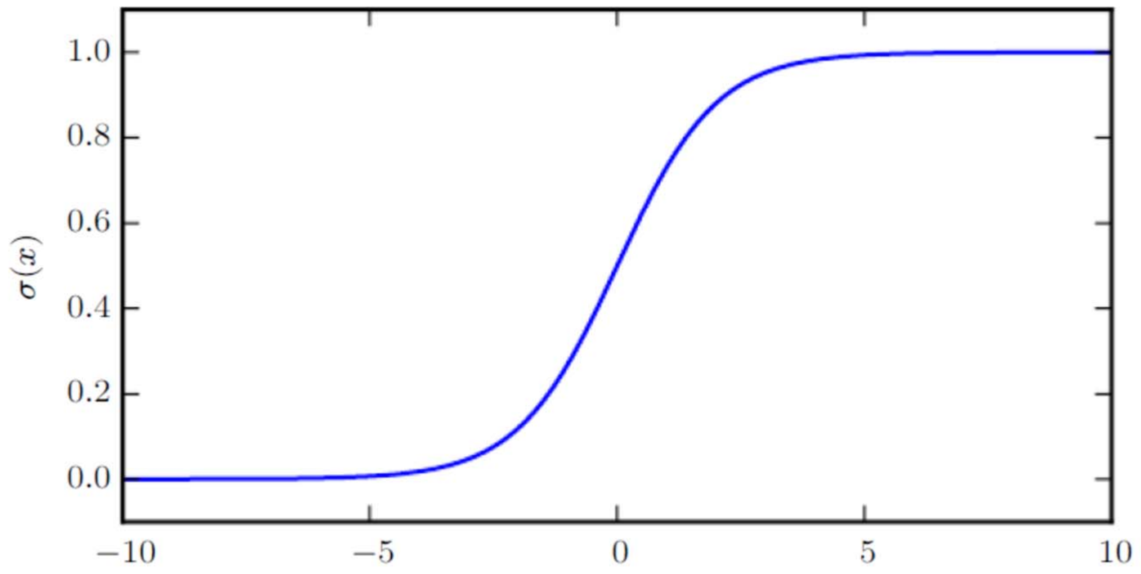
  - *P*(c) is the multinoulli distribution over component identities
  - The empirical distribution over real-valued variables is a mixture distribution with one Dirac component for each training example.

Mixtures of Distributions

- A very powerful and common type of mixture model is the **Gaussian mixture** model
- Each component $p(\mathrm{x} \mid \mathrm{c} = i)$ is a Gaussian distribution parametrized by mean $\boldsymbol{\mu}^{(i)}$ and covariance $\mathbf{\Sigma}^{(i)}$.
- The covariances could be shared across components via the constraint $\mathbf{\Sigma}^{(i)} = \mathbf{\Sigma}$, $\forall i$.
- The parameters of a Gaussian mixture specify the **prior probability** $\alpha_i = P(\mathrm{c} = i)$ given to each component i.
- The word "prior" indicates that it expresses the model's beliefs about c *before* it has observed x.
- $P(\mathrm{c} \mid x)$ is a **posterior probability**, because it is computed *after* observation of x.
- A Gaussian mixture model is a **universal approximator** of densities
  - Any smooth density can be approximated with any specific, non-zero amount of error by a Gaussian mixture model with enough components.

Useful Properties of Common Functions

- Certain functions arise often while working with probability distributions used in deep learning models.
- One of these functions is the **logistic sigmoid**: $\sigma(x) = \frac{1}{1+\exp(-x)}$
- The logistic sigmoid is commonly used to produce the *Φ p*arameter of a Bernoulli distribution because its range is (0, 1)

