Fundamentals for Research

Xiaoran Zhang

September 2020

Contents

1 Deep learning

1.1 Concepts

1.1.1 Curse of dimensionality

Definition

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.

The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality. Also, organizing and searching data often relies on detecting areas where objects form groups with similar properties; in high dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient.

Domains

- **Combinatorics**: In some problems, each variable can take one of several discrete values, or the range of possible values is divided to give a finite number of possibilities.
- **Sampling**: There is an exponential increase in volume associated with adding extra dimensions to a mathematical space.
- **Optimization**: When solving dynamic optimization problems by numerical backward induction, the objective function must be computed for each combination of values. This is a significant obstacle when the dimension of the "state variable" is large.
- **Machine Learning**: In machine learning problems that involve learning a "state-of-nature" from a finite number of data samples in a high-dimensional feature space with each feature having a range of possible values, typically an enormous amount of training data is required to ensure that there are several samples with each combination of values.
- **Distance Functions**: When a measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples.

1.1.2 Why use probabilistic model?

In real-world applications, many things that we are trying to model is very complicated; in particular, it often involves a significant amount of uncertainty (e.g., the price of a house has a certain chance of going up if a new subway station opens within a certain distance). It is therefore very natural to deal with this uncertainty by modeling the world in the form of a probability distribution $p(x, y)$. Given such a model, we could ask questions such as "what is the probability that house prices will rise over the next five years?", or "given that the house costs $100,000, what is the probability that it has three bedrooms?"

1.2 Generative Adversarial Networks

1.2.1 Background

Entropy

The entropy is firstly defined after Boltzman H-theorem, which is used to characterize the uncertainty in gas molecules. It is then introduced by Shannon (2001) to measure the uncertainty (surprise) of variables' potential outcomes. For instance, the flip of a coin has two outcomes, head or tail. To quantify this, we could introduce a one-bit entropy. This example gives the intuition of using logarithmic function to calculate the information measure.

The intuition to quantify the information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

- **Low probability event:** high information (surprising).
- **High probability event:** low information (unsurprising).

From the above two intuitions, we first define the information content (self-information) given an event x with probability $p(x)$

$$I(x) = -\log(x) = \log(\frac{1}{p(x)}). \tag{1.1}$$

Given a discrete random variable X, with possible outcomes $x_1, ..., x_n$, which occur with $p(x_1), .., p(x_n)$, the entropy of X is defined as

$$H(X) = \mathbb{E}(I_X(x_i)) = -\sum_{i=1}^{n} p(x_i)\log p(x_i). \tag{1.2}$$

Different base of log indicates different units. Base 2 gives the unit of bits (or "shannons"), while base e gives the "natural units" nat, and base 10 gives a unit called "dits", "bans", or "hartleys".

Cross entropy

The cross entropy of the distribution Q relative to a distribution P over a given set is defined as follows:

$$H(P,Q) = -\mathbb{E}[\log(Q)] \tag{1.3}$$

and its relation with KL divergence and entropy is

$$H(P,Q) = H(P) + D_{KL}(P||Q). \tag{1.4}$$

Thus, in many applications where P is a given distribution, we could view cross entropy similar to KL divergence, which is used as the dissimilarity metric to characterize the difference between approximated distribution Q and P.

Kullback-Leibler divergence

KL divergence (relative entropy) is a measure of how one probability distribution Q is different from a second P. Usually, P represents the data, the observations, or a probability distribution precisely measured. Distribution Q represents instead a theory, a model, a description or an approximation of P. The KL divergence is then interpreted as the average difference of the number of bits required for encoding samples of P using a code optimized for Q rather than one optimized for P.

For discrete probability distributions P and Q defined on the same probability space, $\mathcal{X}$, the KL divergence from Q to P is defined as

$$D_{\mathrm{KL}}(P||Q) = -\sum_{x\in\mathcal{X}} P(x)\log(Q(x)) - \left(-\sum_{x\in\mathcal{X}} P(x)\log(P(x))\right) \tag{1.5}$$

$$= H(P,Q) - H(P) \tag{1.6}$$

$$= \sum_{x\in\mathcal{X}} P(x)\log(\frac{P(x)}{Q(x)}) \tag{1.7}$$

However, KL divergence is assymetric as when p is close to 0, no matter what large value q gets, it will be disregarded. This will produce buggy result in neural network training. Thus, Jensen-Shannon (JS) divergence is introduced

$$D_{\mathrm{JS}}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2}). \tag{1.8}$$

JS divergence is bounded by [0,1], symmetric and more smooth. An example comparing KL divergence with JS divergence is shown in Fig. 1.1.

