Deep Probabilistic Models

Deep Learning 2 – 2022

Wilker Aziz
w.aziz@uva.nl

University of Amsterdam
Institute for Logic, Language and Computation

Hello

I am an assistant professor at the Institute for Logic, Language and Computation (University of Amsterdam). You can check some of my work here https://probabll.github.io

Stuff I typically work on include

- machine learning
  approximate inference, gradient estimation
  for deep latent variable models, VAEs, normalising flows
- natural language processing
  translation, text classification, question answering, transparent and interpretable models

Goals for this session

1. Recognise a probabilistic model
2. Think of models as tools to reproduce observed statistical patterns
3. Recognise that many DL models are statistical models
4. Recognise that datasets are recorded random experiments
5. Prescribe joint distributions over observed random variables
6. Parameterise distributions using neural networks
7. Estimate parameters via SGD
8. Use a probabilistic model of data to approach a task

Outline

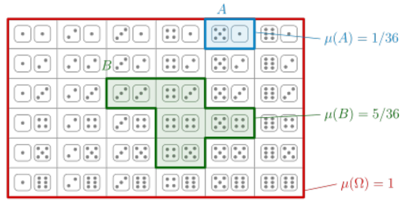
Probabilistic Models

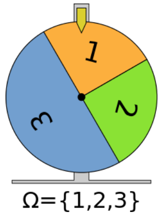
A probabilistic model *prescribes* the probability measure of a random experiment.

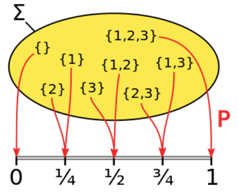
Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0, 1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

Probabilistic Models

A probabilistic model *prescribes* the probability measure of a random experiment.







Figures from Wikimedia (CC-ASA-4.0)

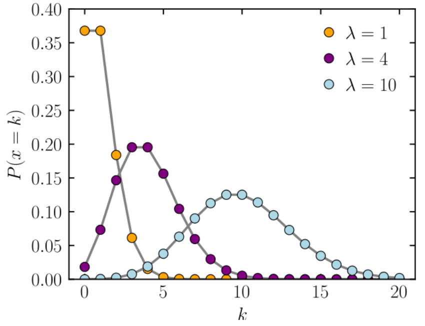
Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0, 1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

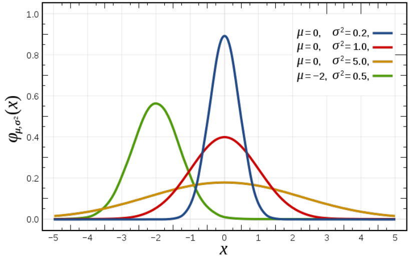
There are so many ways in which we can prescribe a probability measure:

- We may specify the probability of events in the event space, one at a time. This is very tedious and sometimes impossible (e.g., all subsets of natural numbers).

Probabilistic Models

A probabilistic model *prescribes* the probability measure of a random experiment.





Figures from Wikimedia (CC-ASA-4.0)

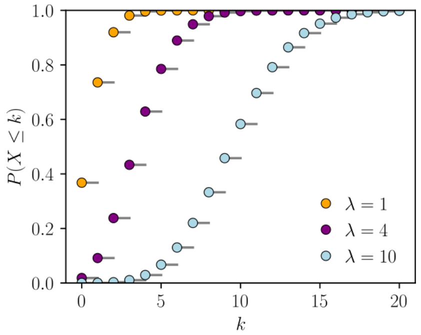
Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0, 1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

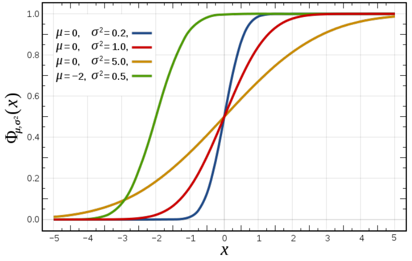
There are so many ways in which we can prescribe a probability measure:

- We may specify the probability of events in the event space, one at a time. This is very tedious and sometimes impossible (e.g., all subsets of natural numbers).
- We may instead specify a probability mass or density function (pmf or pdf) for outcomes of a random variable $X : \Omega \to \mathcal{X} \subseteq \mathbb{R}$. The rv and its pdf in turn identify a probability measure.

Probabilistic Models

A probabilistic model *prescribes* the probability measure of a random experiment.





Figures from Wikimedia (CC-ASA-4.0)

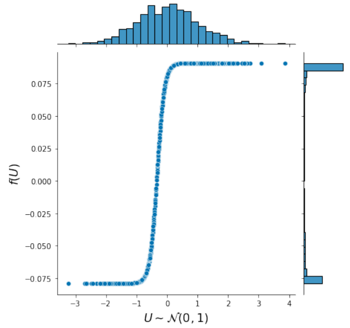
Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0, 1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

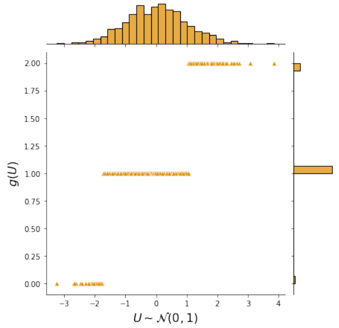
There are so many ways in which we can prescribe a probability measure:

- We may specify the probability of events in the event space, one at a time. This is very tedious and sometimes impossible (e.g., all subsets of natural numbers).
- We may instead specify a probability mass or density function (pmf or pdf) for outcomes of a random variable $X : \Omega \to \mathcal{X} \subseteq \mathbb{R}$. The rv and its pdf in turn identify a probability measure.
- We may instead specify the cumulative distribution function (cdf) of an rv, the cdf in turn identifies a pdf, the rv and its pdf identify a probability measure.

Probabilistic Models

A probabilistic model *prescribes* the probability measure of a random experiment.





$$f(u) = 0.1\tanh(4.2u + 1.4) \text{ and } g(u) = \begin{cases} 0 & \text{if } f(u) < -0.5 \\ 2 & \text{if } f(u) > 0.5 \\ 1 & \text{otherwise} \end{cases}$$

Figures from Wikimedia (CC-ASA-4.0)

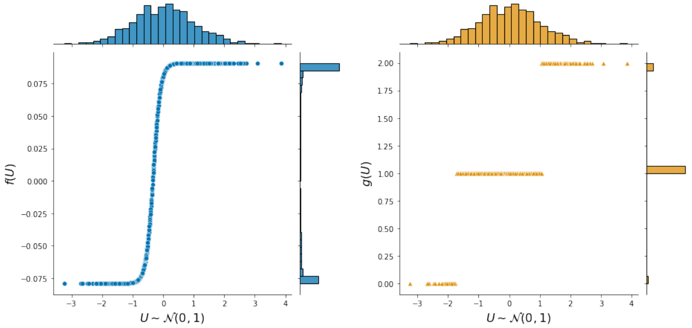
Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0,1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i\in I} E_i) = \sum_{i\in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

There are so many ways in which we can prescribe a probability measure:

- We may specify the probability of events in the event space, one at a time. This is very tedious and sometimes impossible (e.g., all subsets of natural numbers).
- We may instead specify a probability mass or density function (pmf or pdf) for outcomes of a random variable $X : \Omega \to \mathcal{X} \subseteq \mathbb{R}$. The rv and its pdf in turn identify a probability measure.
- We may instead specify the cumulative distribution function (cdf) of an rv, the cdf in turn identifies a pdf, the rv and its pdf identify a probability measure.
- We may specify a simulator (e.g., a function from `rand()` to outcomes of an rv), the simulator identifies an inverse cdf, which in turn ...

Probabilistic Models

A probabilistic model *prescribes* the probability measure of a random experiment.



$$f(u) = 0.1\tanh(4.2u + 1.4) \text{ and } g(u) = \begin{cases} 0 & \text{if } f(u) < -0.5 \\ 2 & \text{if } f(u) > 0.5 \\ 1 & \text{otherwise} \end{cases}$$

Figures from Wikimedia (CC-ASA-4.0)

Random experiment: a sample space Ω, an event space $\Sigma = \mathcal{P}(\Omega)$, and a probability measure $\mathbb{P} : \Sigma \to [0,1]$ where $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\cup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for collections $\{E_i\}$ of pairwise disjoint events.

There are so many ways in which we can prescribe a probability measure:

- We may specify the probability of events in the event space, one at a time. This is very tedious and sometimes impossible (e.g., all subsets of natural numbers).
- We may instead specify a probability mass or density function (pmf or pdf) for outcomes of a random variable $X : \Omega \to \mathcal{X} \subseteq \mathbb{R}$. The rv and its pdf in turn identify a probability measure.
- We may instead specify the cumulative distribution function (cdf) of an rv, the cdf in turn identifies a pdf, the rv and its pdf identify a probability measure.
- We may specify a simulator (e.g., a function from `rand()` to outcomes of an rv), the simulator identifies an inverse cdf, which in turn ...

Probabilistic Modelling and Reasoning

Probabilistic modelling concerns the specification of a joint distribution over random variables of interest.

Probabilistic reasoning concerns fixing a subset of these random variables to some observations and inferring marginal and conditional distributions by application of probability calculus.

The latter is also know as *probabilistic inference*.

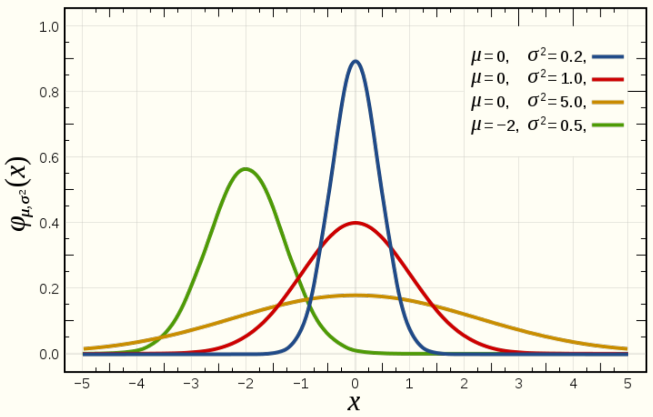
Notation. Capital letters for rvs (e.g., X, Y), lowercase letters for assignments (e.g., $X = x$, $Y = y$), calligraphic letters for range of rvs (e.g., $\mathcal{X}$, $\mathcal{Y}$). I use p_X for the pdf of X and F_X for its cdf. When needed I show the dependency of the probability density on a parameter θ as follows: $p_X(x|\theta)$.

Probability calculus recap. Chain rule $p_{XY}(x, y) = p_X(x)p_{Y|X}(y|x) = p_Y(y)p_{X|Y}(x|y)$. Conditional probability $p_{Y|X}(y|x) = \frac{p_{XY}(x,y)}{p_X(x)}$. Marginalisation $p_X(x) = \int_{\mathcal{Y}} p_{XY}(x, y)\, \mathrm{d}y$.

If you would like to learn *all* about probabilistic graphical models (PGMs), check the excellent book by Koller and Friedman (2009). I'd recommend Part I (on representation of distributions) to *anyone*.

Learning

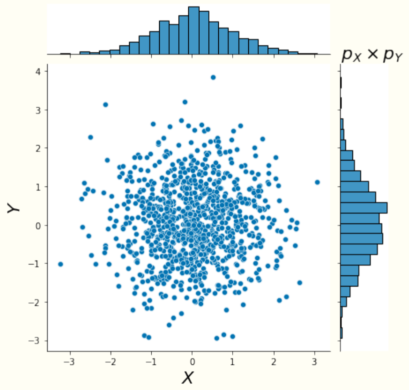
Oftentimes, we begin specifying a probability distribution by specifying a class of distributions (e.g., Normal, Exponential, Categorical).



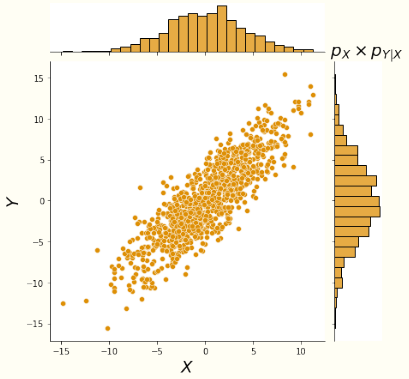
Learning

Oftentimes, we begin specifying a probability distribution by specifying a class of distributions (e.g., Normal, Exponential, Categorical).

For multivariate data, we also choose a factorisation of the joint distribution.