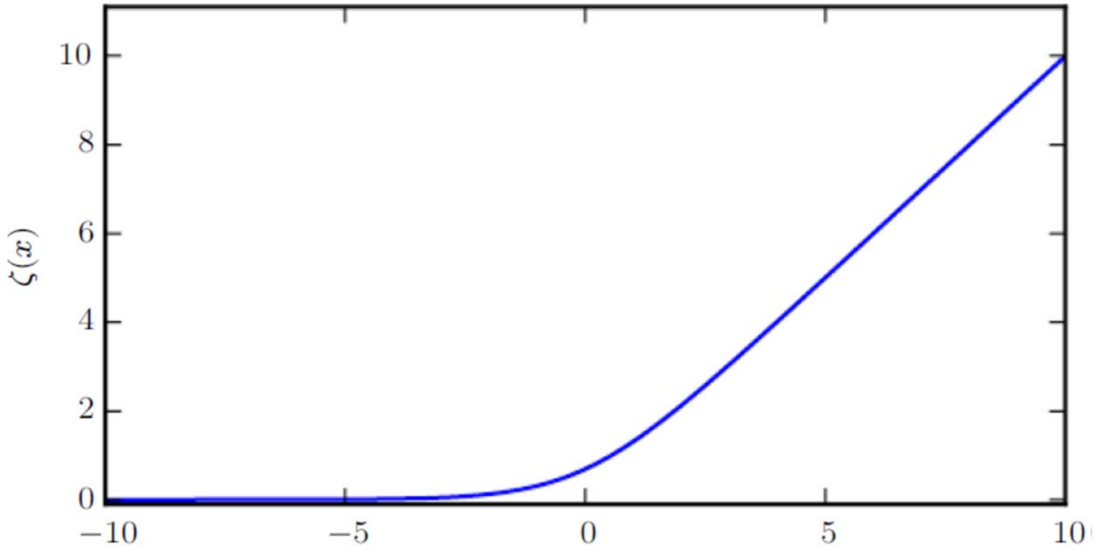


Useful Properties of Common Functions

- Another commonly encountered function is the **softplus** function

$$\zeta(x) = \log\left(1 + \exp(x)\right)$$

- The softplus function can be useful for producing the β or σ parameter of a normal distribution because its range is $(0, \infty)$.
- It is a smoothed or "softened" version of $x^+ = \max(0, x)$.



Useful Properties of Common Functions

$$\sigma(x) = \frac{\exp(x)}{\exp(x) + \exp(0)}$$

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

$$1 - \sigma(x) = \sigma(-x)$$

$$\log \sigma(x) = -\zeta(-x)$$

$$\frac{d}{dx}\zeta(x) = \sigma(x)$$

$$\forall x \in (0,1),\ \sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right)$$

$$\forall x > 0,\ \zeta^{-1}(x) = \log\left(\exp(x) - 1\right)$$

$$\zeta(x) = \int_{-\infty}^{x} \sigma(y)dy$$

$$\zeta(x) - \zeta(-x) = x$$

Bayes' Rule

- We know $P(y \mid x)$ and need to know $P(x \mid y)$.
- If we also know $P(x)$, we can compute the desired quantity using **Bayes' rule**:

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

- It is usually feasible to compute $P(\mathrm{y}) = \sum_x P(\mathrm{y} \mid x)P(x)$

Information Theory

- Information theory is a branch of applied mathematics that revolves around quantifying how much information is present in a signal.
- It was originally invented to study sending messages from discrete alphabets over a noisy channel, such as communication via radio transmission.
- Information theory tells how to design optimal codes and calculate the expected length of messages sampled from specific probability distributions using various encoding schemes.
- In the context of machine learning, we can also apply information theory to continuous variables where some of these message length interpretations do not apply.

Information Theory

- The basic intuition: learning that an unlikely event has occurred is more informative than learning that a likely event has occurred.
- This intuition can be formalized as
  - Likely events should have low information content, and in the extreme case, events that are guaranteed to happen should have no information content whatsoever.
  - Less likely events should have higher information content.
  - Independent events should have additive information.
    - For example, finding out that a tossed coin has come up as heads twice should convey twice as much information as finding out that a tossed coin has come up as heads once.

Information Theory

- In order to satisfy all three of these properties, we define the **self-information** of an event $\mathrm{x} = x$ to be $I(x) = -\log P(x)$.
  - use log to mean the natural logarithm, with base e.
- The definition of $I(x)$ is therefore written in units of **nats**.
  - One nat is the amount of information gained by observing an event of probability $1/e$
- Self-information deals only with a single outcome.
- We can quantify the amount of uncertainty in an entire probability distribution using the **Shannon entropy**:

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

- The Shannon entropy of a distribution is the expected amount of information in an event drawn from that distribution.

Information Theory

- Distributions that are nearly deterministic (where the outcome is nearly certain) have low entropy
- Distributions that are closer to uniform have high entropy.
- If we have two separate probability distributions P(x) and Q(x) over the same random variable x, we can measure how different these two distributions are using the **Kullback-Leibler (KL) divergence**:

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

Information Theory

- The KL divergence has many useful properties
- The KL divergence is nonnegative.
- The KL divergence is 0 if and only if P and Q are the same
- It is often conceptualized as measuring some sort of distance between these distributions.
- It is not a true distance measure because it is not symmetric:

  $D_{\text{KL}}(P\|Q) \neq D_{\text{KL}}(Q\|P)$ for some P and Q.
- This asymmetry means that there are important consequences to the choice of whether to use $D_{\text{KL}}(P\|Q)$ or $D_{\text{KL}}(Q\|P)$.

Information Theory

- A quantity that is closely related to the KL divergence is the **cross-entropy** $H(P, Q) = -\mathbb{E}_{\mathrm{x} \sim P} \log Q(x)$
- $H(P, Q) = H(P) + D_{\mathrm{KL}}(P \| Q)$
- Minimizing the cross-entropy with respect to Q is equivalent to minimizing the KL divergence.
- When computing many of these quantities, it is common to encounter expressions of the form 0 log 0.
- By convention, in the context of information theory, we treat these expressions as $\lim_{x \to 0} x \log x = 0$.