1.2.2 Vanilla GAN

The vanilla GAN is firstly proposed by Goodfellow et al. (2014) and introduces a generative model and a discriminative model to play a minmax game with value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[1-\log(D(G(z)))]. \tag{1.9}$$

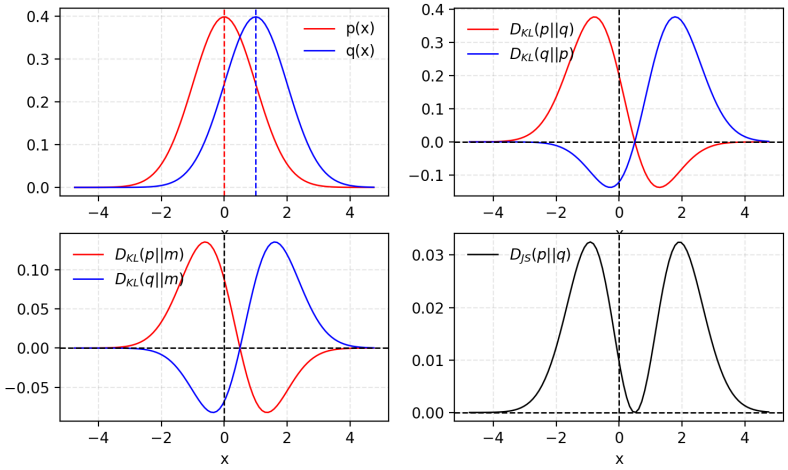

Figure 1.1: Given two Gaussian distribution, p with mean=0 and std=1 and q with mean=1 and std=1. The average of two distributions is labelled as $m = (p + q)/2$. KL divergence D_{KL} is asymmetric but JS divergence D_{JS} is symmetric. Credit: Lilian Weng.

The general intuition for the loss function is a cross entropy term

1.3 Representation Learning

1.3.1 Continuity, Cluster and Manifold Assumptions for Semi-supervised Learning

Continuity Assumption

Points that are close to each other are more likely to share a label

Cluster Assumption

The data tend to form discrete clusters, and points in the same cluster are more likely to share a label

Manifold Assumption

The data lie approximately on a manifold of much lower dimension than the input space.

1.3.2 Metric Learning

Introduction

Many approaches in machine learning require a measure of distance between data points. Traditionally, practitioners would choose a standard distance metric (Euclidean, City-Block, Cosine, etc.) using a priori knowledge of the domain. However, it is often difficult to design metrics that are well-suited to the particular data and task of interest.

Distance metric learning (or simply, metric learning) aims at automatically constructing task-specific distance metrics from (weakly) supervised data, in a machine learning manner. The learned distance metric can then be used to perform various tasks (e.g., k-NN classification, clustering, information retrieval).

1.3.3 Contrastive Learning

Triplet, Max-margin and N-pairs loss

Triplet loss proposed by Schroff et al. (2015) is a loss function for machine learning algorithms where a baseline (anchor) input is compared to a positive (truthy) input and a negative (falsy) input. The distance from the baseline (anchor) input to the positive (truthy) input is minimized, and the distance from the baseline (anchor) input to the negative (falsy) input is maximized. The loss could be written in the following form:

$$\mathcal{L}(A, P, N) = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0) \tag{1.10}$$

where A is an anchor input, P is a positive input of the same class as A, N is a negative input of a different class from A, α is a margin between positive and negative pairs, and f is an embedding.

Max-margin loss aka large-margin loss is proposed by Liu et al. (2016) which aims to encourage intra-class compactness and inter-class-separability. The intuition is to enlarge the margin for inter-class decision boundary by introducing a parameter m from originally softmax $W_1^T x > W_2^T x$ to $||W_1||||x|| \cos(\theta_1) \geq ||W_1||||x|| \cos(m\theta_1) > ||W_2||||x|| \cos(\theta_2)$ for $0 \leq \theta_1 \leq \frac{\pi}{m}$. Thus the L-softmax is defined as

$$L_i = -\log\left(\frac{e^{||W_{y_i}||||x_i||\psi(\theta_{y_i})}}{e^{||W_{y_i}||||x_i||\psi(\theta_{y_i})} + \sum_{j\neq i} e^{||W_{y_j}||||x_i|| \cos(\theta_j)}}\right) \tag{1.11}$$

in which we require

$$\psi(\theta) = \begin{cases} \cos(m\theta) & 0 \leq \theta \leq \frac{\pi}{m} \\ \mathcal{D}(\theta) & \frac{\pi}{m} < \theta \leq \pi \end{cases} \tag{1.12}$$

where m is a integer that is closely related to the classification margin. With larger m, the classification margin becomes larger and the learning objective also becomes harder. Meanwhile, $\mathcal{D}(\theta)$ is required to be a monotonically decreasing function and $\mathcal{D}(\frac{\pi}{m}) = \cos(\frac{\pi}{m})$. Note that in the denominator the inter-class subjects are using the cosine similarity instead of the defined metric. This is obvious because we want the inequality $||W_1||||x|| \cos(\theta_1) \geq ||W_1||||x|| \cos(m\theta_1) > ||W_2||||x|| \cos(\theta_2)$ holds. If using the defined $\psi(\cdot)$ function, it is not enhancing the loss function.