$\mathcal{N}(x|u_1, s_1^2)\mathcal{N}(y|u_2, s_2^2)$



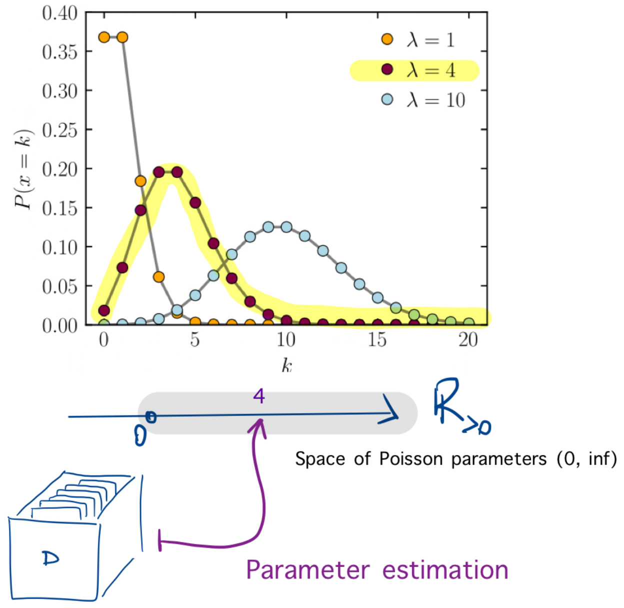
$\mathcal{N}(x|u_1, s_1^2)\mathcal{N}(y|\mu(x), \sigma(x)^2)$

Learning

Oftentimes, we begin specifying a probability distribution by specifying a class of distributions (e.g., Normal, Exponential, Categorical).

For multivariate data, we also choose a factorisation of the joint distribution.

Parameter estimation (e.g., maximum likelihood estimation) singles out a member of the class (e.g., $\mathcal{N}(2, 1)$, $\mathrm{Exponential}(10)$, $\mathrm{Cat}(0.1, 0.2, 0.7)$).

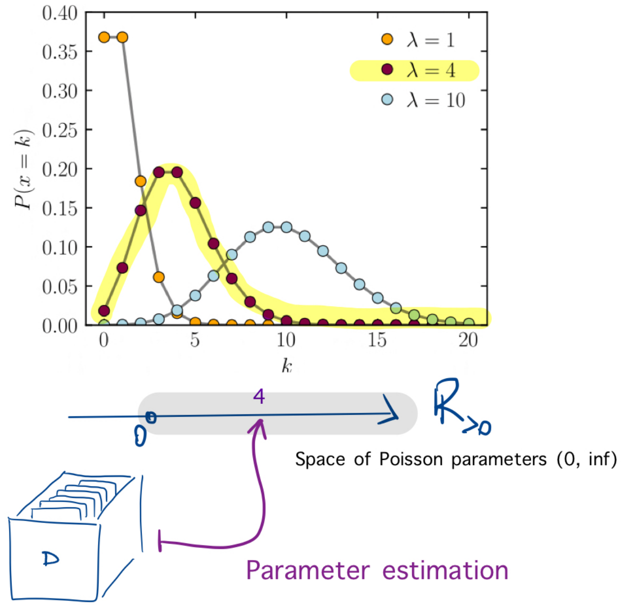


Learning

Oftentimes, we begin specifying a probability distribution by specifying a class of distributions (e.g., Normal, Exponential, Categorical).

For multivariate data, we also choose a factorisation of the joint distribution.

Parameter estimation (e.g., maximum likelihood estimation) singles out a member of the class (e.g., $\mathcal{N}(2,1)$, $\mathrm{Exponential}(10)$, $\mathrm{Cat}(0.1, 0.2, 0.7)$).



Deep Learning

Most modern ML models, including DL models, are probabilistic.

Example: a neural language model factorises the probability of a sequence S using chain rule. For each step of the sequence from left to right, it assigns probability for a word W given its complete history H, i.e.,

$$p_S(\langle x_1, \ldots, x_I \rangle | \theta) = \prod_{i=1}^{I} p_{W|H}(x_i | x_{<i}, \theta)$$

where each conditional is a Categorical distribution predicted by an NN:

$$W|H = x_{<i} \sim \mathrm{Cat}(\mathbf{f}(x_{<i}; \theta))$$

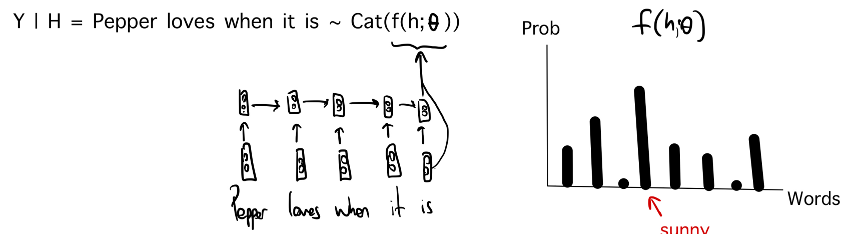
Typical algorithms for parameter estimation require assessing

$$p_{W|H}(k|x_{<i}, \theta) = f_k(x_{<i}; \theta)$$

and its gradient, which take only a forward and a backward pass through a tractable computation graph.

Deep Learning

Most modern ML models, including DL models, are probabilistic.



Deep Learning

Most modern ML models, including DL models, are probabilistic.

This is sometimes not obvious because DL literature often emphasises implementation and algorithmic aspects over statistical considerations about the data and the model.

Example: we use an LSTM decoder with input feeding and at each time step the model is trained to predict the next target word via teacher forcing, the model is trained to minimize the categorical cross entropy loss.

Deep Learning

Most modern ML models, including DL models, are probabilistic.

This is sometimes not obvious because DL literature often emphasises implementation and algorithmic aspects over statistical considerations about the data and the model.

Example: we use an LSTM decoder with input feeding and at each time step the model is trained to predict the next target word via teacher forcing, the model is trained to minimize the categorical cross entropy loss.

Zooming in and out

Implementation aspects are important for obvious reasons: ultimately one has to implement a model that works! This requires being concrete about

- the design of NN architecture blocks (e.g., encoder, decoder)
- how to wire blocks together (e.g., residual connections)
- how to train the model (e.g., objective, curriculum, optimiser)
- and how to make predictions (e.g., beam search, sampling).

Many architectural differences have little statistical substance and abstracting away from them helps us concentrate on issues that are statistical in nature.

Example I: LSTMs are superior to vanilla RNNs for they address numerical problems with repeated application of the chain rule of derivatives. That said, from a statistical standpoint, LSTMs and vanilla RNNs play interchangeable roles (i.e., allow to condition on a sequence of arbitrary length).

Example II: Dropout addresses overfitting in FFNNs, but leads to catastrophic forgetting in recurrent models. From a probabilistic perspective, we can explain why dropout works and how to 'fix it' to support the recurrent case.

Dropout (Srivastava et al., 2014) is a technique where we randomly set some inputs of a layer to 0. A special type of approximations to a Bayesian model corresponds very closely to dropout (Gal and Ghahramani, 2016b) and leads to extensions to more complex architecture blocks (Gal and Ghahramani, 2016a,c).

What are some advantages of probabilistic models?

Probabilistic models allows to incorporate assumptions through

- the choice of distribution
- dependencies among random variables
- the way that distributions uses side information
- stipulate unobserved data and their properties

They return a distribution over outcomes which can be used to

- generate data
- account for unobserved data
- provide explanation and suggest improvements
- inform decision makers

Outline

Modelling random experiments

We treat *data* as outcomes of experiments involving random variables.

A *model* of the data prescribes a distribution for those random variables. Ideally, one that is faithful to statistical properties of our observations. Applications:

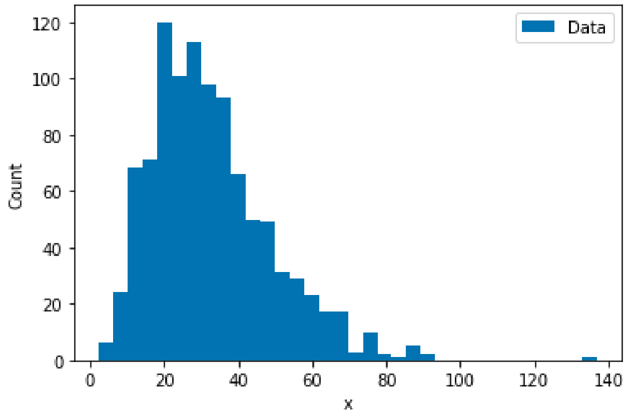
- reveal structure hidden in existing data;
- support decisions about existing and future data.

The main subject of statistical interest is data (as opposed to tasks). Think of a task as a potential application of a (good) model of the data.

Modelling data does not imply solving a predictive task.

For example, a generative classifier is built upon a joint pdf $p_Y(y)p_{X|Y}(x|y)$ over labels $y \in \mathcal{Y}$ and inputs $x \in \mathcal{X}$. Making a specific prediction for a novel input x_* is a decision problem, oftentimes handled independently of model specification and learning. A common decision rule for classification is $y_* = \arg\max_y p_{Y|X}(y|x_*)$.

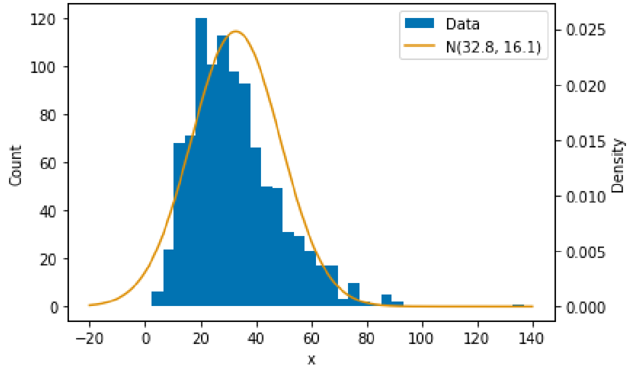
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

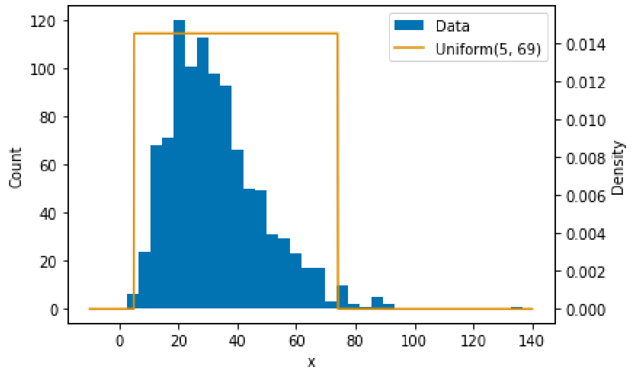
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

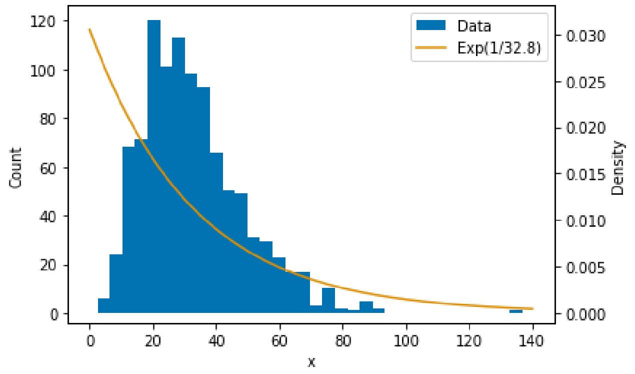
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

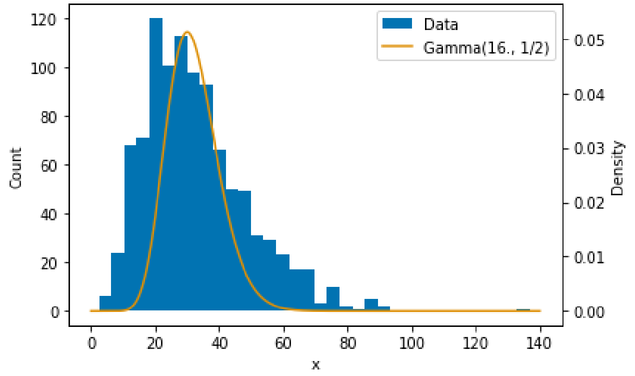
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

