Density Estimation and Generative Models

Dr Chris G. Willcocks

Last Modified: February 12, 2019

Overview

The purpose of this lecture is to revise some basic statistics, and to introduce the theory behind generative models. This revision also serves as an important basis for some of the underlying properties of classification and clustering theory.

Discriminative Models

So far, we've mainly looked at models as function approximators. For example we have a training set of x, y pairs, where X is some space and Y belongs to some target space of outputs. We try to learn a function that maps from the X to Y, ideally in a way that generalises well.

Given n examples and labels x_i, y_i learn $h : X \to Y$

Generative Models

A generative model is just some distribution over observations.

Generative models are slightly different, in that we still have a training set but with only X. However we make an assumption that x was uniformly sampled from some underlying probability distribution.

Given n examples x_i, recover $P(x)$.

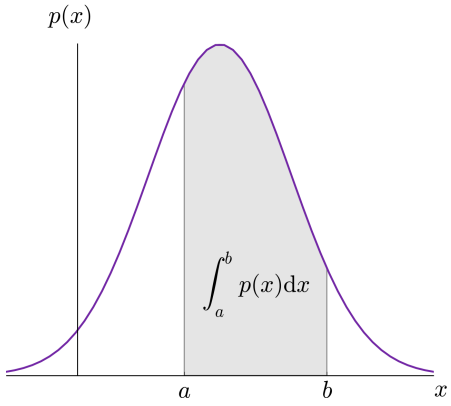
1. Linguists might like to build a distribution over sentences by building a probablistic context-free grammar.
   (a) What is the probability of a given sentence? P(sentence)
   (b) What is the probability of 'Brian being a Dog?'
2. Meteorologists might like to build a distribution that predicts whether its going to rain P(rain).
3. What is the probability of this image being a face? P(face)

4. What is the probability of this waveform being composed by Beethoven? $P(\text{beethoven})$

1 Density Estimation

We'll start with some simple revision on distributions.

Definition 1.1 (Probability density function)**.** The PDF, or density of a continuous random variable, is a function that describes the relative liklihood for a random variable X to take on a given value x.



The integral over the PDF between a and b gives the likelihood of finding the value of x in that range. PDF's must satisfy the following conditions:

The likelihood to find any given value cannot be less than zero

$$\text{PDF}(x) \geq 0 \ \forall x \in \mathbb{R} \tag{1}$$

The total area under the curve is equal to 1

$$\int_{-\infty}^{\infty} \text{PDF}(x)dx = 1 \tag{2}$$

Example 1.1. *Lets say that the mean height of people in Durham is* $\mu = 178\text{cm}$ *and a standard deviation* $\sigma = 8\text{cm}$*. Lets also assume the height of the population follows the Gaussian distribution where*

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}. \tag{3}$$

We can then answer the question, what is the probability of someone being between 180cm and 190cm tall? Then

$$\int_{180}^{190} p(x)dx = 0.334 \tag{4}$$

or we can find the probability of someone being between 210cm and 250cm tall

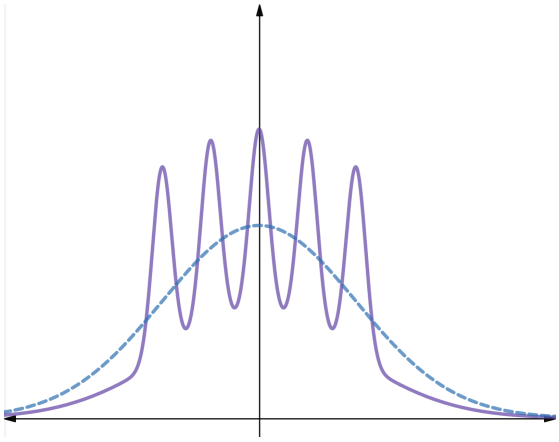
$$\int_{210}^{250} p(x)dx = 0.000003. \tag{5}$$

It's very unlikely someone is that tall.

In the example before, we can parameterize the distribution with the mean μ and standard deviation σ. These statistics are said to be **sufficient** if they are enough to reconstruct the properties of the underlying points, in this case how they are distributed.