Information Theory: Structured Probabilistic Models

- Machine learning algorithms often involve probability distributions over a very large number of random variables.
- Often, these probability distributions involve direct interactions between relatively few variables.
- Using a single function to describe the entire joint probability distribution can be very inefficient
- Instead of using a single function to represent a probability distribution, we can split a probability distribution into many factors that we multiply together.
  - Suppose we have three random variables: a, b and c.
  - a influences the value of b
  - b influences the value of c
  - a and c are independent given b.
  - We can represent the probability distribution: $p(\mathrm{a}, \mathrm{b}, \mathrm{c}) = p(\mathrm{a})p(\mathrm{b} \mid \mathrm{a})p(\mathrm{c} \mid \mathrm{b})$.
- These factorizations can greatly reduce the number of parameters needed to describe the distribution.

Information Theory: Structured Probabilistic Models

- We can describe these kinds of factorizations using graphs.

- Here we use the word “graph” in the sense of graph theory: a set of vertices that may be connected to each other with edges.

- When we represent the factorization of a probability distribution with a graph, we call it a **structured probabilistic model** or **graphical model**.

Information Theory: Structured Probabilistic Models

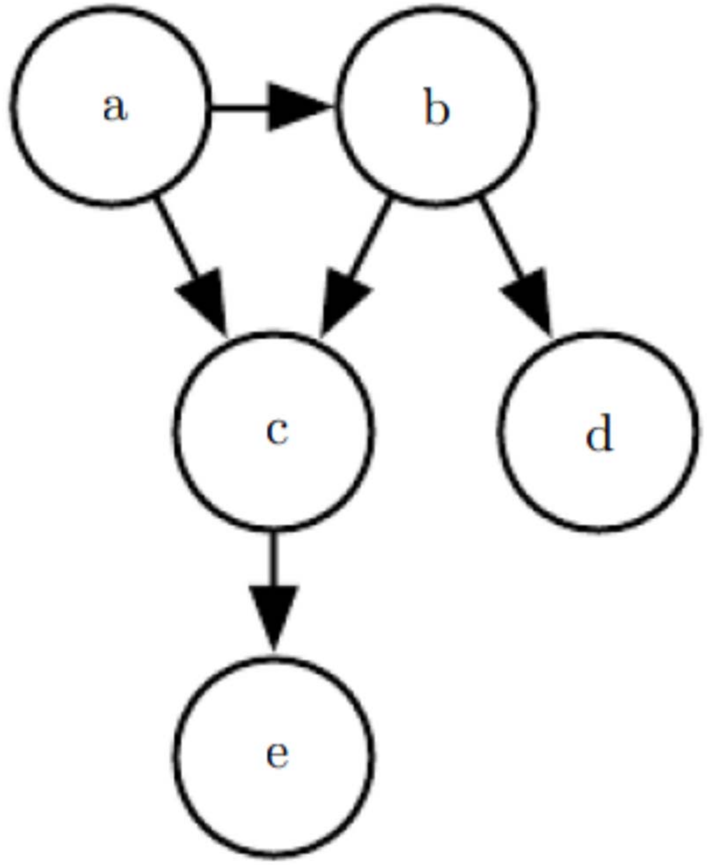
- There are two main kinds of structured probabilistic models: directed and undirected.
- Both kinds of graphical models use a graph G
- Each node in the graph corresponds to a random variable
- An edge connecting two random variables means that the probability distribution is able to represent direct interactions between those two random variables.

Information Theory: Structured Probabilistic Models

- **Directed** models use graphs with directed edges
- They represent factorizations into conditional probability distributions
- A directed model contains one factor for every random variable x$_i$ in the distribution
- That factor consists of the conditional distribution over x$_i$ given the parents of x$_i$, denoted by $Pa_{\mathcal{G}}(\mathrm{x}_i)$:

$$p(\mathbf{x}) = \prod_i p\left(\mathrm{x}_i \mid Pa_{\mathcal{G}}(\mathrm{x}_i)\right)$$

Information Theory: Structured Probabilistic Models



$$p(\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}) = p(\mathrm{a})p(\mathrm{b} \mid \mathrm{a})p(\mathrm{c} \mid \mathrm{a}, \mathrm{b})p(\mathrm{d} \mid \mathrm{b})p(\mathrm{e} \mid \mathrm{c})$$