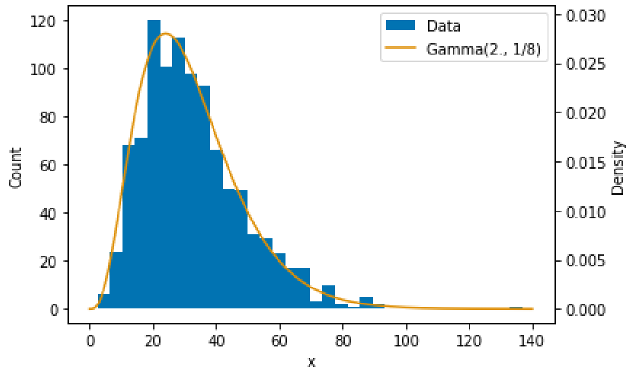
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

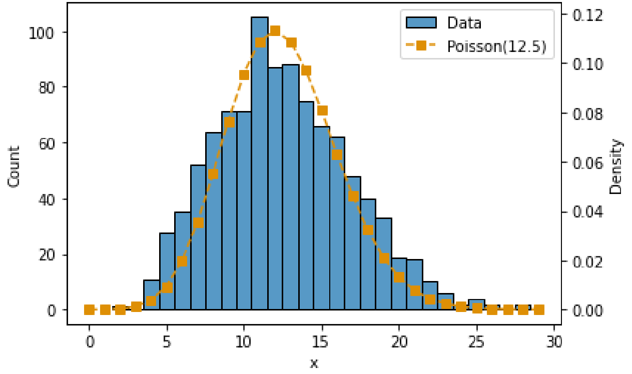
Faithfulness



Consider the data in the example

- the measurements are continuous and positive
- the sample mean is close to 32
- the sample stddev is close to 16
- they concentrate around a single value (unimodal)
- they stretch to the right (skew)

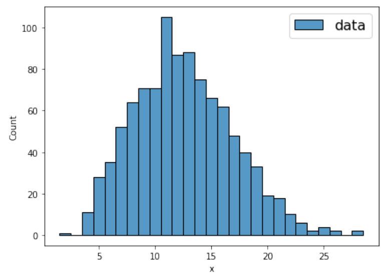
Hidden structure

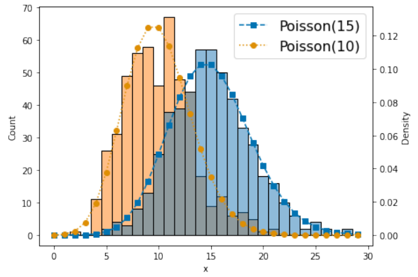


Here the measurements are natural numbers, the sample mean is close to 12.5 and the median is 12.

A Poisson distribution can capture the mean, but not the spread (recall that the Poisson mean and variance are equal).

Hidden structure



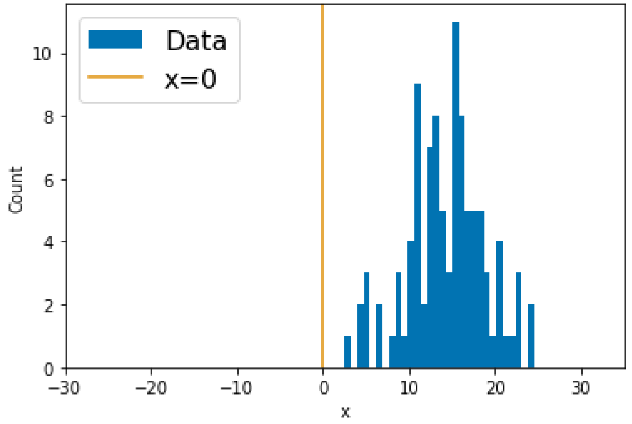


A different probabilistic model may reveal the presence of two groups mixed in a single population.

Here the measurements are natural numbers, the sample mean is close to 12.5 and the median is 12.

A Poisson distribution can capture the mean, but not the spread (recall that the Poisson mean and variance are equal).

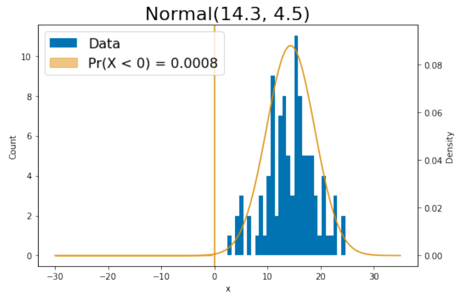
Decisions about future data

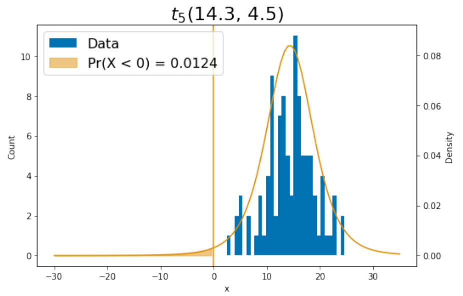


Here we observe continuous measurements from a sensor in a car. Data come in in batches of 100 measurements.

Suppose that if 1% (or more) of the readings drop below 0, the driver is at risk.

Decisions about future data





The heavier tails of the Student's t reserve much more probability for unseen data.

Here we observe continuous measurements from a sensor in a car. Data come in in batches of 100 measurements.

Suppose that if 1% (or more) of the readings drop below 0, the driver is at risk.

Modelling observed random variables

Our goal is to learn a distribution over a set of **observed** random variables.

Observed random variables are the result of random experiments that have already happened: e.g., sentences in a collection of news articles, number of stars in a product review.

Modelling observed random variables

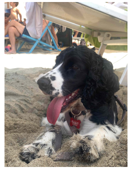
Our goal is to learn a distribution over a set of **observed** random variables.

Observed random variables are the result of random experiments that have already happened: e.g., sentences in a collection of news articles, number of stars in a product review.

Typical use in ML: *conditional models.*
▷ *We are given some variables (inputs) and we are interested in making predictions about other variables (outputs)*

- such inputs are also called *predictors* (or *covariates*)
- with some probability, *predicted by the model*, an output takes on a certain *outcome* in a sample space

Modelling conditionally - Examples

Predictor	**Outcome**	**Sample space**
Why did they bother recording this???	$\star$	$\{\star, \star\star, \star\star\star, \star\star\star\star, \star\star\star\star\star\}$
Source: geen standaard MT: no standard	compare('no step')=0.5	$[0, 1]$
he proposed a famous solution to an inverse probability problem in the 18th century	https://en.wikipedia.org/wiki/Thomas_Bayes	$\mathcal{W}_{\text{en}}$
	*Pepper loves the beach!*	Σ^*_{en}
That's not possible!	*Dat is niet mogelijk!*	Σ^*_{nl}

1. text classification
2. machine translation quality estimation
3. question answering
4. image captioning
5. machine translation

To model or not to model?

Oftentimes, a model sees some observations as *deterministic* predictors. These are never *modelled*, they are only *conditioned on*:

- if a variable is not modelled, our statistical model cannot assign a probability to any observed value of that variable nor generate random draws for that variable
- the variable can, however, be used in some calculation

Example: a review in a sentiment classifier

- some NN "reads it" to compute a probability distribution over sentiment levels (e.g., negative, neutral, positive)
- the model has no clue how reviews come about, it is only concerned with reading them, not modelling their generative process

Deterministic predictors are usually the kinds of input we talk about when we mean 'inputs at test time' (e.g., the source sentence in MT).

An example of stochastic predictor is the prefix of already generated words in a partial translation. The model can assign probability to those (for example, in training), and it might even have generated those (for example, in test). Once those outcomes are already in place, they act as predictors so we can continue translating.

What if we model all observations?

That is, including the predictors. Then we get *joint* or *generative models*.

Think of these models as models that can generate their own predictors with some probability.

All previous examples can be modelled generatively.

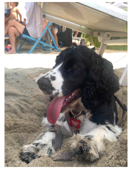
In some contexts, especially classification and regression, conditional models are called discriminative models.

Generative models are sometimes also called joint models.

Don't get too caught up with the generative vs discriminative debate. Again, different points of view will lead to different uses of these labels. Some people will read 'discriminative' as something about the training algorithm, others as something about the nature of the distribution, others just appeal to analogies or traditions, there are also cases where things get tricky because of other disciplines (e.g., generative syntax in linguistics has nothing to do with statistics, though generative syntactic formalisms can be given statistical treatment, including via discriminative models).

For us, these are all *probabilistic* (or statistical) models.

Joint modelling - Examples

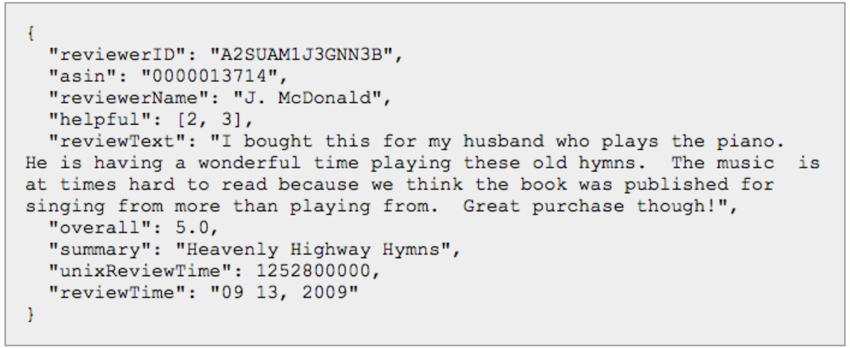
Joint outcome		Sample space
Why did they bother recording this???	$\star$	$\Sigma_{\text{en}}^* \times \{\star, \star\star, \star\star\star, \star\star\star\star, \star\star\star\star\star\}$
Source: geen standaard MT: no standard	compare('no step')=0.5	$\Sigma_{\text{nl}}^* \times \Sigma_{\text{en}}^* \times [0, 1]$
he proposed a famous solution to an inverse probability problem in the 18th century	https://en.wikipedia.org/wiki/Thomas_Bayes	$\Sigma_{\text{en}}^* \times \mathcal{W}_{\text{en}}$
	*Pepper loves the beach!*	$[256, 256, 256]^{h\times w} \times \Sigma_{\text{en}}^*$
That's not possible!	*Dat is niet mogelijk!*	$\Sigma_{\text{en}}^* \times \Sigma_{\text{nl}}^*$

The statistical model predicts a distribution over the sample space. Sample spaces can grow rather large, especially so when former deterministic predictors are to be treated as random variables (that is, they are now part of the random experiment we aim to predict).

Outline

Example - Music reviews

I downloaded some reviews from Amazon, this is what a datum looks like

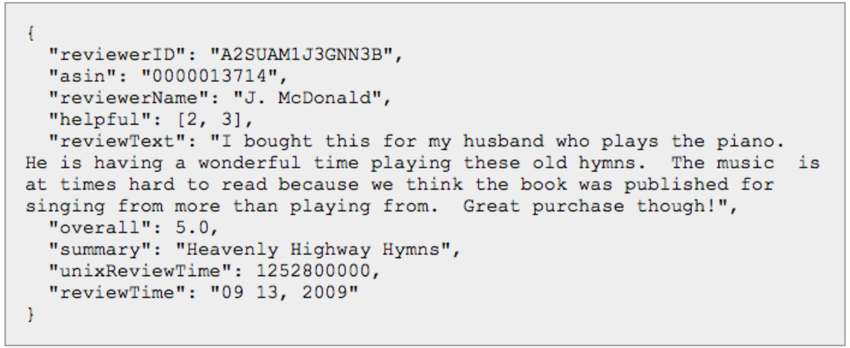


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

We can observe the outcomes of many random variables here.
But do we care about all of them?