N-pair loss is proposed by Sohn (2016) to solve the issue that existing frameworks of deep metric learning based on contrastive loss and triplet loss often suffer from slow convergence, partially because they employ only one negative example while not interacting with the other negative classes in each update. One immediate concern with (N+1)-tuplet loss is that it quickly becomes intractable when scaling up since the number of examples to evaluate in each batch grows in quadratic to the number of tuplets and their length N. Note that the scaling is because there are N samples and for each sample we have to construct a tuplet with size N+1 to evaluate the loss. The key ideas behind N-pair loss are 1) combine contrastive loss with triplet loss 2) construct a tuplet containing 2N samples. The loss is defined as following:

$$\mathcal{L}_{N-pair-mc}(\{(x_i, x_i^+)\}_{i=1}^N; f) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \sum_{j\neq i} \exp\left(f_i^T f_j^+ - f_i^T f_i^+\right)\right) \tag{1.13}$$

when N=2. it is very similar to triplet loss. How they reduce the sample to 2N is quite trivial. Given the pairs such as $\{(x_1, x_1^+), ..., (x_N, x_N^+)\}$, they use positive samples in other pairs to construct the negative samples (shown in Fig. 1.2), which is 2N.

1.3.4 Variational Autoencoder (VAE)

Latent Variable Models

Latent variable will help the generative model to decide before generating something, such as an image. Take generating handwritten digits for example. The model will first samples a digit value from 0,1,...,9 before generating the character. Otherwise it would result in chaos as obviously the combination of left half of digit 5 with right half of digit 0 will produce nothing

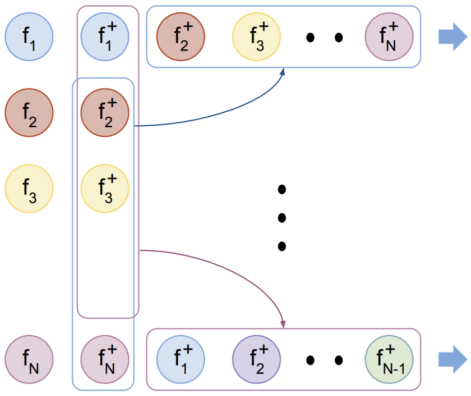

Figure 1.2: N-pair-mc loss

meaningful. Following this intuition, we have a vector of latent variables $z \in \mathcal{Z}$ and could be easily sampled followed by some probability density function $p(z)$ defined over $\mathcal{Z}$. Then, we have a family of deterministic functions $f(z;\theta)$ parameterized by θ in some Θ and $f(\cdot)$ is a deterministic function with mapping $\mathcal{Z} \times \Theta \rightarrow \mathcal{X}$. $f(z;\theta)$ is the thing we generated, similar to X. We wish to optimize θ such that we can sample z from $P(z)$ and, with high probability, $f(z;\theta)$ will be like the X's in our dataset. To formulate the generative problem, we have

$$P(X) = \int P(X|z;\theta)P(z)dz \tag{1.14}$$

VAE Formulation

To solve the above equation, there are two problems that VAEs must deal with: (1) how to define the latent variables z (i.e., decide what information they represent) and (2) how to deal with the integral over z.

(1) How to define the latent variables:

From the hand-written digit samples, we want to select the a set of random variables that represent certain aspects of the final outcome, such as digit tilting angle, stroke width, stylistic properties etc. These things might be correlated. Thus, VAEs take an unusual approach to dealing with this problem: they assume that there is no simple interpretation of the dimensions of z, and instead assert that samples of z can be drawn from a simple distribution, i.e., $\mathcal{N}(0, I)$, where I is the identity matrix. Following the intuition in Fig. 1.3, we could learn the function to transform z to any latent representation through our training data and then map those latent variables to X, given strong function approximators. "*In fact, recall that* $P(X|z;q) = N(X|f(z;q), \sigma^2 * I)$. *If* $f(z;q)$ *is a multi-layer neural network, then we can imagine the network using its first few layers to map the normally distributed z's to the latent values (like digit identity, stroke weight, angle, etc.) with exactly the right statistics. Then it can use later layers to map those latent values to a fully-rendered digit. In general, we don't need to worry about ensuring that the latent structure*