Example 1.2. *In a more complex setting, we may have a density such as the 'Bart Simpson' density, as Nando de Freitas likes to call it*

$$p(x) = \frac{1}{2}\phi(x; 0, 1) + \frac{1}{10}\sum_{j=0}^{4}\phi(x; (j/2) - 1, 1/10) \tag{6}$$



where ϕ is the normal density with mean μ and standard deviation σ. This density cannot be sufficiently estimated with a normal distribution, as the result is over-smoothed (blue).

Definition 1.2 (Histograms)**.** One of the most simple density estimators is the histogram estimator, which divides the space (for example assuming $\mathcal{X} = [0, 1]^d$) up into bins $B_1, B_2, ..., B_N$ and counts the elements inside the bins

$$\hat{p}_{\text{hist}}(x) = \sum_{j=1}^{n}\frac{\hat{\theta}_j}{h^d}I(x \in B_j), \tag{7}$$

where $\hat{\theta}_j$ is the proportion of observations in that bin, scaled by a factor that integrates to one

$$\hat{\theta}_j = \frac{1}{n} \sum_{i=1}^{n} I(X_i \in B_j) \tag{8}$$

we still have some tuning parameter for the number of bins $N \approx (\frac{1}{h})^d$.

Definition 1.3 (Kernel density estimators)**.** Another popular approach for simple cases is to employ techniques known as kernel density estimators

$$\hat{p}_{\text{kde}}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{||x - X_i||}{h} \right). \tag{9}$$

where at each point x, $\hat{p}_{\text{kde}}(x)$ is the average of the kernel function centered over the data points X_i, and h is the bandwidth.

Here's a nice interactive visualisation link:

https://mathisonian.github.io/kde/

what's important to understand is that, for whatever estimator you use, there's inevitably going to be some sort of tuning parameter(s). The fundamental thing in density estimation is choosing the right tuning parameters (choosing how much smoothing to do).

In the case of trillions of data examples (images, audio, ...), we want to find a representation that's small and succinct, describing all of the examples - effectively compressing the universe.

Definition 1.4 (Sampling a distribution)**.** The notation we use for sampling from a distribution is $\sim$ 'simulate' (I prefer 'drawn from', but sometimes also called 'imagine' or 'hallucinate')

$$x \sim P \equiv p(x) \tag{10}$$

for example we may say $x \sim N(\mu, \sigma)$ meaning the data x follows the Gaussian distribution.

When you simulate from the model, you're basically drawing random numbers/examples according to that distribution.

Definition 1.5 (Cummulative distribution function)**.** The CDF is defined

$$\text{CDF}(x) = \int_{-\infty}^{x} \text{PDF}(z) dz. \tag{11}$$

This allows us to project a uniform distribution, which is easy to draw samples from, onto a more generic PDF.

(will draw diagram)

Definition 1.6 (Expectation)**.** The expectation of a PDF, given that all probabilities sum to 1, is like a weighted average of probabilities, showing the average outcome after an infinite number of trials

$$\mathbb{E}_{x\sim P}[x] = \int x \ \mathrm{PDF}(x)dx. \tag{12}$$

The expectation or expected value of some function $f(x)$ with respect to a probability distribution $P(x)$ is the mean value that f takes on when x is drawn from P