Information Theory: Structured Probabilistic Models

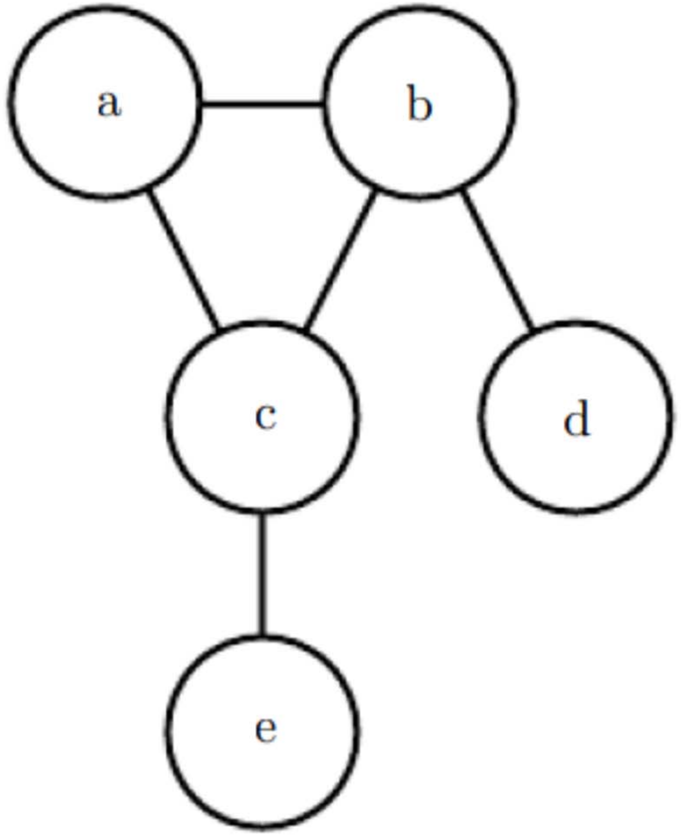
- **Undirected** models use graphs with undirected edges
- They represent factorizations into a set of functions which are usually not probability distributions of any kind.
- Any set of nodes that are all connected to each other in G is called a clique.
- Each clique $C^{(i)}$ in an undirected model is associated with a factor $\Phi^{(i)}(C^{(i)})$.
- These factors are just functions, not probability distributions.
- The output of each factor must be non-negative
- There is no constraint that the factor must sum or integrate to 1 like a probability distribution.

Information Theory: Structured Probabilistic Models

- The probability of a configuration of random variables is **proportional** to the product of all of these factors
- Define a normalizing constant *Z* to be the sum or integral over all states of the product of the *Φ* functions
- We divide by *Z* to obtain a normalized probability distribution:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_i \phi^{(i)} \left(\mathcal{C}^{(i)} \right)$$

Information Theory: Structured Probabilistic Models



$$p(\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}) = \frac{1}{Z}\phi^{(1)}(\mathrm{a}, \mathrm{b}, \mathrm{c})\phi^{(2)}(\mathrm{b}, \mathrm{d})\phi^{(3)}(\mathrm{c}, \mathrm{e})$$

Numerical Computation

- Machine learning algorithms usually require a high amount of numerical computation.
- This typically refers to algorithms that solve mathematical problems by methods that update estimates of the solution via an iterative process
- Common operations
  - Optimization
    - finding the value of an argument that minimizes or maximizes a function
  - Solving systems of linear equations.

Overflow and Underflow

- The fundamental difficulty in performing continuous math on a digital computer is that we need to represent infinitely many real numbers with a finite number of bit patterns.
- For almost all real numbers, we incur some approximation error when we represent the number in the computer.
- In many cases, this is just rounding error.
- One form of rounding error is **underflow**
- Underflow occurs when numbers near zero are rounded to zero.
- Another highly damaging form of numerical error is **overflow**.
- Overflow occurs when numbers with large magnitude are approximated as ∞ or $-\infty$.

Overflow and Underflow

- One example of a function that must be stabilized against underflow and overflow is the softmax function.
- The softmax function is defined as $\operatorname{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$
- The softmax function is often used to predict the probabilities associated with a multinoulli distribution.

Overflow and Underflow

- Consider what happens when all of the x_i are equal to some constant c.
- Analytically, we can see that all of the outputs should be equal to $1/n$.
- Numerically, this may not occur when c has large magnitude.
  - If c is very negative, then exp(c) will underflow. This means the denominator of the softmax will become 0, so the final result is undefined.
  - When c is very large and positive, exp(c) will overflow, again resulting in the expression as a whole being undefined.
- Both of these difficulties can be resolved by instead evaluating softmax(z) where $z = x - \max_i x_i$.
- Subtracting max$i$ xi results in the largest argument to exp being 0, which rules out the possibility of overflow.
- At least one term in the denominator has a value of 1, which rules out the possibility of underflow in the denominator leading to a division by zero.

Poor Conditioning

- Conditioning refers to how rapidly a function changes with respect to small changes in its inputs.
- Functions that change rapidly when their inputs are perturbed slightly can be problematic for scientific computation
  - rounding errors in the inputs can result in large changes in the output.
- Consider the function $f(\boldsymbol{x}) = \boldsymbol{A}^{-1}\boldsymbol{x}$. When $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ has an eigenvalue decomposition, its **condition number** is

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

- This is the ratio of the magnitude of the largest and smallest eigenvalue.
- When this number is large, matrix inversion is particularly sensitive to error in the input.

Gradient-Based Optimization

- Most deep learning algorithms involve optimization of some sort.
- Optimization refers to the task of either minimizing or maximizing some function $f(x)$ by altering x.
- We usually phrase most optimization problems in terms of minimizing $f(x)$.
- Maximization may be accomplished via a minimization algorithm by minimizing $-f(x)$.
- The function we want to minimize or maximize is called the **objective function** or **criterion**.
- When we are minimizing it, we may also call it the **cost function**, **loss function**, or **error function**.
- We often denote the value that minimizes or maximizes a function with a superscript $*$: $x^* = \text{argmin}\ f(x)$.