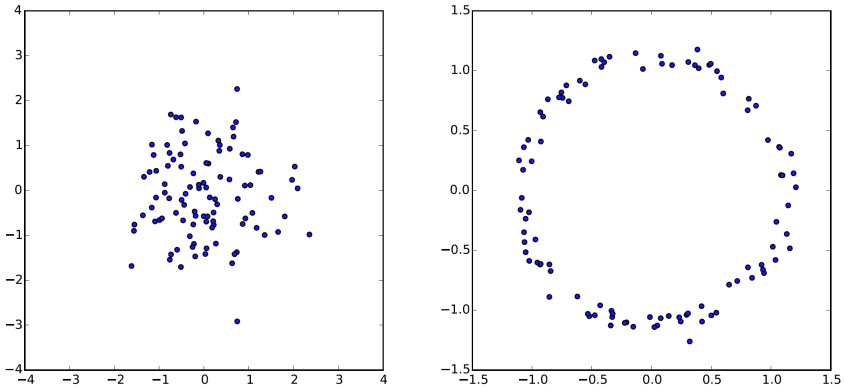

Figure 1.3: Given a random variable z with one distribution, we can create another random variable $X = g(z)$ with a completely different distribution. Left: samples from a gaussian distribution. Right: those same samples mapped through the function $g(z) = z/10 + z/||z||$ to form a ring. This is the strategy that VAEs use to create arbitrary distributions: the deterministic function $g(\cdot)$ is learned from data.

exists. If such latent structure helps the model accurately reproduce (i.e. maximize the likelihood of) the training set, then the network will learn that structure at some layer."

(2) How to deal with the integral to maximize $P(X)$

2 Mathematics

2.1 Probability Theory

2.1.1 Probability distribution functions

Cumulative distribution function (CDF)

CDF is defined for both discrete and continuous random variables.

$$F(x) = \Pr(X \leq x) \tag{2.1}$$

Probability density function (PDF) and probability mass function (PMF)

PDF is defined for continuous random variables and PMF is defined for discrete random variables.

$$f(x) = \Pr(X = x) \tag{2.2}$$

Relation between CDF and PDF/PMF

For continuous random variables, the relation could be characterized as

$$F(x) = \int_{-\infty}^{x} f(x)\mathrm{d}x \tag{2.3}$$

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x} \tag{2.4}$$

For discrete random variables, the relation is

$$F(x) = \sum_{x_i \leq x} f(x_i) \tag{2.5}$$

where $f(x_i)$ could also be written as $p(x_i)$.

2.1.2 Some common random variables

Discrete random variables

A good reference for the recap http://cs229.stanford.edu/section/cs229-prob.pdf. $P(X = x)$.

- $X \sim Bernoulli(p)$, the probability of a coin flip's results in head ($x = 1$) and tail ($x = 0$).

$$p(x) = \begin{cases} p & x = 1 \\ 1 - p & x = 0 \end{cases} \tag{2.6}$$

- $X \sim Binomial(n, p)$, the sum of N-times Bernoulli with probability p of coin's head.

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{2.7}$$

- $X \sim Geometric(p)$, the number of flips of coin with head probability p until the first heads

$$p(x) = p(1-p)^{x-1} \tag{2.8}$$

- $X \sim Poisson(\lambda)$, a probability distribution modeling the occurrence of rare events

$$p(x) = e^{-\lambda} \frac{\lambda^x}{x!} \tag{2.9}$$

Continuous random variables

- $X \sim Uniform(a, b)$ equal probability density between range $[a, b]$

$$f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & otherwise \end{cases} \tag{2.10}$$

- $X \sim Exponential(\lambda)$ time between two occurrences in Poisson's distribution

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & otherwise \end{cases} \tag{2.11}$$

- $X \sim Normal(\mu, \sigma^2)$ Gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{2.12}$$

Relation between exponential and Poisson distribution

- Suppose N_t is the number of occurrence at time t. X_t is the time for the next occurrence at time t. Thus, we have for $X_t > x$, $N_t \equiv N_{t+x}$. We have

$$1 - p(X_t \leq x) = p(N_t \equiv N_{t+x}) \tag{2.13}$$

$$= e^{-\lambda x} \frac{\lambda x^0}{0!} \tag{2.14}$$

$$= e^{-\lambda x} \tag{2.15}$$

  Thus we have $p(X_t \leq x) = 1 - e^{-\lambda x}$, which is a exponential distribution.

- The Poisson process is based on the assumption that the occurrence of different events are independent (independent increment process).

2.2 Convex Optimization

2.2.1 Introduction

2.3 Monte Carlo Methods

2.3.1 Importance Sampling

Importance sampling is a general technique for estimating properties of a particular distribution while only having samples generated from a different distribution than the distribution of interest.

2.4 Bayesian Statistics

2.4.1 Why we usually assumes Gaussian in distribution?

It is hard to characterize a distribution using a limited number of parameters. However, Gaussian distribution is completely determined by its mean and covariance and thus we usually assume Gaussian.

2.4.2 Isotropic Gaussian

As a random Gaussian distribution is determined by mean and covariance, $\mathcal{N} \sim (\mu, \Sigma)$ and a isotropic Gaussian is determined as a Gaussian with covariance $\sigma^2 I$, which means that each dimension is independent with each other. In this case, the Gaussian will be circular or spherical.

2.4.3 Maximum likelihood estimation

Definition

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.

