Bayesian framework

Dmitry Vetrov
Research professor at HSE
Senior researcher at Yandex
Head of Bayesian methods research group
http://bayesgroup.ru

Deep|Bayes

Outline

- Bayes theorem
- Frequentist vs. Bayesian
- Generative and discriminative models
- Learning Bayesian models
- Advantages of Bayesian ML models
- KL-divergence between the distributions

Conditional and marginal distributions

Just to remind...

- Conditional distribution

$$\texttt{Conditional} = \frac{\texttt{Joint}}{\texttt{Marginal}}, \quad p(x|y) = \frac{p(x,y)}{p(y)}$$

- Product rule: Any joint distribution can be expressed as a product of one-dimensional conditional distributions

$$p(x,y,z) = p(x|y,z)p(y|z)p(z) = p(z|x,y)p(x|y)p(y)$$

- Sum rule: Any marginal distribution can be obtained from the joint distribution by **intergrating out** unnessesary variables

$$p(y) = \int p(x,y)dx = \int p(y|x)p(x)dx = \mathbb{E}_x p(y|x)$$

Arbitrary conditioning

- Assume we have a joint distribution over three groups of variables $p(X, Y, Z)$
- We observe Z and are interested in predicting X
- Values of Y are unknown and irrelevant for us
- How to estimate $p(X|Z)$ from $p(X, Y, Z)$?

Arbitrary conditioning

- Assume we have a joint distribution over three groups of variables $p(X, Y, Z)$
- We observe Z and are interested in predicting X
- Values of Y are unknown and irrelevant for us
- How to estimate $p(X|Z)$ from $p(X, Y, Z)$?

$$p(X|Z) = \frac{p(X, Z)}{p(Z)} = \frac{\int p(X, Y, Z) dY}{\int p(X, Y, Z) dY \, dX}$$

- Sum rule allows to build arbitrary conditional distributions at least in theory

Bayes theorem

- Conditionals inversion (follows from product rule):

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(y|x)p(x)}{p(y)}$$

- Bayes theorem (follows from conditionals inversion and sum rule):

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x)p(x)dx}$$

- Bayes theorem defines the rule for uncertainty conversion when new information arrives

$$\texttt{Posterior} = \frac{\texttt{Likelihood} \times \texttt{Prior}}{\texttt{Evidence}}$$

Statistical inference

- Consider standard problem of statistical inference. Given i.i.d. data $X = (x_1, \ldots, x_n)$ from distribution $p(x|\theta)$ one needs to estimate θ
- Maximum likelihood estimation (MLE):

$$\theta_{ML} = \arg\max p(X|\theta) = \arg\max \prod_{i=1}^{n} p(x_i|\theta) = \arg\max \sum_{i=1}^{n} \log p(x_i|\theta)$$

- Bayesian inference: encode uncertainty about θ in terms of a distribution $p(\theta)$ and apply Bayesian inference

$$p(\theta|X) = \frac{\prod_{i=1}^{n} p(x_i|\theta)p(\theta)}{\int \prod_{i=1}^{n} p(x_i|\theta)p(\theta)d\theta}$$

Frequentist vs. Bayesian frameworks

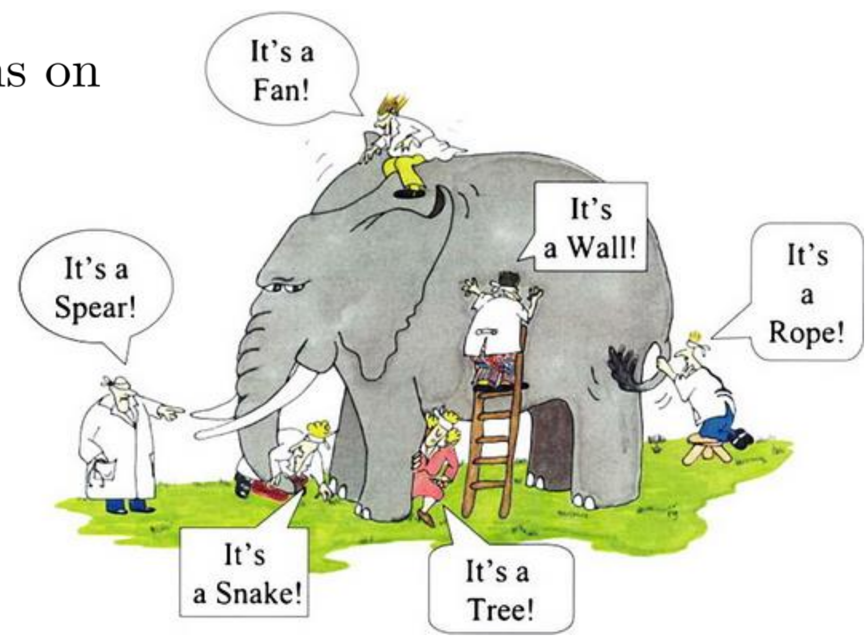
	Frequentist	Bayesian
Randomness	Objective indefiniteness	Subjective ignorance
Variables	Random and Deterministic	Everything is random
Inference	Maximal likelihood	Bayes theorem
Estimates	ML-estimates	Posterior or MAP-estimates
Applicability	$n \gg 1$	$\forall n$