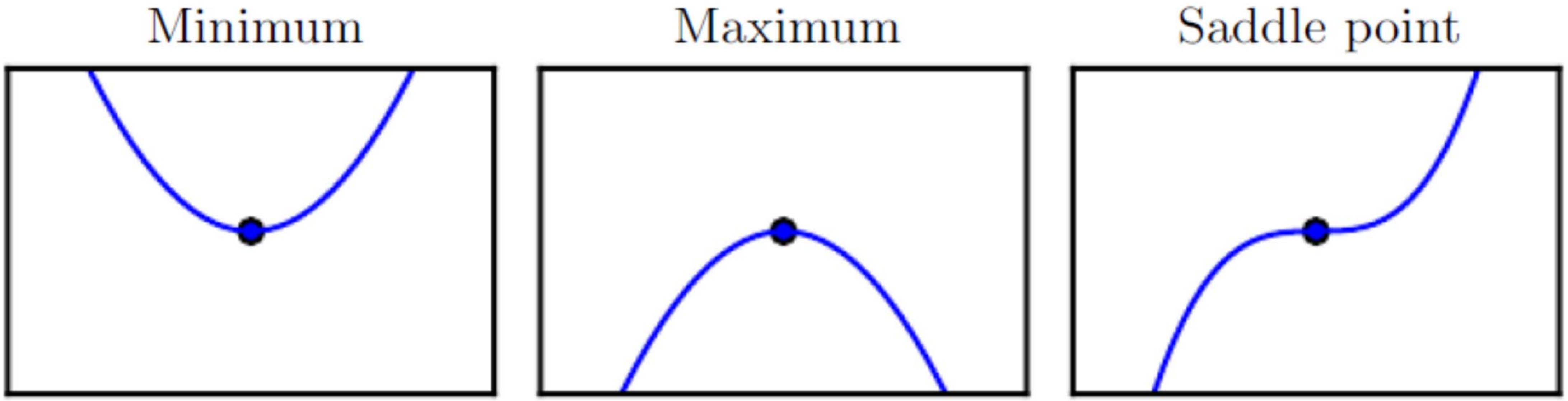
Gradient-Based Optimization

- Suppose we have a function $y = f(x)$, where both x and y are real numbers.
- The **derivative** of this function is denoted as $f'(x)$ or $\frac{dy}{dx}$
- The derivative $f'(x)$ gives the slope of $f(x)$ at the point x.
- It specifies how to scale a small change in the input in order to obtain the corresponding change in the output: $f(x + \varepsilon) \approx f(x) + \varepsilon f'(x)$.
- The derivative is therefore useful for minimizing a function because it tells us how to change x in order to make a small improvement in y.
    - $f(x - \varepsilon\,\mathrm{sign}(f'(x)))$ is less than $f(x)$ for small enough ε
- We can thus reduce $f(x)$ by moving x in small steps with opposite sign of the derivative.
- This technique is called **gradient descent**

Gradient-Based Optimization

- Points where $f'(x) = 0$ are known as **critical points** or **stationary**
- **points**.
- A **local minimum** is a point where $f(x)$ is lower than at all neighboring points.
- A **local maximum** is a point where $f(x)$ is higher than at all neighboring points
- Some critical points are neither maxima nor minima and they are known as **saddle points**

Gradient-Based Optimization



Gradient-Based Optimization

- A point that obtains the absolute lowest value of $f(x)$ is a **global minimum**.
- It is possible for there to be only one global minimum or multiple global minima of the function.
- It is possible that local minima are not globally optimal.
- In the context of deep learning, we optimize functions that may have many local minima and many saddle points
- All of this makes optimization very difficult, especially when the input to the function is multidimensional.
- We usually find a low value of f that is not necessarily minimal in any formal sense.

Gradient-Based Optimization

- We often minimize functions that have multiple inputs: $f: \mathbb{R}^n \rightarrow \mathbb{R}$.
- There must still be only one (scalar) output.
- For functions with multiple inputs, we make use of **partial derivatives**.
- The partial derivative $\frac{\partial f(x)}{\partial x_i}$ measures how f changes as only the variable x_i increases at point x.
- The gradient of f is the vector containing all of the partial derivatives, denoted $\nabla_x f(x)$.
- Critical points are points where every element of the gradient is equal to zero.