$$\mathbb{E}_{x\sim P}[f(x)] = \int f(x)P(x)dx, \tag{13}$$

and for the discrete case this is computed using a summation

$$\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x). \tag{14}$$

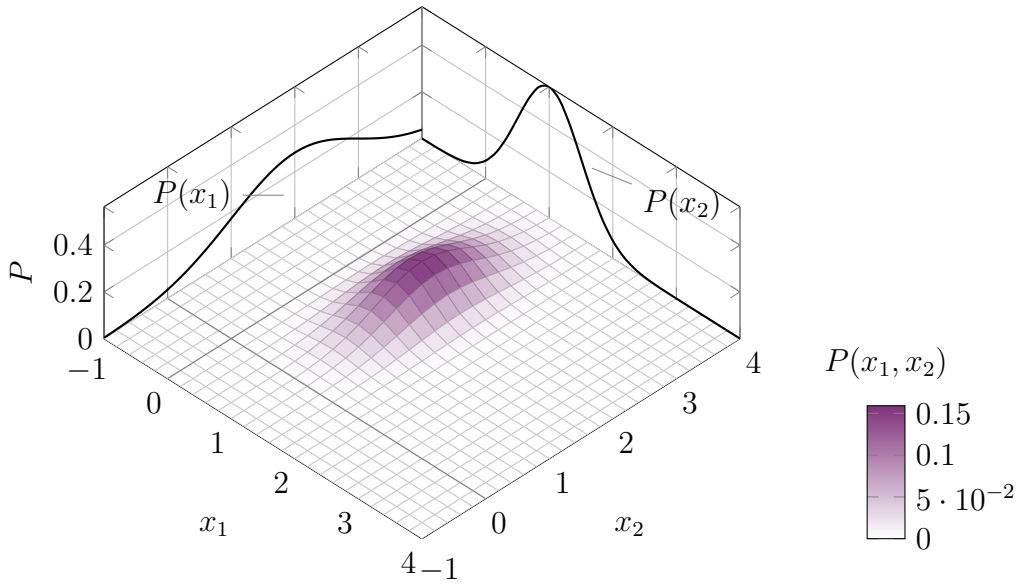
Question 1.1. *What is the expectation of a Gaussian distribution?*
Question 1.2. *What is the expected value of throwing a six-sided die drawn from a uniform distribution?*

Definition 1.7 (Joint probability density function)**.** The probability density function for two or more continuous random variables (bivariate $n = 2$ or multivariate) $f(s_1, s_2, \ldots, s_n)$ must satisfy

$$\int_{s_1} \cdots \int_{s_n} f(s_1, \ldots, s_n)\, ds_1 \ldots ds_n = 1, \tag{15}$$

and where the value of the distribution function is always greater than zero for any value of the random variables

$$f(s_1, s_2, ..., s_n) > 0. \tag{16}$$



1.1 Maximum likelihood estimation

Up until this point, we've looked up basic definitions and common estimators. What if we could learn a good estimator based on some principle from which we can derive good estimator functions. While there are many such principles, a common one is the maximum likelihood.

Definition 1.8 (Maximum likelihood)**.** This is where we try to find the optimal value for the distribution, given a bunch n of observed measures (observations $\mathbb{X} = \{x_1, x_2, ..., x_n\}$)

$$\begin{aligned} p(x) &= \arg\max_{\theta} p_{\text{model}}(\mathbb{X}; \theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{n} p_{\text{model}}(x_i; \theta) \end{aligned} \tag{17}$$

This product over the probabilities suffers numerical instability, so we take the logarithm to get a more workable sum

$$\begin{aligned} \theta_{\text{ML}} &= \arg\max_{\theta} \sum_{i=1}^{n} \log p_{\text{model}}(x_i; \theta) \\ &= \arg\max_{\theta} \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \left[\log p_{\text{model}}(x; \theta)\right] \end{aligned} \tag{18}$$

Definition 1.9 (Conditional log-likelihood)**.** This can be generalised to the case

of supervised learning, estimating the conditional probability $P(y \mid x; \theta)$

$$\begin{aligned} \theta_{\text{ML}} &= \arg\max_{\theta} P(Y \mid X; \theta) \\ &= \arg\max_{\theta} \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \left[\log P(y_i \mid x_i; \theta)\right] \end{aligned} \tag{19}$$

Definition 1.10 (PixelRNNs and PixelCNNs)**.** These similarly model the joint distribution of pixels over an image x as the product of conditional distributions, where x_i is a single pixel

$$p(x) \prod_{i=1}^{n^2} p(x_i \mid x_1, ..., x_{i-1}). \tag{20}$$

This needs an ordering of 'previous pixels' to be defined (very slow), and it is a complex distribution (can be expressed with a neural network). It would be better if we could decide on high-level characteristics first, rather than start at a pixel level. See how the model captures texture quite well but fails with global coherency:

1.2 Loss functions for density estimation

We've seen that designing a good density estimator typically has some bias/variance tradeoff problem.

How do we measure success of a density estimator? This means we need to choose a loss function $\mathcal{L}(p, \hat{p})$, from which there are many different options available.