Construct a likelihood function

Discrete probability distribution: Let X be the random variable with pmf p depending on parameter θ. Then the likelihood function is constructed as

$$\mathcal{L}(\theta|x) = p_\theta(x) = P_\theta(X = x) \tag{2.16}$$

This could be interpreted as given the parameter θ the likelihood (probability) of observing x. Note that the likelihood function $\mathcal{L}(\theta|x)$ is a function over θ and should not be confused with $p(\theta|x)$. $p_\theta(x)$ could also be written as $p(x|\theta)$.

Continuous probability distribution: Similarly, we have

$$\mathcal{L}(\theta|x) = f_\theta(x) \tag{2.17}$$

Estimation goal

The goal of MLE is to obtain $\hat{\theta}$ such that

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \hat{\mathcal{L}}(\theta|x) \tag{2.18}$$

2.4.4 Maximum a posteriori estimation

Definition

In Bayesian statistics, a maximum a posteriori probability (MAP) estimate is an estimate of an unknown quantity, that equals the mode of the posterior distribution.

Estimation goal

We now have a prior distribution of θ denoted as $g(\theta)$. Now we want to calculate the posteriori probability $p(\theta|x)$ that

$$f(\theta|x) = \frac{f(x|\theta)g(\theta)}{\int_{\mathcal{V}} f(x|v)g(v)dv} \tag{2.19}$$

Thus we have

$$\hat{\theta}_{MAP} = \arg\max_{\theta} f(\theta|x) = \arg\max_{\theta} \frac{f(x|\theta)g(\theta)}{\int_{\mathcal{V}} f(x|v)g(v)dv} = \arg\max_{\theta} f(x|\theta)g(\theta) \tag{2.20}$$

2.4.5 Difference between MLE and MAP

Assumption is different

MLE belongs to a frequency assumption. We assume that the world is static. We are trying to find the parameter that will result

MAP belongs to a bayesian assumption. We assume that the world is dynamic. If people have different prior assumption over the parameter, then the estimated parameter will be different.

An example

Suppose we have a coin, which we are not sure whether it is biased or not. We flipped the coin 10 times and the results are all head. According to the MLE, the coin will be biased with $p = 1$ for head. According to the MAP, if we have a prior distribution over the coin, then estimated parameter will be in a range $[0.5, 1]$ instead of making the wild prediction because of the $g(\theta)$ in the objective function.

2.4.6 Overview of probabilistic analysis

Usually a probabilistic analysis includes three different parts: representation (how to specify a model), inference (how to ask the model questions), and learning (how to fit a model to real-world data). These three themes will also be closely linked: to derive efficient inference and learning algorithms, the model will need to be adequately represented; furthermore, learning models will require inference as a subroutine.

2.4.7 Topological space

A topological space is an ordered pair (X, τ), where X is a set and τ is a collection of subsets of X, satisfying the following axioms:

- The empty set and X itself belong to τ.
- Any arbitrary (finite or infinite) union of members of τ still belongs to τ.
- The intersection of any finite number of members of τ still belongs to τ.

The elements of τ are called open sets and the collection τ is called a topology on X.

2.4.8 Elements in tree of directed graph theory

- **Parent:** The vertex connected to v on the path to the root; every vertex has a unique parent except the root which has no parent.
- **Child:** A vertex of which v is the parent.
- **Ascendant:** Any vertex which is either the parent of v or is (recursively) the ascendant of the parent of v
- **Descendant:** Any vertex which is either the child of v or is (recursively) the descendant of any of the children of v.
- **Leaf:** A vertex with no children
- **Sibling:** Any other vertex on the tree which has the same parent as v
- **Internal vertex:** A vertex that is not a leaf.

2.4.9 Random field

Given a probability space $(\Omega, \mathcal{F}, P)$, an X-valued random field is a collection of X-valued random variables indexed by elements in a topological space T. That is, a random field F is a collection

$$\{F_t : t \in T\} \tag{2.21}$$

where each F_t is an X-valued random variable.

2.4.10 Bayes' nets

A good reference for the recap https://ermongroup.github.io/cs228-notes/.

Definition

Bayesian network is a directed acyclic graph $G = (V, E)$ together with

- A random variable x_i for each node $i \in V$

- One conditional probability distribution $p(x_i, x_{A_i}$ per node, specifying the probability of x_i conditioned on its parents' values.

Thus, Bayesian network defines a probability distribution p. Conversely, we could say that a probability p factorizes over a DAG G if it can be decomposed into a product of factors, as specified by G.