Gradient-Based Optimization

- The **directional derivative** in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$.
- The directional derivative is the derivative of the function $f(\boldsymbol{x} + \alpha\boldsymbol{u})$ with respect to α, evaluated at $\alpha = 0$.
- Using the chain rule, we can see that $\frac{\partial}{\partial\alpha} f(\boldsymbol{x} + \alpha\boldsymbol{u})$ equals $\boldsymbol{u}^\top \nabla_x f(\boldsymbol{x})$ when $\alpha = 0$.
- To minimize f, we would like to find the direction in which f decreases the fastest.
- Using the directional derivative:

$$\min_{\boldsymbol{u},\boldsymbol{u}^\top\boldsymbol{u}=1} \boldsymbol{u}^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{u},\boldsymbol{u}^\top\boldsymbol{u}=1} ||\boldsymbol{u}||_2 ||\nabla_{\boldsymbol{x}} f(\boldsymbol{x})||_2 \cos\theta = ||\nabla_{\boldsymbol{x}} f(\boldsymbol{x})||_2 \min_{\boldsymbol{u}} \cos\theta$$

- This is minimized when u points in the opposite direction as the gradient.

Gradient-Based Optimization

- We can decrease f by moving in the direction of the negative gradient.
- This is known as the **method of steepest descent** or **gradient descent**.
- Steepest descent proposes a new point: $\boldsymbol{x}' = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$
- ϵ is the **learning rate**, a positive scalar determining the size of the step.
- We can choose ϵ in several different ways.
    - A popular approach is to set ϵ to a small constant.
    - We can solve for the step size that makes the directional derivative vanish.
    - A **line search** approach is to evaluate $f(\boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}))$ for several values of ϵ and choose the one that results in the smallest objective function value.
- Steepest descent converges when every element of the gradient is zero or, in practice, very close to zero.

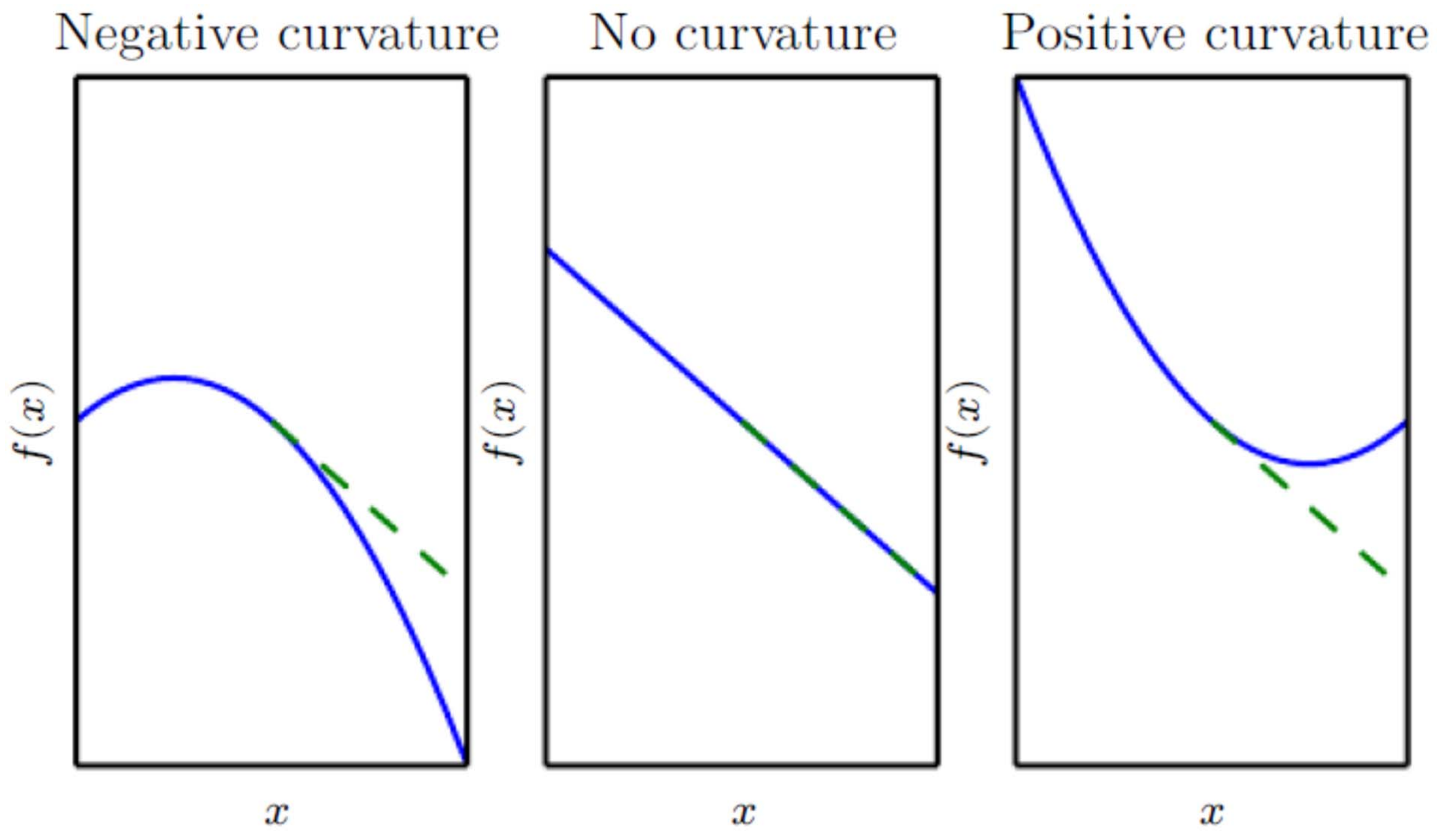
Hessian Matrices

- We are interested in the **second derivative**, a derivative of a derivative.
- For a function $f\colon \mathbb{R}^n \rightarrow \mathbb{R}$, the derivative with respect to x_i of the derivative of f with respect to x_j is denoted by $\frac{\partial^2}{\partial x_i \partial x_i} f(\boldsymbol{x})$.
- In a single dimension, we can denote $\frac{d^2}{dx^2} f(x)$ by $f''(x)$
- The second derivative tells us how the first derivative will change as we vary the input.