Definition 1.11 (L_2 squared error loss function)**.** Perhaps not surprisingly the most common loss function

$$\mathcal{L}_2(p, \hat{p}) = \int (\hat{p}(x) - p(x))^2 dx, \tag{21}$$

which is easy to use and has a nice natural breakdown of bias and variance.

Definition 1.12 (L_1 loss function)**.** There's a book 'Nonparametric Density Estimation: The L_1 point of view' making the point that L_1 in main ways is 'the correct' loss function for density estimation.

$$\mathcal{L}_1(p, \hat{p}) = \int |\hat{p}(x) - p(x)| \ dx, \tag{22}$$

Luc Devroye and László Györfi highlight that

$$\mathcal{L}_1(p, \hat{p}) = 2 \cdot \mathrm{TV}(P, \hat{P}) \tag{23}$$

where

$$\mathrm{TV}(P, \hat{P}) = \sup_A |P(A) - Q(A)| \tag{24}$$

so the probabilities never differ by some value over any event. It also has a form of transformation invariance, where it doesn't matter what scale you're working on - you'll get the same answer. However in practice, deriving the smoothing parameters for L_1, while possible, in practice is very complicated and computationally expensive – so most people use L_2.

Definition 1.13 (Kullback-Leibler Divergence)**.** This is a very appealing non-symmetric distance with lots of nice theoretical properties (especially from information theory, and it comes up naturally from maximum likelihood)

$$\mathrm{KL}(P \parallel Q) = \int p(x) \log(\frac{p(x)}{q(x)}) dx \tag{25}$$

however it's a poor choice for density estimation (don't use it) as it's sensitive to the tails, which are irrelevant in density estimation – there's a lack of data for the tails in practice.

We can make a symmetric distance version by

$$\mathrm{D}(P, Q) = \mathrm{KL}(P \parallel Q) + \mathrm{KL}(Q \parallel P) \tag{26}$$

(sometimes divided by two).

Definition 1.14 (Cross entropy)**.** This relates to the KL divergence, but omitting the left term

$$\begin{aligned} H(P, Q) &= H(P) + \mathrm{KL}(P \parallel Q) \\ &= -\mathbb{E}_{x \sim P} \log Q(x). \end{aligned} \tag{27}$$

In summary there are lot's of ways (many more than explained here) to define distances between two different density estimators, each with varying properties and characteristics. For most practical applications, most people assume using L_2 for convenience and simplicity.

There are also lots of density estimators we can use (kernel density estimators, histograms, ...), each with various bias/variance tradeoffs.

Density estimation is an important difficult open research problem.

1.3 Generative adversarial networks

What if we stop trying to explicitly model the density, but focus on sampling? GANs take a game-theoretic approach to the problem, with two models:

1. A discriminator D (sometimes called a critic) that estimates the probability of a given sample coming from the real dataset or being a generated sample.
2. A generator function G that learns to map noise drawn from some prior distribution (typically Gaussian) to generated examples that capture p_{data}.

In other words the discriminator D tries to estimate a high probability for real data by maximizing $\mathbb{E}_{x\sim p_{\text{real}}(x)}[\log D(x)]$ and estimate a low probability close to zero for generated data by **maximizing** $\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$.

On the otherhand, the generator G is trained with the conflicting objective to **minimize** $\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$.

Definition 1.15 (Generative adversarial networks)**.** Putting this together we get a two-player minimax game with the value function $V(G,D)$

$$\begin{aligned}\min_G \min_G L(D,G) &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \\ &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_{\text{g}}(z)}[\log(1-D(x))].\end{aligned} \tag{28}$$

This is a zero-sum non-cooperative game (a zero-sum game is also called minimax). If one wins, the other looses. At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs $\frac{1}{2}$ everywhere. In game theory, the GAN model converges when G and D reach a Nash equilibrium.

In practice, training GANs is nontrivial due issues such as divergence in training and collapsed modes. In future lectures, we will further study their characteristics and various extensions that are required before being able to generate hyperrealistic images such as in the following example. The field incorporates concepts from semi-supervised learning (making use of class labels), attention, and is also part of a big compute arms race.