Example - Music reviews

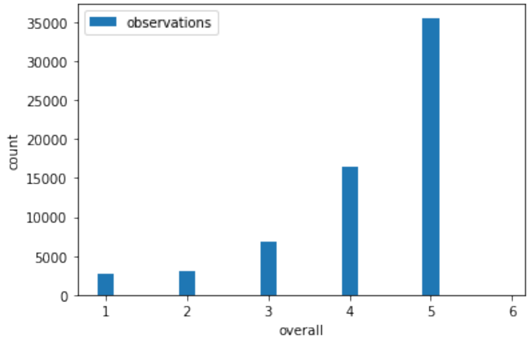
I downloaded some reviews from Amazon, this is what a datum looks like



```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

Let's say we care about outcomes of *overall* score and let's visualise the observations available.

Example - Visualise data



Let's say we take the overall score assigned to any one review as a random variable (and ignore everything else in the dataset).

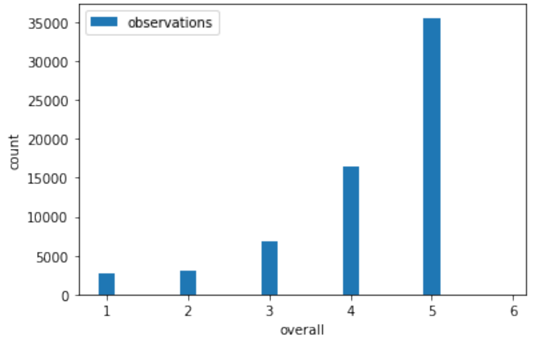
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



Can we capture the general pattern by prescribing some known probability distribution?

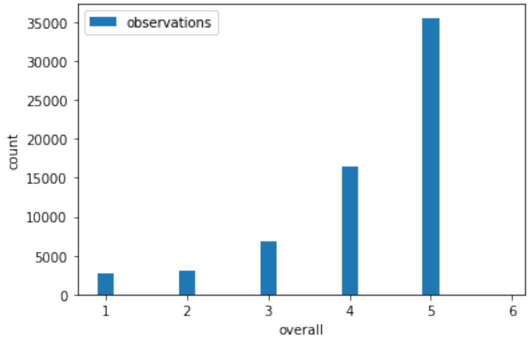
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



Let's say that overall scores are drawn from some Categorical distribution. That's fine, but which one?

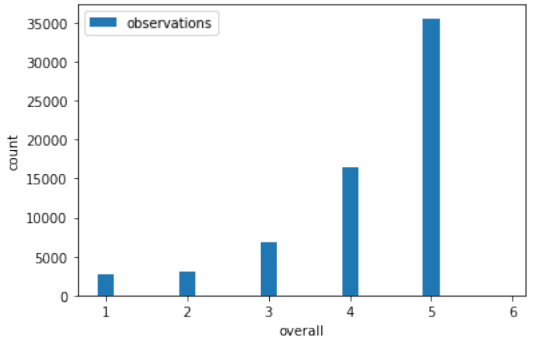
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



A member of the Categorical family of distributions is specified by a parameter: a vector of dense positive values whose elements sum to 1.

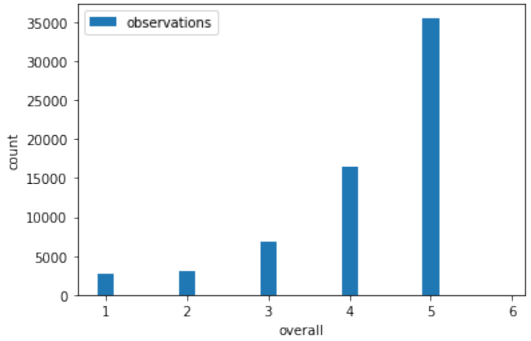
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



Let's pick the one that makes our dataset as probable as possible. And let's assume that our observations were obtained independently.

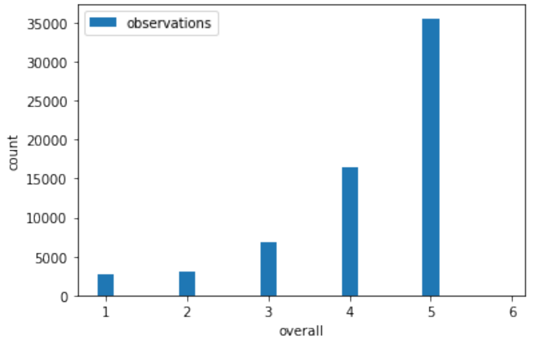
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



We are making an i.i.d. assumption and we have chosen a parametric family for the model: $Y_1 \sim Y_2 \sim \cdots \sim Y_N \sim \text{Cat}(\phi)$ with $\phi \in \Delta_{5-1}$.

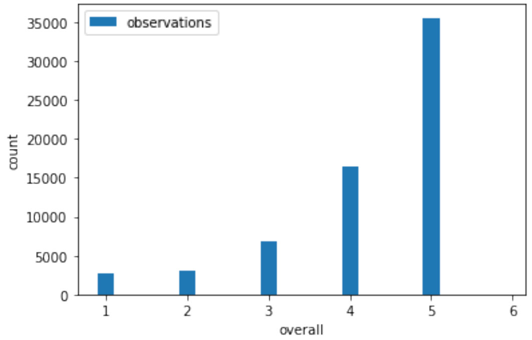
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



Thus $L_{y_{1:N}}(\phi) = p(y_{1:N}|\phi) = \prod_{i=1}^{N} \text{Cat}(y_i|\phi) = \prod_{i=1}^{N} \phi_{y_i}$ is the likelihood of ϕ given the observed data $y_{1:N}$.

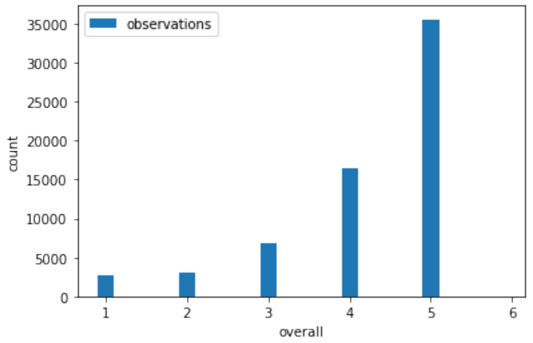
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



MLE tells us to pick $\phi^\star$ such that
$\phi^\star = \arg\max_\phi L_{y_{1:N}}(\phi)$.

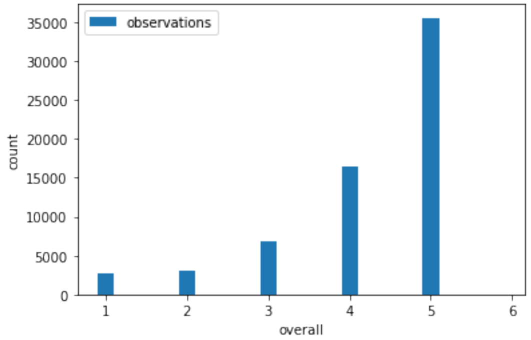
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



We call $L_{y_{1:N}}(\phi)$ the *likelihood function*, it is a function of ϕ for fixed data and model family.

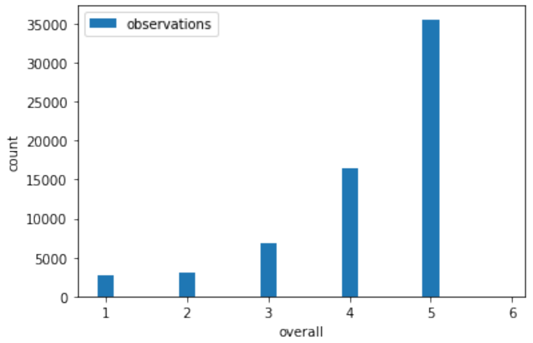
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



We can equivalently search for ϕ under the log-likelihood function $\mathcal{L}_{y_{1:N}}(\phi) = \log L_{y_{1:N}}(\phi)$ and solve $\arg\max_\phi \mathcal{L}_{y_{1:N}}(\phi)$.

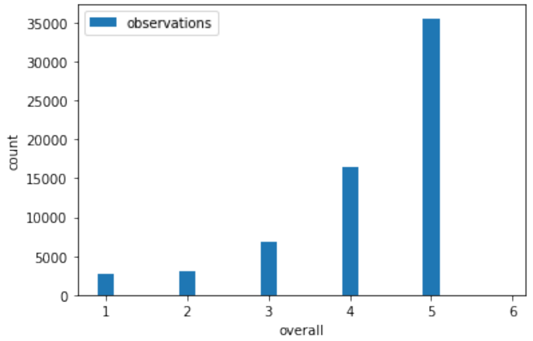
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Visualise data



If you solve this, you will see that the *maximum likelihood estimate* of the Categorical distribution is given by dividing the bars in the plot by N.

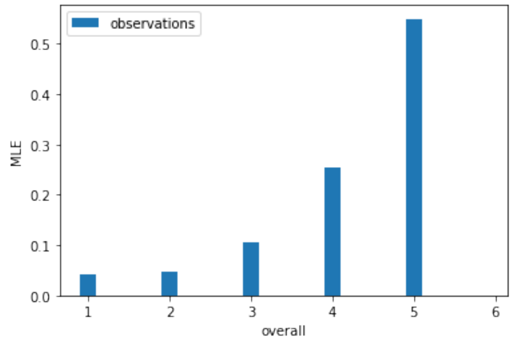
The first thing to note is that we can observe many outcomes.

The second thing to note is that we can capture the general pattern using a probability distribution.

For example, assume that overall scores are drawn from some Categorical distribution. Which one? One option is to pick the one that makes these observations as probable as possible. Or, equivalently, given the data and the model family (Categorical), we pick the most likely parameter.

To derive an exact MLE for a Categorical likelihood you will need some proficiency with partial derivatives and Lagrangian multipliers. It's okay to also just find the MLE on Wikipedia or in a textbook.

Example - Fit a Categorical likelihood model by MLE



This clearly captures the exact pattern we saw before. Does this level of analysis meet the expectation of our target audience or do we need a more fine grained picture?

Whether we ‘meet the expectation of the audience’ is a matter of application. From the point of view of the statistical model, the job is done.

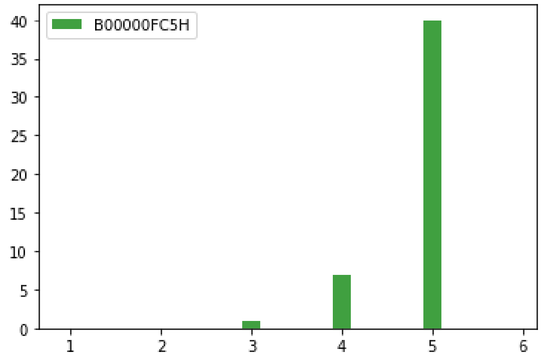
If the user is interested in learning more about products, buyers, and their relationship then this is probably insufficient.

For example, can we get a more fine grained picture if we condition on some predictors?

Recall, the review record contained a lot more information.

Example - Conditioning on predictors

Let's see how overall scores distribute for some **products**



There is a certain amount of variance that is characteristic of how people feel about the product. Modelling the data means modelling this variance, not ignoring it!

Can we say the first product is highly appreciated? Can we say the opposite about the second one?

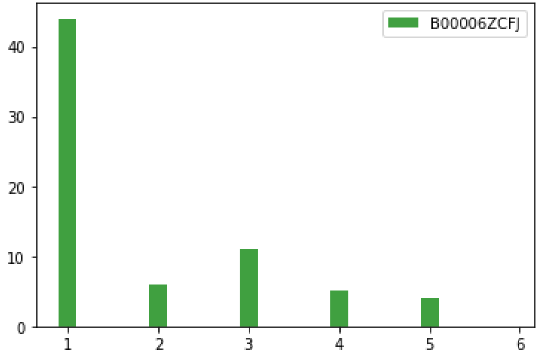
What can be said about the 3rd and the 4th?

Generally, what can be said is that committing to any overall score hides variance present in the data.

Whereas for some purposes we may have to choose a single score for a product, from the point of view of the statistical model that is not at all the goal. The goal is to model the data well, that is, closely reproducing statistical properties of the observed data (mean, variance, skew, ...).

Example - Conditioning on predictors

Let's see how overall scores distribute for some **products**



There is a certain amount of variance that is characteristic of how people feel about the product. Modelling the data means modelling this variance, not ignoring it!

Can we say the first product is highly appreciated? Can we say the opposite about the second one?

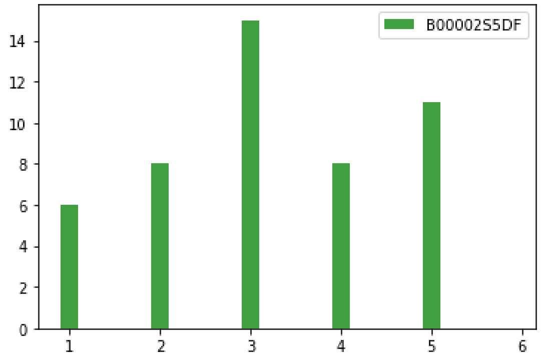
What can be said about the 3rd and the 4th?

Generally, what can be said is that committing to any overall score hides variance present in the data.

Whereas for some purposes we may have to choose a single score for a product, from the point of view of the statistical model that is not at all the goal. The goal is to model the data well, that is, closely reproducing statistical properties of the observed data (mean, variance, skew, ...).

Example - Conditioning on predictors

Let's see how overall scores distribute for some **products**



There is a certain amount of variance that is characteristic of how people feel about the product. Modelling the data means modelling this variance, not ignoring it!

Can we say the first product is highly appreciated? Can we say the opposite about the second one?

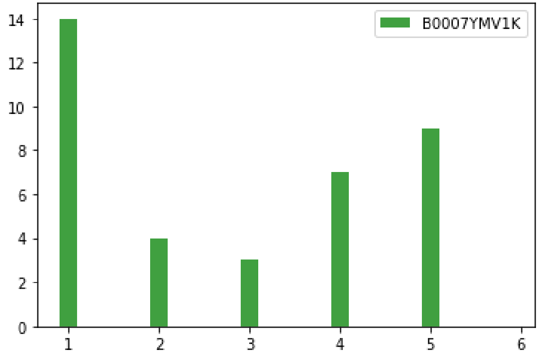
What can be said about the 3rd and the 4th?