Hessian Matrices

- We can think of the second derivative as measuring **curvature**.
- Suppose we have a quadratic function
- If such a function has a second derivative of zero, then there is no curvature.
    - It is a perfectly flat line, and its value can be predicted using only the gradient.
    - If the gradient is 1, then we can make a step of size ϵ along the negative gradient, and the cost function will decrease by ϵ.
- If the second derivative is negative, the function curves downward, so the cost function will actually decrease by more than ϵ.
- If the second derivative is positive, the function curves upward, so the cost function can decrease by less than ϵ.

Hessian Matrices



Hessian Matrices

- When our function has multiple input dimensions, these derivatives can be collected together into a matrix called the **Hessian matrix**.
- The Hessian matrix $H(f)(x)$ is defined such that

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

- If the second partial derivatives are continuous, they are commutative.

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}) = \frac{\partial^2}{\partial x_j \partial x_i} f(\boldsymbol{x})$$

- The Hessian matrix is symmetric

Hessian Matrices

- Most of the functions we encounter in the context of deep learning have a symmetric Hessian almost everywhere.
    - We can decompose it into a set of real eigenvalues and an orthogonal basis of eigenvectors.
- The second derivative in a specific direction represented by a unit vector $\boldsymbol{d}$ is given by $\boldsymbol{d}^{\top}\boldsymbol{H}\boldsymbol{d}$.
- When $\boldsymbol{d}$ is an eigenvector of $\boldsymbol{H}$, the second derivative in that direction is given by the corresponding eigenvalue.
- For other directions of $\boldsymbol{d}$, the directional second derivative is a weighted average of all of the eigenvalues
    - weights are between 0 and 1
    - eigenvectors that have smaller angle with $\boldsymbol{d}$ receiving more weight.
- The maximum eigenvalue determines the maximum second derivative
- The minimum eigenvalue determines the minimum second derivative.

Hessian Matrices

- The second derivative tells us how well we can expect a gradient descent step to perform.
- Make a second-order Taylor series approximation to the function $f(\boldsymbol{x})$ around the current point $\boldsymbol{x}^{(0)}$:

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

  - $\boldsymbol{g}$ is the gradient and $\boldsymbol{H}$ is the Hessian at $\boldsymbol{x}^{(0)}$.
- If we use a learning rate ϵ, then the new point x will be given by $\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}$.
- Substituting into the approximation gives

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

Hessian Matrices

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

- Three terms here: the original value of the function, the expected improvement due to the slope of the function, and the correction to account for the curvature of the function.
- When the last term is too large, the gradient descent step can actually move uphill.
- When $\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$ is zero or negative, the Taylor series approximation predicts that choosing an appropriate ϵ will decrease f
- When $\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$ is positive, solving for the optimal step size that decreases the Taylor series approximation of the function the most yields

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

- When $\boldsymbol{g}$ aligns with the eigenvector of $\boldsymbol{H}$ corresponding to the maximal eigenvalue $\lambda_{\max}$, then this optimal step size is given by $1/\lambda_{\max}$.

Hessian Matrices

- The second derivative can be used to determine whether a critical point is a local maximum, a local minimum, or saddle point.
- When the second derivative $f''(x)$> 0, the first derivative $f'(x)$ increases as we move to the right and decreases as we move to the left.
  - $f'(x-\epsilon) < \mathbf{0}$ and $f'(x+\epsilon) > \mathbf{0}$ for small enough ϵ.
- When $f'(x)$=0 and $f''(x)$>0, x is a local minimum.
- When $f'(x)$=0 and $f''(x)$<0, x is a local maximum.
- when $f'(x)$=0 and $f''(x)$=0, x may be a saddle point or a part of a flat region.

Hessian Matrices

- In multiple dimensions, the simplest method to utilize Hessian matrices is known as **Newton's method**
- Newton's method is based on using a second-order Taylor series expansion to approximate *f* (*x*) near some point $x^{(0)}$

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

- The critical point of this function is $\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \boldsymbol{H}(f)(\boldsymbol{x}^{(0)})^{-1} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)})$
- Iteratively updating the approximation and jumping to the minimum of the approximation can reach the critical point much faster than gradient descent would.
- This is a useful property near a local minimum, but it can be a harmful property near a saddle point.