Determine dependencies in Bayesian networks

- **Three-node examples:** Shown in Fig. 2.1. 1) Common parent: if G is of form $A \leftarrow B \rightarrow C$, and B is observed, then $A \perp\!\!\!\perp C \mid B$, if B is unobserved, then $A \not\perp\!\!\!\perp C$. 2) Cascade: if G is in form $A \leftarrow B \leftarrow C$ or $A \rightarrow B \rightarrow C$, then if B is observed, then $A \perp\!\!\!\perp C \mid B$, if B is unobserved, then $A \not\perp\!\!\!\perp C$. 3) V-structure: if G is $A \leftarrow B \rightarrow C$, then if B is observed, then $A \not\perp\!\!\!\perp C$, if B is unobserved, then $A \perp\!\!\!\perp C \mid B$. This is the opposite then the above two cases.
- **d-separation**: "d" standards for dependence. Let Q, W, and O be three sets of nodes in a Bayesian network G. We say that Q and W are d-separated given O if Q and W are not connected by an active path. An path in G is called active if for every consecutive triples (X, Y, Z) on the path, one of the following holds: 1) $X \leftarrow Y \leftarrow Z$ and Y is unobserved $Y \notin O$ 2) $X \rightarrow Y \rightarrow Z$ and Y is unobserved $Y \notin O$ 3) $X \leftarrow Y \rightarrow Z$ and Y is unobserved $Y \notin O$ 4) $X \leftarrow Y \rightarrow Z$ and Y is unobserved or any of its descendants are observed.

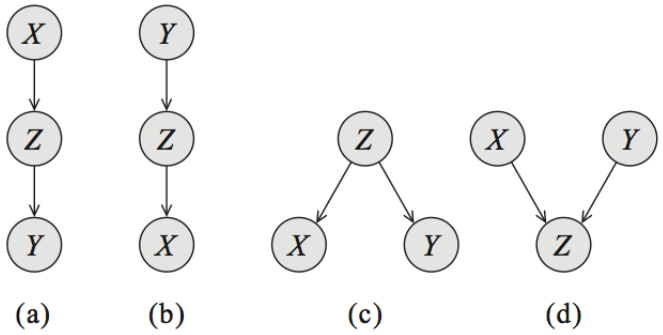

Figure 2.1: Bayesian networks over three variables, encoding different types of dependencies: cascade (a,b), common parent (c), and v-structure (d).

Equivalence of directed graphs

- A single DAG may not express all the independencies of any distribution p. Example: Sample $X, Y \sim Bernouli(0.5)$, $Z = X \text{xor} Y$.
- One intuition to find a minimal I-map for p is to start with a fully connected one, and then remove each edge until it is not a I-map.
- A perfect DAG that defines all the independencies might not be unique.

- Two Bayes nets are said to be I-equivalent if they encode the same dependencies $I(G_1) = I(G_2)$.
- If G, G' have the same skeleton and the same V-structures, then $I(G) = I(G')$.

2.4.11 Markov random field

Definition

Given an undirected graph $G = (V, E)$, a set of random variables $X = (X_v)_{v \in V}$ indexed by V form a Markov random field with respect to G if they satisfy the local Markov properties:

- **Local Markov property:** A variable is conditionally independent of all other variables given its neighbors: $X_v \perp\!\!\!\perp X_{V \setminus N(v)} \mid X_{N(v)}$ where $N(v)$ is the set of neighbors of v, and $N(v) \cup v$ is a closed neighbor of v.

The following properties are equivalent to the local Markov property when the probability is positive. But generally, global Markov property is stronger than local Markov, local Markov is stronger than pairwise Markov.

- **Global Markov property:** Any two subsets of variables are conditionally independent given a separating subset: $X_A \perp\!\!\!\perp X_B \mid X_S$ where every path from a node in A to a node in B passes through S.
- **Pairwise Markov property:** Any two non-adjacent variables are conditionally independent given all other variables: $X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}}$.
- If p factorizes over G, then $I(G) \subseteq I(p)$. We say that G is a I-map (independent map) of p. However, the converse is not true. If a distribution is jointly independent, then any independent maps hold. A single map doesn't necessary catch the necessary features.

Exponential family

2.4.12 Gibbs random field

2.4.13 Conditional random field

2.4.14 Gaussian random field

2.5 Control Theory

2.5.1 Kalman Filter

Intuition

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

Characteristics

- Optimality of the Kalman filter assumes that the errors are Gaussian.
- Though regardless of Gaussianity, if the process and measurement covariances are known, the Kalman filter is the best possible linear estimator in the minimum mean-square-error sense.
- Extensions and generalizations to the method have also been developed, such as the extended Kalman filter and the unscented Kalman filter which work on nonlinear systems.

Underlying dynamical system model

In order to use the Kalman filter to estimate the internal state of a process given only a sequence of noisy observations one must model the process in accordance with the following framework. This means specifying the following matrices:

- F_k: the state-transition model;
- H_k: the observation model;
- Q_k: the covariance of the process noise;
- R_k: the covariance of the observation noise;
- B_k: the control-input model, for each time-step, k, as described below.