Generally, what can be said is that committing to any overall score hides variance present in the data.

Whereas for some purposes we may have to choose a single score for a product, from the point of view of the statistical model that is not at all the goal. The goal is to model the data well, that is, closely reproducing statistical properties of the observed data (mean, variance, skew, ...).

Example - Conditioning on predictors

Let's see how overall scores distribute for some **products**



There is a certain amount of variance that is characteristic of how people feel about the product. Modelling the data means modelling this variance, not ignoring it!

Can we say the first product is highly appreciated? Can we say the opposite about the second one?

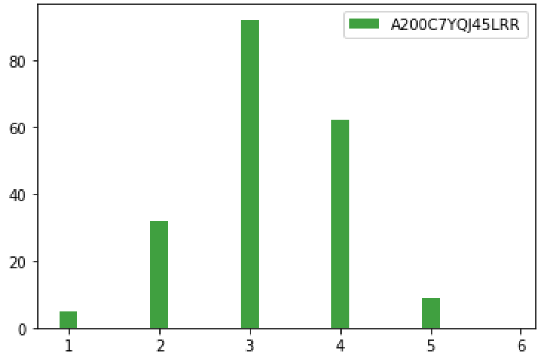
What can be said about the 3rd and the 4th?

Generally, what can be said is that committing to any overall score hides variance present in the data.

Whereas for some purposes we may have to choose a single score for a product, from the point of view of the statistical model that is not at all the goal. The goal is to model the data well, that is, closely reproducing statistical properties of the observed data (mean, variance, skew, ...).

Example - Conditioning on predictors

Let's see how overall scores distribute for some **reviewers**



There is a certain amount of variance that is typical of a reviewer (e.g., some people might not bother leaving a review unless they have a strong opinion).

Note that no matter how we view the data, we still find variance.

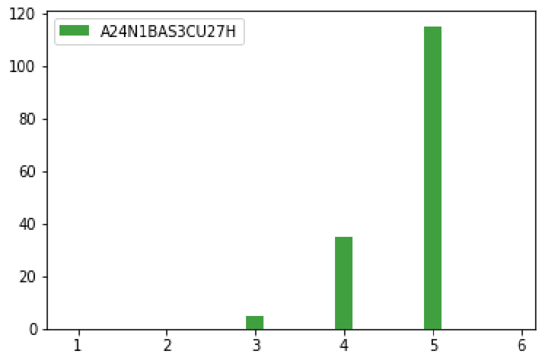
Explanation 1: *the data is the data* is the data.

Explanation 2: noisy data.

We need to agree that *noisy data* is *the data*. Everything you observe gets to be called *data*. There is no such a thing as noisy-free data and no such a thing as *outliers*. If you get a dataset and *remove outliers* you essentially created a different dataset that is artificially simpler. Besides, in this application, you would be saying some people's opinions don't matter.

Example - Conditioning on predictors

Let's see how overall scores distribute for some **reviewers**



There is a certain amount of variance that is typical of a reviewer (e.g., some people might not bother leaving a review unless they have a strong opinion).

Note that no matter how we view the data, we still find variance.

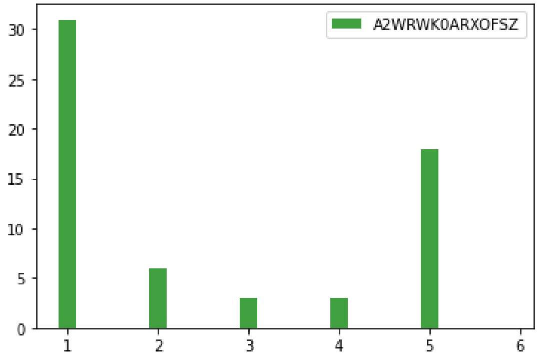
Explanation 1: *the data is the data* is the data.

Explanation 2: noisy data.

We need to agree that *noisy data* is *the data*. Everything you observe gets to be called *data*. There is no such a thing as noisy-free data and no such a thing as *outliers*. If you get a dataset and *remove outliers* you essentially created a different dataset that is artificially simpler. Besides, in this application, you would be saying some people's opinions don't matter.

Example - Conditioning on predictors

Let's see how overall scores distribute for some **reviewers**



There is a certain amount of variance that is typical of a reviewer (e.g., some people might not bother leaving a review unless they have a strong opinion).

Note that no matter how we view the data, we still find variance.

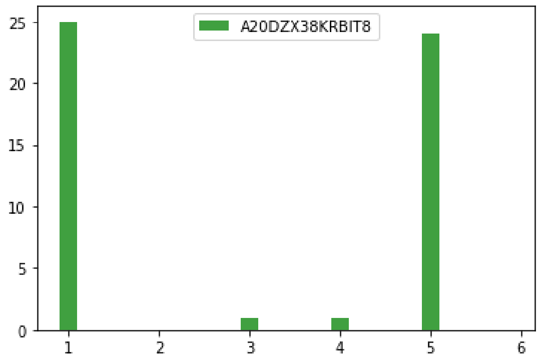
Explanation 1: *the data is the data* is the data.

Explanation 2: noisy data.

We need to agree that *noisy data* is *the data*. Everything you observe gets to be called *data*. There is no such a thing as noisy-free data and no such a thing as *outliers*. If you get a dataset and *remove outliers* you essentially created a different dataset that is artificially simpler. Besides, in this application, you would be saying some people's opinions don't matter.

Example - Conditioning on predictors

Let's see how overall scores distribute for some **reviewers**



There is a certain amount of variance that is typical of a reviewer (e.g., some people might not bother leaving a review unless they have a strong opinion).

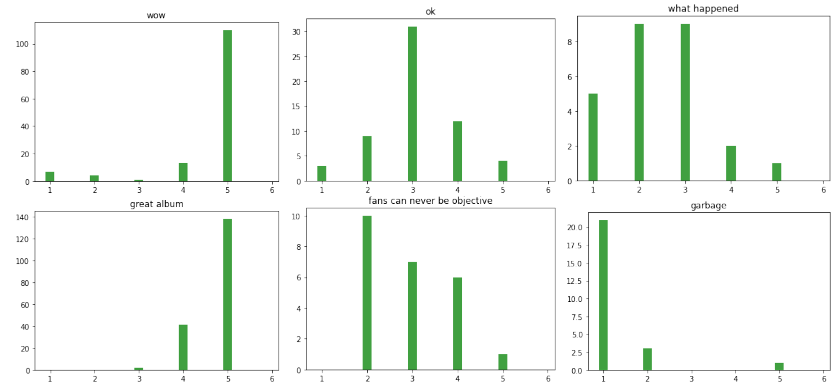
Note that no matter how we view the data, we still find variance.

Explanation 1: *the data is the data* is the data.

Explanation 2: noisy data.

We need to agree that *noisy data* is *the data*. Everything you observe gets to be called *data*. There is no such a thing as noisy-free data and no such a thing as *outliers*. If you get a dataset and *remove outliers* you essentially created a different dataset that is artificially simpler. Besides, in this application, you would be saying some people's opinions don't matter.

Example - Conditioning on high-dimensional predictors



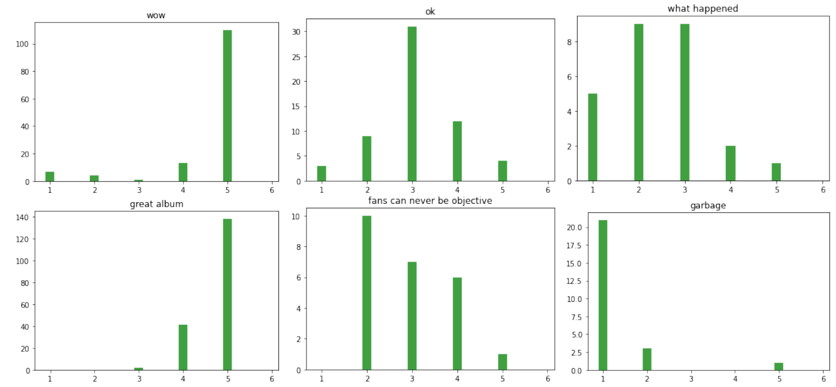
Reviewers contribute long reviews, but also short summaries. These are short enough that we can gather more than 20 different reviews per summary.

Top-left: wow
Top-middle: ok
Top-right: what happened
Bottom-left: great album
Bottom-middle: fans can never be objective
Bottom-right: garbage

Some remarks:

- note the trends are quite different from the general trend
- note that we cannot always be very confident about the overall score

Example - Conditioning on high-dimensional predictors



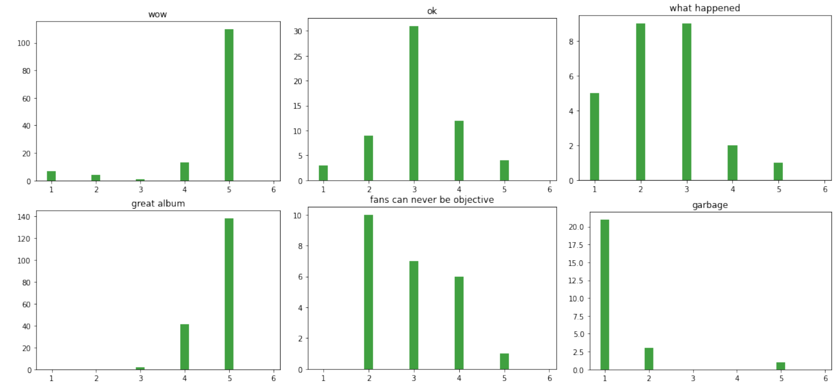
We could model the overall score Y given a summary $X = s$ as a draw from the summary-specific Categorical distribution $\text{Cat}(\phi^{(s)})$.

Top-left: wow
Top-middle: ok
Top-right: what happened
Bottom-left: great album
Bottom-middle: fans can never be objective
Bottom-right: garbage

Some remarks:

- note the trends are quite different from the general trend
- note that we cannot always be very confident about the overall score

Example - Conditioning on high-dimensional predictors



But then we would have to estimate as many Categorical distributions as there are unique summaries. Clearly we will struggle in the future, when a novel summary pops up.

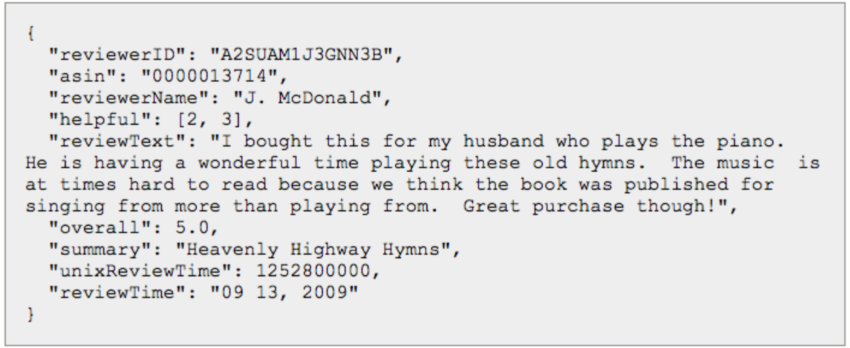
Top-left: wow
Top-middle: ok
Top-right: what happened
Bottom-left: great album
Bottom-middle: fans can never be objective
Bottom-right: garbage

Some remarks:

- note the trends are quite different from the general trend
- note that we cannot always be very confident about the overall score

Example - Very high-dimensional predictors

Consider a predictor like `reviewText`