Hessian Matrices

- Optimization algorithms that use only the gradient, such as gradient descent, are called **first-order optimization algorithms**.
- Optimization algorithms that also use the Hessian matrix, such as Newton's method, are called **second-order optimization algorithms**

- A Lipschitz continuous function is a function f whose rate of change is bounded by a **Lipschitz constant** $\mathcal{L}$:

$$\forall \boldsymbol{x}, \forall \boldsymbol{y}, |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \mathcal{L}||\boldsymbol{x} - \boldsymbol{y}||_2$$

- A small change in the input will have a small change in the output.

Constrained Optimization

- We may wish to find the maximal or minimal value of $f(x)$ for values of x in some set S.
- This is known as **constrained optimization**.
- Points x that lie within the set S are called **feasible** points.
- One simple approach to constrained optimization is simply to modify gradient descent taking the constraint into account.
- If we use a small constant step size ϵ, we can make gradient descent steps, then project the result back into S.
- If we use a line search, we can search only over step sizes ϵ that yield new x points that are feasible, or we can project each point on the line back into the constraint region.

Constrained Optimization

- A more sophisticated approach is to design a different, unconstrained optimization problem whose solution can be converted into a solution to the original, constrained optimization problem.
- For example, we want to minimize $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^2$ with $\boldsymbol{x}$ constrained to have exactly unit L^2 norm.
- We can instead minimize $g(\theta) = f([\cos \theta, \sin \theta])$ with respect to θ, and then return $[\cos \theta, \sin \theta]$ as the solution to the original problem.

Constrained Optimization

- The **Karush–Kuhn–Tucker** (KKT) approach provides a very general solution to constrained optimization.
- With the KKT approach, we introduce a new function called the **generalized Lagrangian**
- $\mathbb{S} = \{\boldsymbol{x} \mid \forall i, g^{(i)}(\boldsymbol{x}) = 0 \text{ and } \forall j, h^{(j)}(\boldsymbol{x}) \leq 0\}$
- For each constraint, introduce new variables λ_i and α_j, the KKT multipliers.
- The generalized Lagrangian is then defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x})$$

Constrained Optimization

- Suppose at least one feasible point exists and $f(x)$ is not permitted to have value ∞
- We have $\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \min_{\boldsymbol{x} \in \mathbb{S}} f(\boldsymbol{x})$
- Because $\max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x})$
- Karush-Kuhn-Tucker (KKT) conditions:
  - The gradient of the generalized Lagrangian is zero.
  - All constraints on both $\boldsymbol{x}$ and the KKT multipliers are satisfied.
  - The inequality constraints exhibit “complementary slackness”: $\boldsymbol{\alpha} \odot h(\boldsymbol{x}) = 0$.

Example: Linear Least Squares

- Suppose we want to find the value of *x* that minimizes $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$.
- We need to obtain the gradient: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}$.

Algorithm 4.1 An algorithm to minimize $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$ with respect to $\boldsymbol{x}$ using gradient descent, starting from an arbitrary value of $\boldsymbol{x}$.

```
Set the step size ($\epsilon$) and tolerance ($\delta$) to small, positive numbers.
while $||\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}||_2 > \delta$ do
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \epsilon\left(\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}\right)$
end while
```

Example: Linear Least Squares

- One can also solve this problem using Newton’s method.
- In this case, because the objective function is quadratic, the quadratic approximation employed by Newton’s method is exact,
- The algorithm converges to the global minimum in a single step.

Example: Linear Least Squares

- Suppose we wish to minimize the same function, but subject to the constraint $\boldsymbol{x}^{\top}\boldsymbol{x} \leqslant 1$.
- Introduce the Lagrangian: $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda \left(\boldsymbol{x}^{\top} \boldsymbol{x} - 1 \right)$
- We solve the problem: $\min_{\boldsymbol{x}} \max_{\lambda, \lambda \geq 0} L(\boldsymbol{x}, \lambda)$
- The smallest-norm solution to the unconstrained least squares problem can be found using the pseudoinverse: $\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{b}$.
- If this point is feasible, then it is the solution to the constrained problem.
- By differentiating the Lagrangian with respect to x, we obtain the equation: $\boldsymbol{A}^{\top}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^{\top}\boldsymbol{b} + 2\lambda\boldsymbol{x} = 0$ or $\boldsymbol{x} = (\boldsymbol{A}\boldsymbol{A} + 2\lambda\boldsymbol{I})^{-1}\boldsymbol{A}^{\top}\boldsymbol{b}$.
- The magnitude of λ must be chosen such that the result obeys the constraint.

Example: Linear Least Squares

- We can find this value by performing gradient ascent on λ.
- Observe $\frac{\partial}{\partial \lambda} L(\boldsymbol{x}, \lambda) = \boldsymbol{x}^\top \boldsymbol{x} - 1$
- When the norm of $\boldsymbol{x}$ exceeds 1, this derivative is positive
- To follow the derivative uphill, we increase λ
- Increasing λ yields a solution with a smaller norm
- The process of solving the linear equation and adjusting λ continues until x has the correct norm and the derivative on λ is 0.