The Kalman filter model assumes the true state at time k is evolved from the state at $k-1$ according to

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{2.22}$$

where

- F_k is the state transition model which is applied to the previous state x_{k-1};
- B_k is the control-input model which is applied to the control vector u_k;
- w_k s the process noise, which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k: $w_k \sim \mathcal{N}(0, Q_k)$

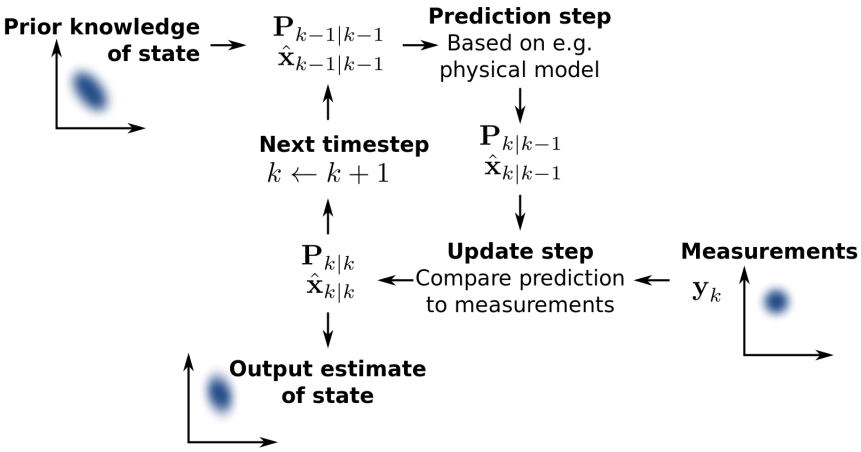

Figure 2.2: Kalman filter pipeline. Credit: Wikipedia.

At time k an observation (or measurement) z_k of the true state x_k is made according to

$$z_k = H_k x_k + v_k \tag{2.23}$$

where

- H_k is the observation model, which maps the true state space into the observed space and
- v_k is the observation noise, which is assumed to be zero mean Gaussian white noise with covariance $R_k : v_k \sim \mathcal{N}(0, R_k)$

Filter details

Fig. 2.2 shows the general pipeline for Kalman filter. The state of the filter is represented by two variables:

- $\hat{x}_{k|k}$, the a posteriori state estimate at time k given observations up to and including at time k;
- $P_{k|k}$, the a posteriori estimate covariance matrix (a measure of the estimated accuracy of the state estimate).

The Kalman filter is mostly conceptualized as two distinct phases: "Predict" and "Update".

Phase 1: Predict (Prior state estimate)

The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. This predicted state estimate is also known as the a priori state estimate because, although it is an estimate of the state at the current timestep, it does not include observation information from the current timestep.

(1) Predicted (a priori) state estimate

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \tag{2.24}$$

(2) Predicted (a priori) estimate covariance

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.25}$$

Phase 2: Update (Posteriori state estimate)

In the update phase, the current a priori prediction is combined with current observation information to refine the state estimate. This improved estimate is termed the a posteriori state estimate.

(1) Innovation or measurement pre-fit residual

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \tag{2.26}$$

(2) Innovation (or pre-fit residual) covariance

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.27}$$

(3) Optimal Kalman gain

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.28}$$

(4) Updated (a posteriori) state estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.29}$$

(5) Updated (a posteriori) estimate covariance

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.30}$$

(6) Measurement post-fit residual

$$\tilde{y}_{k|k} = z_k - H_k \hat{x}_{k|k} \tag{2.31}$$

A more intuitive way to express the Updated state estimate ($\hat{x}_{k|k}$) is

$$\hat{x}_{k|k} = (I - K_k H_k)(\hat{x}_{k|k-1}) + (K_k)(H_k x_k + v_k) \tag{2.32}$$

This expression reminds us of a linear interpolation $x = (1-t)(a) + t(b)$ for t between $[0, 1]$. In our case:

- t is the Kalman gain K_k, a matrix that takes high values from 0 (high error in the sensor) to I (low error).
- Kalman gain could also be viewed as (belief in measurement)/(belief in state).
- a is the value estimated from the model.
- b is the value from the measurement.

Relation with the Markov models

Kalman filters are based on linear dynamical systems discretized in the time domain. They are modeled on a Markov chain built on linear operators perturbed by errors that may include Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the observed outputs from the true ("hidden") state. The Kalman filter may be regarded as analogous to the hidden Markov model, with the key difference that the hidden state variables take values in a continuous space as opposed to a discrete state space as in the hidden Markov model. There is a strong analogy between the equations of the Kalman Filter and those of the hidden Markov model.

Why KF assumes Gaussian noise

Assumption of a Gaussian process allows us to obtain optimality. In the derivation of the KF for optimality, we update the mean and covariance and Gaussian process is completemly defined by its mean and variance. We could use KF in the real world when the noise is not Gaussian but then it wouldn't reach optimality. However, even if the noise is not Gaussian, the Kalman filter is still the optimal linear filter.

2.5.2 Extended Kalman Filter

Formulation

In the extended Kalman filter, the state transition and observation models don't need to be linear functions of the state but may instead be differentiable functions.

$$x_k = f(x_{k-1}, u_k) + w_k \tag{2.33}$$

$$z_k = h(x_k) + v_k \tag{2.34}$$

Disadvantages

Unlike its linear counterpart, the extended Kalman filter in general is not an optimal estimator (it is optimal if the measurement and the state transition model are both linear, as in that case the extended Kalman filter is identical to the regular one). In addition, if the initial estimate of the state is wrong, or if the process is modeled incorrectly, the filter may quickly diverge, owing to its linearization. Another problem with the extended Kalman filter is that the estimated covariance matrix tends to underestimate the true covariance matrix and therefore risks becoming inconsistent in the statistical sense without the addition of "stabilising noise".