```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

We could not use it in the same way we used the summary. Given a certain reviewText, we typically only get 1 data point making the problem look deterministic. This is a fallacy though, it only *looks* deterministic because of a modelling choice.

If you are not convinced that the determinism is artificial, consider the following thought experiment. Ask everyone in the classroom to assign stars to a product based on a given review, what might you observe?

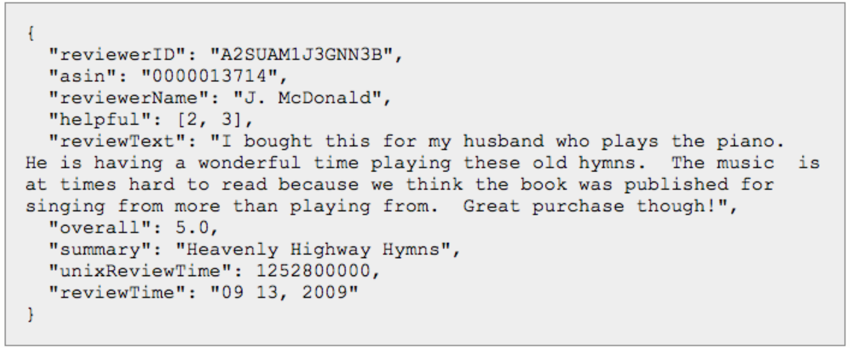
We cannot expect determinism, neither we should really look for it. Variance does not mean we are doing something wrong, the data are like that indeed: we lack knowledge about all factors involved in the data generating process.

Consider a person performing some annotation: variability can be result of lapse in attention, long working hours, or simply inherent to the task.

Whereas determining logical entailment seems rather trivial (well, it still depends on the excerpts of text we are given), translating is a much more creative process.

Machine learning and pattern recognition

Here is where things get interesting, even if a bit less well defined



```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

We want to condition on rich predictors but we want to learn to identify and exploit patterns that *generalise* a certain hidden aspect of the data (e.g., how people relate products, their views, and a certain score).

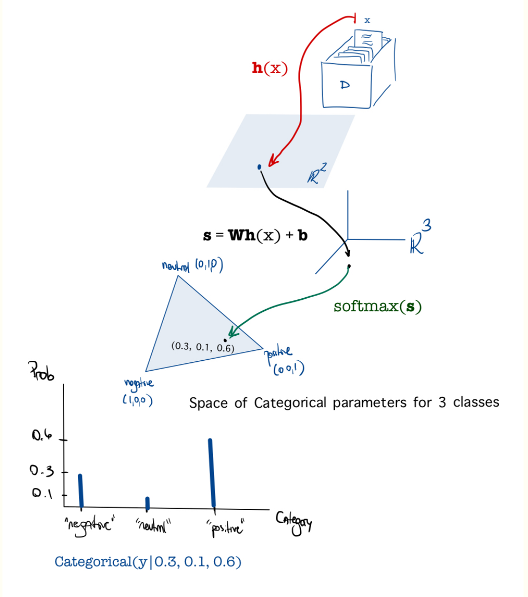
Why do I say this is a bit less well defined?

We are embracing the idea of learning patterns that are specific enough to explain away most variance in the data, yet general enough to be reusable in the future.

We are alluding to some notion of *generalisation* without a clear idea of what it is or how to operationalise it.

Example - Learning to use very rich predictors

Before DL was popular, we dealt with this mostly by identifying informative features $h(x)$ of the available predictor x. We would then map these features to the parameter of a Categorical distribution (e.g., via a log-linear model) on demand: $Y|X = x \sim \text{Cat}(\text{softmax}(Wh(x) + b))$.



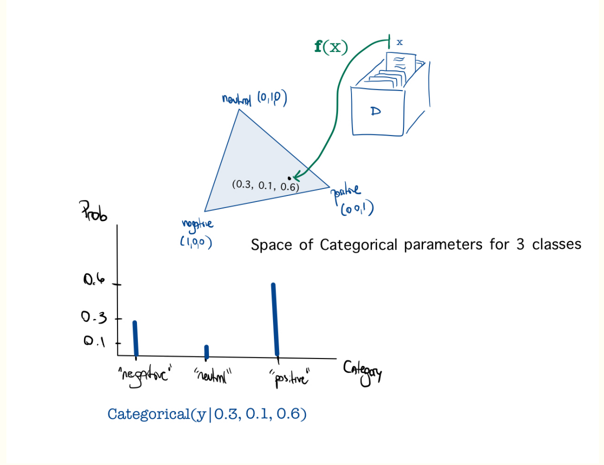
Example - Learning to use very rich predictors

Before DL was popular, we dealt with this mostly by identifying informative features $h(x)$ of the available predictor x. We would then map these features to the parameter of a Categorical distribution (e.g., via a log-linear model) on demand: $Y|X = x \sim \text{Cat}(\text{softmax}(Wh(x) + b))$.

Nowadays, we tend to condition on **everything available to us** by learning how to map from **arbitrarily complex** data to the parameters of our distributions. We do so with NNs: $Y|X = x \sim \text{Cat}(f(x;\theta))$.

There's a lot of research on how to design $f(\cdot;\theta)$ and estimate θ effectively. DL1 and DL2 cover those aspects too!

In $Y|X = x \sim \text{Cat}(f(x;\theta))$, $f(\cdot;\theta)$ is a NN architecture with parameters θ, it maps any covariate x, say a long review in English, to the parameters of the Categorical distribution that *by assumption* govern the conditional response variable.



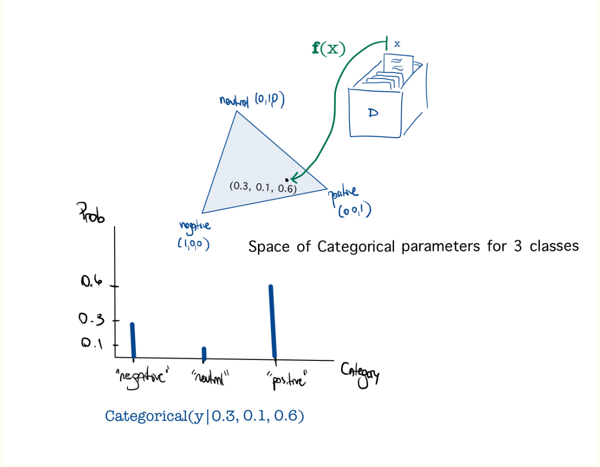
Example - Learning to use very rich predictors

Before DL was popular, we dealt with this mostly by identifying informative features $h(x)$ of the available predictor x. We would then map these features to the parameter of a Categorical distribution (e.g., via a log-linear model) on demand: $Y|X = x \sim \text{Cat}(\text{softmax}(Wh(x) + b))$.

Nowadays, we tend to condition on **everything available to us** by learning how to map from **arbitrarily complex** data to the parameters of our distributions. We do so with NNs: $Y|X = x \sim \text{Cat}(f(x;\theta))$.

There's a lot of research on how to design $f(\cdot;\theta)$ and estimate θ effectively. DL1 and DL2 cover those aspects too!

In $Y|X = x \sim \text{Cat}(f(x;\theta))$, $f(\cdot;\theta)$ is a NN architecture with parameters θ, it maps any covariate x, say a long review in English, to the parameters of the Categorical distribution that *by assumption* govern the conditional response variable.



Shallow statistical models

We have data $y^{(1)}, \dots, y^{(N)}$ generated by some **unknown** procedure which we assume can be captured by a probabilistic model

- with **known** probability (mass/density) function e.g.

$$Y \sim \mathrm{Cat}(\phi_1, \dots, \phi_K) \qquad \text{or} \qquad Y \sim \mathcal{N}(\mu, \sigma^2)$$

There are a few distributions for which we know the exact MLE solution, for all others we can try our chances with gradient-based optimisation.

Shallow statistical models

We have data $y^{(1)}, \dots, y^{(N)}$ generated by some **unknown** procedure which we assume can be captured by a probabilistic model

- with **known** probability (mass/density) function e.g.

$$Y \sim \mathrm{Cat}(\phi_1, \dots, \phi_K) \qquad \text{or} \qquad Y \sim \mathcal{N}(\mu, \sigma^2)$$

and estimate parameters of the pdf to attain maximum likelihood given observations

There are a few distributions for which we know the exact MLE solution, for all others we can try our chances with gradient-based optimisation.

Parameterisation by NNs

Let x be all side information available
e.g. *inputs/features/predictors/covariates*

Have neural networks predict parameters of our probabilistic model

$$Y|x \sim \text{Cat}(f(x;\theta)) \qquad \text{or} \qquad Y|x \sim \mathcal{N}(\mu(x;\theta), \sigma(x;\theta)^2)$$

and proceed to estimate parameters θ of the NNs

NNs compute the parameters of the statistical model. We estimate NN parameters.

Now that we have NNs in the parameterisation, we will never be able to derive a closed-form solution for the maximum likelihood estimate of the NN parameters.

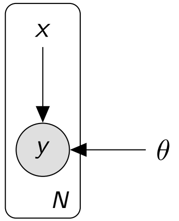
Graphical model

Random variables

- observed data
  $y^{(1)}, \dots, y^{(N)}$

Deterministic variables

- predictors $x^{(1)}, \dots, x^{(N)}$
  non-random observed variable
- model parameters θ
  non-random and unobservable variable



For now we are taking the Frequentist point of view where parameters are assumed known. Clearly we do not just happen to *know* the parameters of a statistical model, though we may be able to make a somewhat informed choice based of the available (training) data.

In Frequentism parameters are *determined* via optimisation of a likelihood-based criterion. This is known as parameter estimation.

When we discuss Bayesian principles we will see that alternatively we may acknowledge that parameters are random variables and that we don't know much about them (besides what can be coded in a choice of governing distribution known as prior). Then we dispense with parameter estimation altogether, rather using probabilistic inference to reason about quantities of interest such as probability queries about unobserved random variables given observed data. This is called posterior inference.

Task-driven feature extraction

Often our side information is itself some high dimensional object

- x is a sentence and y a tree
- x is the source sentence and y is the target
- x is an image and y is a caption

and part of the job of the NNs that parametrise our models is to also deterministically encode that input in a low-dimensional space

In representation learning, these encodings are the subject of interest, much more than the model itself.

Example: word embedding models learn to predict probability distributions over neighbouring words, but ultimately one only cares about the representations of those words, which is internal to the parameterisation of the distribution.

In fact, in representation learning we find many instances of deep learning models that are not probabilistic.

NN as efficient parametrisation

From a statistical point of view, NNs do not generate data

- they parametrise distributions that *by assumption* generated our data
- compact and efficient way to map from complex side information to parameter space

From a statistical point of view, it is inconsequential whether you can do anything useful with intermediate representations inside of an NN.

Example: word embeddings by word2vec are *accidental* from a statistical point of view. The embeddings are not a unobserved random variable we modelled, they are part of the specification of some other distribution. I use the word 'accidental' just to get your attention. Clearly, the designer engineered a specific task, and a specific classifier to get word2vec embeddings to be useful in the way they are. The designer exploited inductive biases aimed at making embeddings function as if they captured lexical semantics.

What I hope is that you will see that there are multiple, complementary, ways to code inductive biases. Manipulating statistical properties of the model is another one.

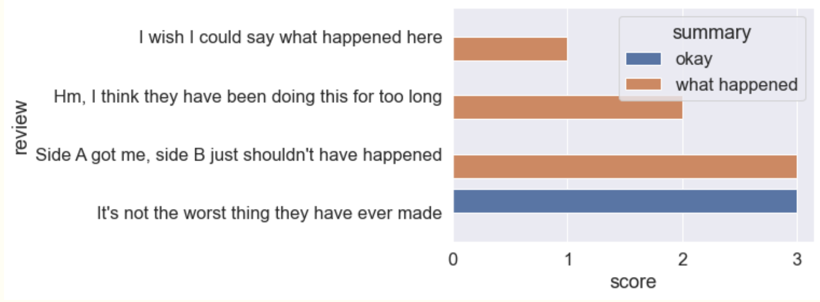
It looks different, but is it?

The ability to predict a distribution per input efficiently makes it look like everything changed, but did it?

We are asking our models the predict the observed pattern, but data sparsity makes the response variable appear deterministic given the input.

We delegate to the NN the job of figuring out regularities in input space, the degree to which we can influence what regularities will be captured (if any) is somewhat limited to bottlenecks we plant in their design (i.e., architecture design).

MLE has not built-in pressure for generalisation.

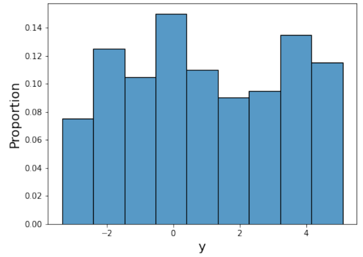


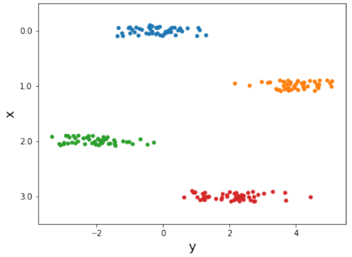
Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails

Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails

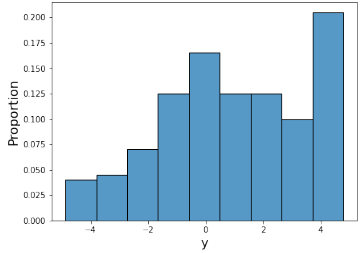


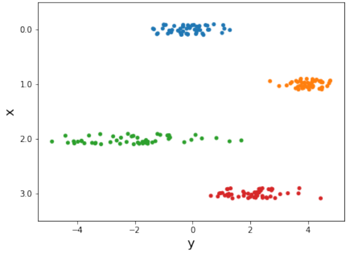


- Don't be mislead by the marginal if you intend to model conditionally

Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails

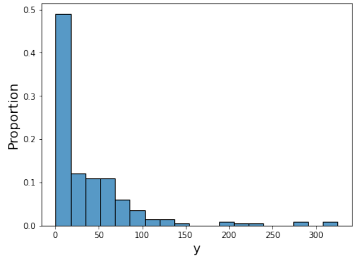


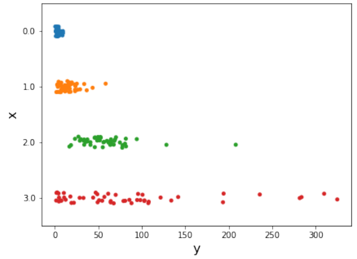


- Don’t be mislead by the marginal if you intend to model conditionally
- Do you expect variance to differ?

Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails

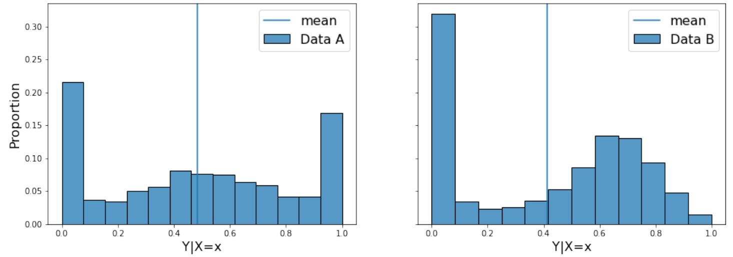




- Don't be mislead by the marginal if you intend to model conditionally
- Do you expect variance to differ?
- Do you expect skew?

Choosing a model family

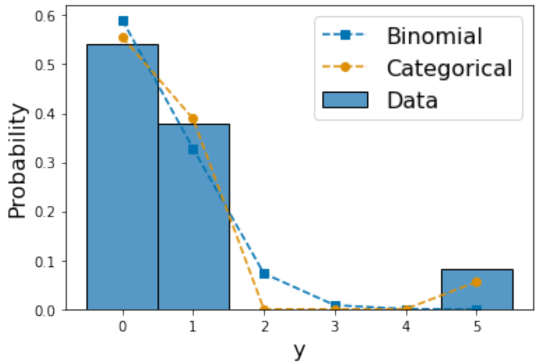
1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails



- Don't be mislead by the marginal if you intend to model conditionally
- Do you expect variance to differ?
- Do you expect skew?
- Bounded support? Multimodality? Asymmetry?

Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails



- Don't be mislead by the marginal if you intend to model conditionally
- Do you expect variance to differ?
- Do you expect skew?
- Bounded support? Multimodality? Asymmetry?
- Unsure about categorical or ordinal treatment?

Choosing a model family

1. data type: countable (binary, categorical, ordinal), uncountable, univariate or multivariate, combinatorial, etc.
2. match properties of the data and distribution: overdispersion, skewness, heavy tails

- Don't be mislead by the marginal if you intend to model conditionally
- Do you expect variance to differ?
- Do you expect skew?
- Bounded support? Multimodality? Asymmetry?
- Unsure about categorical or ordinal treatment?

Outline

Maximum likelihood estimation

We have a probability model of a random variable Y, and this model may condition on available covariates X. This model has parameters θ and assigns probability $p(y|x, \theta)$ to an observation.

I may omit the subscripts from the pdfs whenever I find it unambiguous. That is, I write $p(y|x, \theta)$ instead of $p_{Y|X}(y|x, \theta)$.

Maximum likelihood estimation

We have a probability model of a random variable Y, and this model may condition on available covariates X. This model has parameters θ and assigns probability $p(y|x, \theta)$ to an observation.

Given a dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of i.i.d. observations,

I may omit the subscripts from the pdfs whenever I find it unambiguous. That is, I write $p(y|x, \theta)$ instead of $p_{Y|X}(y|x, \theta)$.

Maximum likelihood estimation

We have a probability model of a random variable Y, and this model may condition on available covariates X. This model has parameters θ and assigns probability $p(y|x, \theta)$ to an observation.

Given a dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of i.i.d. observations, the log-likelihood function gives us a criterion for parameter estimation

$$\mathcal{L}_{\mathcal{D}}(\theta) =$$

I may omit the subscripts from the pdfs whenever I find it unambiguous. That is, I write $p(y|x, \theta)$ instead of $p_{Y|X}(y|x, \theta)$.

Maximum likelihood estimation

We have a probability model of a random variable Y, and this model may condition on available covariates X. This model has parameters θ and assigns probability $p(y|x, \theta)$ to an observation.

Given a dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of i.i.d. observations, the log-likelihood function gives us a criterion for parameter estimation

$$\mathcal{L}_{\mathcal{D}}(\theta) = \log \prod_{s=1}^{N} p(y^{(s)}|x^{(s)}, \theta) =$$

I may omit the subscripts from the pdfs whenever I find it unambiguous. That is, I write $p(y|x, \theta)$ instead of $p_{Y|X}(y|x, \theta)$.

Maximum likelihood estimation

We have a probability model of a random variable Y, and this model may condition on available covariates X. This model has parameters θ and assigns probability $p(y|x,\theta)$ to an observation.

Given a dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of i.i.d. observations, the log-likelihood function gives us a criterion for parameter estimation

$$\mathcal{L}_{\mathcal{D}}(\theta) = \log \prod_{s=1}^{N} p(y^{(s)}|x^{(s)},\theta) = \sum_{s=1}^{N} \log p(y^{(s)}|x^{(s)},\theta)$$

I may omit the subscripts from the pdfs whenever I find it unambiguous. That is, I write $p(y|x,\theta)$ instead of $p_{Y|X}(y|x,\theta)$.

MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation gives us the gradient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) =$$

Differentiable

Consider the example of a Categorical likelihood:

- for a data point (x, y) the log-likelihood is $\log \mathrm{Cat}(y|f(x;\theta)) = \log f_y(x;\theta)$ This shows that the Categorical likelihood $\mathrm{Cat}(y|f(x;\theta))$ is differentiable with respect to its parameter $f_y(x;\theta)$.
- To satisfy differentiability with respect to θ for any (x, y), we need $f(\cdot;\theta)$, to be differentiable with respect to θ in its domain (the space $\mathcal{X}$ of all covariates).

Tractable The evaluation of $f(x;\theta)$ is tractable for any $x \in \mathcal{X}$.

Beyond Think about other likelihoods (e.g., Bernoulli, Binomial, Multinomial, Poisson, Geometric, Gaussian, Exponential, Gamma), can you imagine differentiable and tractable parameterisations of the model?

MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation gives us the gradient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \nabla_\theta \sum_{s=1}^{N} \log p(y^{(s)}|x^{(s)}, \theta) \quad =$$

Differentiable

Consider the example of a Categorical likelihood:

- for a data point (x, y) the log-likelihood is $\log \mathrm{Cat}(y|f(x;\theta)) = \log f_y(x;\theta)$
  This shows that the Categorical likelihood $\mathrm{Cat}(y|f(x;\theta))$ is differentiable with respect to its parameter $f_y(x;\theta)$.
- To satisfy differentiability with respect to θ for any (x, y), we need $f(\cdot;\theta)$, to be differentiable with respect to θ in its domain (the space $\mathcal{X}$ of all covariates).

Tractable The evaluation of $f(x;\theta)$ is tractable for any $x \in \mathcal{X}$.

Beyond Think about other likelihoods (e.g., Bernoulli, Binomial, Multinomial, Poisson, Geometric, Gaussian, Exponential, Gamma), can you imagine differentiable and tractable parameterisations of the model?

MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation gives us the gradient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \nabla_\theta \sum_{s=1}^{N} \log p(y^{(s)}|x^{(s)}, \theta) \quad = \sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)$$

Differentiable

Consider the example of a Categorical likelihood:

- for a data point (x, y) the log-likelihood is $\log \text{Cat}(y|f(x;\theta)) = \log f_y(x;\theta)$
  This shows that the Categorical likelihood $\text{Cat}(y|f(x;\theta))$ is differentiable with respect to its parameter $f_y(x;\theta)$.
- To satisfy differentiability with respect to θ for any (x, y), we need $f(\cdot;\theta)$, to be differentiable with respect to θ in its domain (the space $\mathcal{X}$ of all covariates).

Tractable The evaluation of $f(x;\theta)$ is tractable for any $x \in \mathcal{X}$.

Beyond Think about other likelihoods (e.g., Bernoulli, Binomial, Multinomial, Poisson, Geometric, Gaussian, Exponential, Gamma), can you imagine differentiable and tractable parameterisations of the model?

MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation gives us the gradient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \nabla_\theta \sum_{s=1}^{N} \log p(y^{(s)}|x^{(s)}, \theta) \quad = \sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)$$

and we can update θ in the direction

$$\gamma \nabla_\theta \mathcal{L}_\mathcal{D}(\theta)$$

to attain a local maximum of the likelihood function

Differentiable

Consider the example of a Categorical likelihood:

- for a data point (x, y) the log-likelihood is $\log \text{Cat}(y|f(x;\theta)) = \log f_y(x;\theta)$
  This shows that the Categorical likelihood $\text{Cat}(y|f(x;\theta))$ is differentiable with respect to its parameter $f_y(x;\theta)$.
- To satisfy differentiability with respect to θ for any (x, y), we need $f(\cdot;\theta)$, to be differentiable with respect to θ in its domain (the space $\mathcal{X}$ of all covariates).

Tractable The evaluation of $f(x;\theta)$ is tractable for any $x \in \mathcal{X}$.