2.5.3 Unscented Kalman Filter

Intuition

- When the state transition and observation models—that is, the predict and update functions $f(\cdot)$ and $h(\cdot)$ are highly nonlinear, the extended Kalman filter can give particularly poor performance.
- The unscented Kalman filter (UKF) uses a deterministic sampling technique known as the unscented transformation (UT) to pick a minimal set of sample points (called sigma points) around the mean. The sigma points are then propagated through the nonlinear functions, from which a new mean and covariance estimate are then formed.

2.5.4 Differences between KF, EKF and UKF

- **KF**: The Kalman Filter is a filter that works as a least square error optimizer, and, for this to work, it is necessary that the system that you consider inside the filter is linear.
- **EKF**: In order to make state estimation on nonlinear systems, or parameter estimation, using the Kalman filter, one of the possible approaches is to linearize the system under investigation around its current state and force the filter to use this linearized version of your system as a model, usually in a first-order or second-order approximation manner. This is the Extended Kalman Filter, or EKF.
- **UKF**: However, the EKF is not very stable and many times, when it does converge to the "right" solution, it does it very slowly. In order to improve this filter, instead of using linearization to predict the behavior of the system under investigation, some authors started using the Unscented Transformation. Because the Unscented transformation somehow describes the nonlinear system better than the linearization, hence this filter converges to the right solution more rapdly. However, as the EKF, this filter may become unstable and results may be biased. Hence, the Kalman Filter with the Unscented transformation is called Unscented Kalman Filter, or UKF.

2.6 Computational Anatomy

2.6.1 Homeomorphism

Intuition

In the mathematical field of topology, a homeomorphism, topological isomorphism, or bicontinuous function is a continuous function between topological spaces that has a continuous inverse function.

Definition

A function $f : X \to Y$ between two topoligcal spaces is a homeomorphism if it has the following properties:

- f is a bijection
- f is continuous
- the inverse function f^{-1} is continuous

2.6.2 Manifold

Definition

An n-dimensional manifold, or n-manifold for short, is a topological space with the property that each point has a neighborhood that is homeomorphic to the Euclidean space of dimension n.

Charts, atlases, and transition maps

- **Charts**: A coordinate map, a coordinate chart, or simply a chart, of a manifold is an invertible map between a subset of the manifold and a simple space such that both the map and its inverse preserve the desired structure.
- **Atlases**: The description of most manifolds requires more than one chart (a single chart is adequate for only the simplest manifolds). A specific collection of charts which covers a manifold is called an atlas. An atlas is not unique as all manifolds can be covered multiple ways using different combinations of charts. Two atlases are said to be equivalent if their union is also an atlas.
- **Transition maps**: Charts in an atlas may overlap and a single point of a manifold may be represented in several charts. If two charts overlap, parts of them represent the same region of the manifold, just as a map of Europe and a map of Asia may both contain Moscow. Given two overlapping charts, a transition function can be defined which goes from an open ball in $\mathbb{R}^n$ to the manifold and then back to another (or perhaps the same) open ball in $\mathbb{R}^n$.

2.6.3 Diffeomorphism

Definition

A diffeomorphism is an isomorphism of smooth manifolds. It is an invertible function that maps one differentiable manifold to another such that both the function and its inverse are smooth.

3 Imaging

3.1 Digital image processing

3.1.1 Time-invariant and shift-invariant systems

Time-invariance

Given a system with a time-dependent output function $y(t)$, and a time-dependent input function $x(t)$; the system will be considered time-invariant if a time-delay on the input $x(t+\delta)$ directly equals to a time-delay of the output $y(t+\delta)$ function. For example, if time t is elapsed time and then time-invariance implies that the relationship between the input function $x(t)$ and the output function $y(t)$ is constant with respect to time t:

$$y(t) = f(x(t), t) = f(x(t)) \tag{3.1}$$

Shift-invariance

Shift-invariance is the discrete equivalent of a time-invariant system, defined such that if $y[n]$ is the response of the system to $x[n]$, then $y[n-k]$ is the response of the system to $x[n-k]$. Shift-invariance indicates that the relationship between the input and the output is not a direct function of shift n:

$$y[n] = f(x[n], n) = f(x[n]) \tag{3.2}$$

Bibliography

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680.

Liu, W., Y. Wen, Z. Yu, and M. Yang (2016). Large-margin softmax loss for convolutional neural networks. In *ICML*, Volume 2, pp. 7.

Schroff, F., D. Kalenichenko, and J. Philbin (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review 5*(1), 3–55.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pp. 1857–1865.