Beyond Think about other likelihoods (e.g., Bernoulli, Binomial, Multinomial, Poisson, Geometric, Gaussian, Exponential, Gamma), can you imagine differentiable and tractable parameterisations of the model?

Big Data

For large N, computing the gradient is inconvenient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)}_{\text{too many terms}}$$

We are looking for a principled way to approximate the exact gradient. Being principled here means enjoying some guarantees (this usually requires satisfying certain properties, as we shall see).

Note that we introduced the notion of a *stochastic gradient*, a random variable whose range is the space of gradient vectors of our model's log-likelihood function.

We have expressed the exact gradient as the expected value of that random variable. Can you see how we are going to estimate it with a computation that does not depend on N?

Big Data

For large N, computing the gradient is inconvenient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)}_{\text{too many terms}}$$

$$= \sum_{s=1}^{N} \frac{1}{N} N \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)$$

We are looking for a principled way to approximate the exact gradient. Being principled here means enjoying some guarantees (this usually requires satisfying certain properties, as we shall see).

Note that we introduced the notion of a *stochastic gradient*, a random variable whose range is the space of gradient vectors of our model's log-likelihood function.

We have expressed the exact gradient as the expected value of that random variable. Can you see how we are going to estimate it with a computation that does not depend on N?

Big Data

For large N, computing the gradient is inconvenient

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)}_{\text{too many terms}}$$

$$= \sum_{s=1}^{N} \frac{1}{N} N \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)$$

$$= \sum_{s=1}^{N} \mathcal{U}(s|1/N) N \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)$$

We are looking for a principled way to approximate the exact gradient. Being principled here means enjoying some guarantees (this usually requires satisfying certain properties, as we shall see).

Note that we introduced the notion of a *stochastic gradient*, a random variable whose range is the space of gradient vectors of our model's log-likelihood function.

We have expressed the exact gradient as the expected value of that random variable. Can you see how we are going to estimate it with a computation that does not depend on N?

Big Data

For large N, computing the gradient is inconvenient

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta) &= \underbrace{\sum_{s=1}^{N} \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta)}_{\text{too many terms}} \\
&= \sum_{s=1}^{N} \frac{1}{N} N \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta) \\
&= \sum_{s=1}^{N} \mathcal{U}(s|{}^1/_N) N \nabla_\theta \log p(y^{(s)}|x^{(s)}, \theta) \\
&= \mathbb{E}_{S\sim\mathcal{U}({}^1/_N)}\left[N \nabla_\theta \log p(y^{(S)}|x^{(S)}, \theta)\right]
\end{aligned}$$

S selects data points uniformly at random

We are looking for a principled way to approximate the exact gradient. Being principled here means enjoying some guarantees (this usually requires satisfying certain properties, as we shall see).

Note that we introduced the notion of a *stochastic gradient*, a random variable whose range is the space of gradient vectors of our model's log-likelihood function.

We have expressed the exact gradient as the expected value of that random variable. Can you see how we are going to estimate it with a computation that does not depend on N?

Stochastic optimisation

For large N, we can use a gradient estimate

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\mathbb{E}_{S\sim\mathcal{U}(1/N)}\left[N\nabla_\theta \log p(y^{(S)}|x^{(S)}, \theta)\right]}_{\text{expected gradient :)}}$$

The theory of stochastic optimisation (Robbins and Monro, 1951) tells us that we will converge to a local optimum of the objective as long as we take steps that are correct *on average*. This means we can optimisation with stochastic gradient estimates, for as long as they are unbiased estimates of the exact gradient.

Do you see the guarantee and the condition?

There are more conditions, however. The learning rate must comply with some key properties. Luckily many learning rate schedules have been documented in the literature, and most our famous optimisers meet the Robbis and Monro conditions (though not all).

If you want to read more, but need something more accessible than the 1951 paper, check (Bottou, 2010).

Stochastic optimisation

For large N, we can use a gradient estimate

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\mathbb{E}_{S\sim\mathcal{U}(1/N)}\left[N\nabla_\theta \log p(y^{(S)}|x^{(S)},\theta)\right]}_{\text{expected gradient :)}}$$

$$\overset{\text{MC}}{\approx} \frac{1}{M}\sum_{m=1}^{M} N\nabla_\theta \log p(y^{(s_m)}|x^{(s_m)},\theta)$$

$$S_m \sim \mathcal{U}(1/N)$$

The theory of stochastic optimisation (Robbins and Monro, 1951) tells us that we will converge to a local optimum of the objective as long as we take steps that are correct *on average*. This means we can optimisation with stochastic gradient estimates, for as long as they are unbiased estimates of the exact gradient.

Do you see the guarantee and the condition?

There are more conditions, however. The learning rate must comply with some key properties. Luckily many learning rate schedules have been documented in the literature, and most our famous optimisers meet the Robbis and Monro conditions (though not all).

If you want to read more, but need something more accessible than the 1951 paper, check (Bottou, 2010).

Stochastic optimisation

For large N, we can use a gradient estimate

$$\nabla_\theta \mathcal{L}_\mathcal{D}(\theta) = \underbrace{\mathbb{E}_{S\sim\mathcal{U}(1/N)}\left[N\nabla_\theta \log p(y^{(S)}|x^{(S)},\theta)\right]}_{\text{expected gradient :)}}$$

$$\overset{\text{MC}}{\approx} \frac{1}{M}\sum_{m=1}^{M} N\nabla_\theta \log p(y^{(s_m)}|x^{(s_m)},\theta)$$

$$S_m \sim \mathcal{U}(1/N)$$

and take a step in the direction

$$\gamma\frac{N}{M}\underbrace{\nabla_\theta \mathcal{L}_\mathcal{B}(\theta)}_{\text{stochastic gradient}}$$

where $\mathcal{B} = \{(x^{(s_1)}, y^{(s_1)}), \ldots, (x^{(s_M)}, y^{(s_M)})\}$ is a random mini-batch

The theory of stochastic optimisation (Robbins and Monro, 1951) tells us that we will converge to a local optimum of the objective as long as we take steps that are correct *on average*. This means we can optimisation with stochastic gradient estimates, for as long as they are unbiased estimates of the exact gradient.

Do you see the guarantee and the condition?

There are more conditions, however. The learning rate must comply with some key properties. Luckily many learning rate schedules have been documented in the literature, and most our famous optimisers meet the Robbis and Monro conditions (though not all).

If you want to read more, but need something more accessible than the 1951 paper, check (Bottou, 2010).

DL in NLP recipe

Maximum likelihood estimation

- tells you which loss to optimise
  (i.e. negative log-likelihood)

Automatic differentiation (*backprop*)

- "give me a tractable forward pass and I will give you gradients"

Stochastic optimisation powered by backprop

- general purpose gradient-based optimisers

How about binary cross entropy? Or Categorical cross-entropy? Or MSE?

Those are all (very) closely-related to the negative log-likelihood of a probability model under a certain choice of output distribution. There is no need to memorise any such cross entropy, you can derive the correct expression from basic principles. It's also easier to talk about and think of it in terms of log-likelihood, as a 'cross-entropy' requires a non-trivial understanding of the data in terms of *observed distributions* (rather than observed outcomes).

Constraints

Differentiability

- intermediate representations must be continuous
- activations must be differentiable

Tractability

- the likelihood function must be evaluated exactly, thus it's required to be tractable

Outline

How about tasks?

Some models support decision makers in performing predictions about future data. These ‘tasks’ or downstream applications we find for our models guide our design assumptions, but they normally have little impact on the statistics of parameter estimation.

We train our model to reproduce statistical patterns of the data, and then use them to make **decisions**.

Some exceptions

- Loss-calibrated probabilistic inference
- Reinforcement learning

How would you decide in these cases?

You can pick one outcome, what will you pick?

What task does MLE provide a decision rule for?

What task does MLE provide a decision rule for?

The task of assigning probability mass/density to a joint outcome of the random variables of interest!

What task does MLE provide a decision rule for?

The task of assigning probability mass/density to a joint outcome of the random variables of interest!

If the task you care about is not to assign probability mass/density, but rather to return an outcome, you will have to come up with a decision rule (i.e., an algorithm) for that.

Decision rules

A **decision rule** is an algorithm for making decisions. If you have ever designed a C-way classifier, you are familiar with the most probable class rule.

MLE does not really imply a specific algorithm for decision making. So, we chose our decision rules mostly axiomatically.

Statistical decision theory gives us a useful axiom when deciding under a conditional distribution $Y|X = x$:

$$\begin{aligned} y^\star &= \arg\max_{c \in \mathcal{Y}} \ \mathbb{E}[u(Y, c)|X = x] \\ &= \arg\max_{c \in \mathcal{Y}} \ \int_{\mathcal{Y}} u(y, c) p_{Y|X}(y|x) \end{aligned}$$

A utility function $u(y, c)$ assesses the benefit of choosing candidate c when y is the correct response.

We do not know what responses are correct, so we compute the utility of a candidate c against the entire conditional distribution, that is, we compute the utility of c in expectation.

This is how we generalise the 'most probable class' popular when deciding under a Categorical distribution to other decision problems (e.g., deciding under a Normal, or under a mixture of Poissons, or under a mixture of Gammas, or under a simulator, or under a sequence model, or under a graph model, etc.)

The decision rule that returns the mode of the conditional distribution is obtained by using $u(y, c) = [y = c]$.

Approximations

Depending on the utility function and on the factorisation of the model, we may face challenges:

- expected utility may be intractable to compute (e.g., mixture models, autoregressive generators)
- exact search may be intractable (e.g., combinatorial sample space)

For sequence generators, a greedy approximation (beam search) to the most probable sequence is very popular. For alternatives see (Eikema and Aziz, 2021).

In general, we can always estimate expected utility via Monte Carlo. Exact search is much more difficult to approximate (but see BayesOpt (Snoek et al., 2012)).

Next class

Deep latent variable models with discrete latent variables

- Exact inference
- Approximate inference

Outline

Probability versus simulation: what came first?

Sometimes a model has a mechanism to generate random draws and this mechanism could be used–in principle–to compute probability values, but the computation is intractable. These are called **implicit** models or *simulators*.

The alternative to an implicit model is a **prescribed** model. In this case the model has an explicit mechanism to assess the probability value of a given outcome. This means that–in principle–the model can also be used to generate random draws, but sometimes this requires intractable computations.

An example of the former is a GAN, an example of the latter is a Boltzmann machine.

Inputs and outputs

Watch out! What is to be considered an input or an output depends on who is doing the processing.

Point of view	Input	Output
Statistician	domain knowledge data	model specification
Parameter estimation	model specification observed data	parameters
Statistical model	data	probability distribution
Decision maker	model specification parameters decision rule novel data	decision

The statistician designs a model

Data can be used to estimate free parameters

The model predicts a distribution

Decisions are made based on the output of the model (a distribution).

Besides, NNs have inputs, they are encodings of variables that undergo transformation (sometimes these inputs were not available from some dataset of observations, they are outputs from the model via some decision rule).

NNs have outputs, they can be thought of as alternative views of data (encodings) or statistical parameters of distributions. But they are not model outputs, nor outputs from the point of view of a specific application (end user).

Latent or Hidden?

For us, and for enough of the community out there, a **latent** variable is an **unobserved random variable**. The two terms are equivalent.

The word **hidden** is more overloaded and we will simply avoid it (except perhaps when talking about NN architecture design)

- The *hidden state* of the classic Hidden Markov Model is a latent variable
- A *hidden unit* in an NN is seldom a latent variable. In fact most hidden units are not at all unknown: it's just that it takes a forward pass through the network for one to be able to 'see' (or get to know) them.

In doubt, when talking about a latent variable you can simply emphasise stochasticity by saying 'unobserved random variable' or 'latent random variable' (to some the latter will sound repetitive, but not enough to sound strange).

Supervised, unsupervised, semi-, self-, . . .

Oh this is a difficult one! Just too many views, many reasonably logical.

I like to think of it as learning in the presence or absence of latent variables. If we have unobserved random variables that stay unobserved throughout, that's unsupervised learning. If a subset of my unobserved variables become available as observations, that's semi-supervise. Otherwise it's supervised.

But remember word2vec? Its inventor wanted to learn word embeddings (sounds unsupervised, right?), but these are parameters of a binary classifier involving only observed random variables. How about 'self-supervised'?

I could confuse you for the rest of your lives if I were to get into all proposals.

My advice: stay away from the debate, just be clear about what you do. And if you can, stay coherent without being too confrontational.

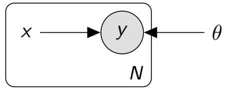
If I were to justify my point of view I would say it is model-centred (as opposed to application-centred): that means I do not take the intended use of the model into account in order to assign one such label. I only consider the presence or absence of unobserved random variables.

In my view then something like word2vec, or anything by today's standard 'self-supervised', is supervised learning. That is, learning in the complete absence of latent variables.

You can think of these terms from the point of view of the nature of the random variables (observed, unobserved), or from the point of view of the purpose of the model, or from the point of view of the type of distribution, each view leads to a different way to assign the labels. And there are more, some will say that transfer learning techniques lead to semi-supervised models.

On self-supervised learning: note that it sounds like the learner (say the model) found its own supervision, but really we (modellers) were the ones to find a task for which a cheap-to-obtain observation leads to some representation of the data that's useful in other situations.

Multiple problems, same language



(Conditional) Density estimation

	Predictor (x)	Outcome (y)
Parsing	a sentence	its syntactic/semantic parse tree/graph
Translation	a sentence	its translation
Captioning	an image	caption in English
Entailment	a text and hypothesis	entailment relation

References I

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Bryan Eikema and Wilker Aziz. Sampling-based minimum bayes risk decoding for neural machine translation. *arXiv preprint arXiv:2108.04718*, 2021.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *ICLR - workshop track*, 2016a.

References II

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, June 2016b. PMLR. URL `http://proceedings.mlr.press/v48/gal16.html`.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in neural information processing systems 29*, pages 1019–1027. Curran Associates, Inc., 2016c. URL `http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-rec` `pdf`.

References III

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. Publisher: JSTOR.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

References IV

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 (56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.