Bayesian framework

- Encodes ignorance in terms of distributions
- Makes use of **Bayes Theorem**

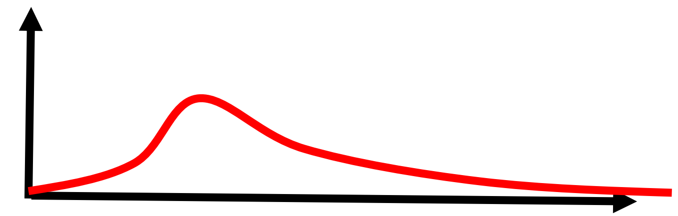
$$\texttt{Posterior} = \frac{\texttt{Likelihood} \times \texttt{Prior}}{\texttt{Evidence}}, \quad p(\theta|X) = \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta}$$

- Posteriors may serve as new priors, i.e. may combine multiple models!
- **BigData:** we can process data streams on an update-and-forget basis
- Support distributed processing



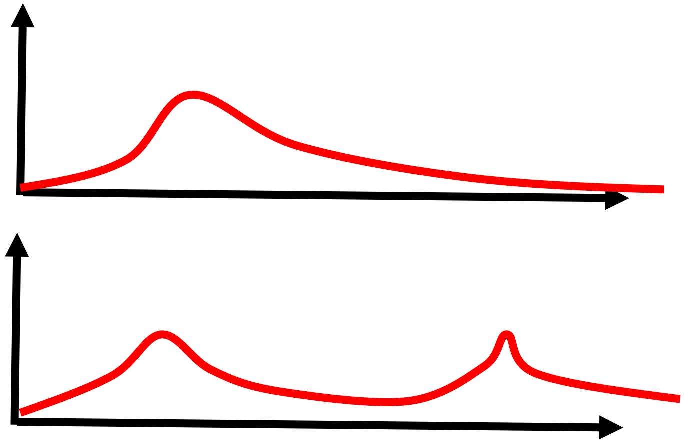
Bayesian inference

- Consider blind wisdomers who try to estimate the mass of an elephant using their tactile measurements.
- They start with common knowledge about animals typical masses $p(\theta)$



Bayesian inference

- Consider blind wisdomers who try to estimate the mass of an elephant using their tactile measurements.
- They start with common knowledge about animals typical masses $p(\theta)$
- The first wisdomer touches a tail



$$p(\theta|x_1) = \frac{p_1(x_1|\theta)p(\theta)}{\int p_1(x_1|\theta)p(\theta)d\theta}$$

Bayesian inference

- Consider blind wisdomers who try to estimate the mass of an elephant using their tactile measurements.

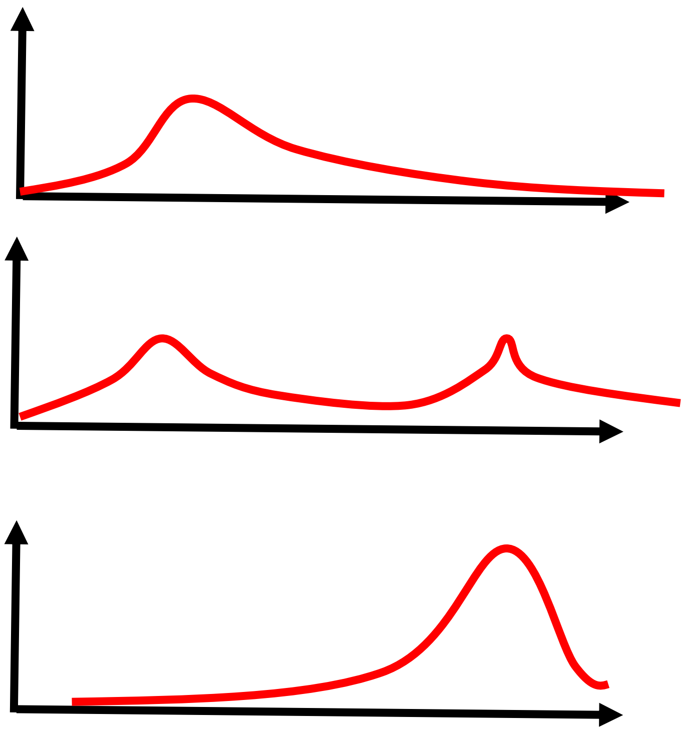
- They start with common knowledge about animals typical masses $p(\theta)$

- The first wisdomer touches a tail

$$p(\theta|x_1) = \frac{p_1(x_1|\theta)p(\theta)}{\int p_1(x_1|\theta)p(\theta)d\theta}$$

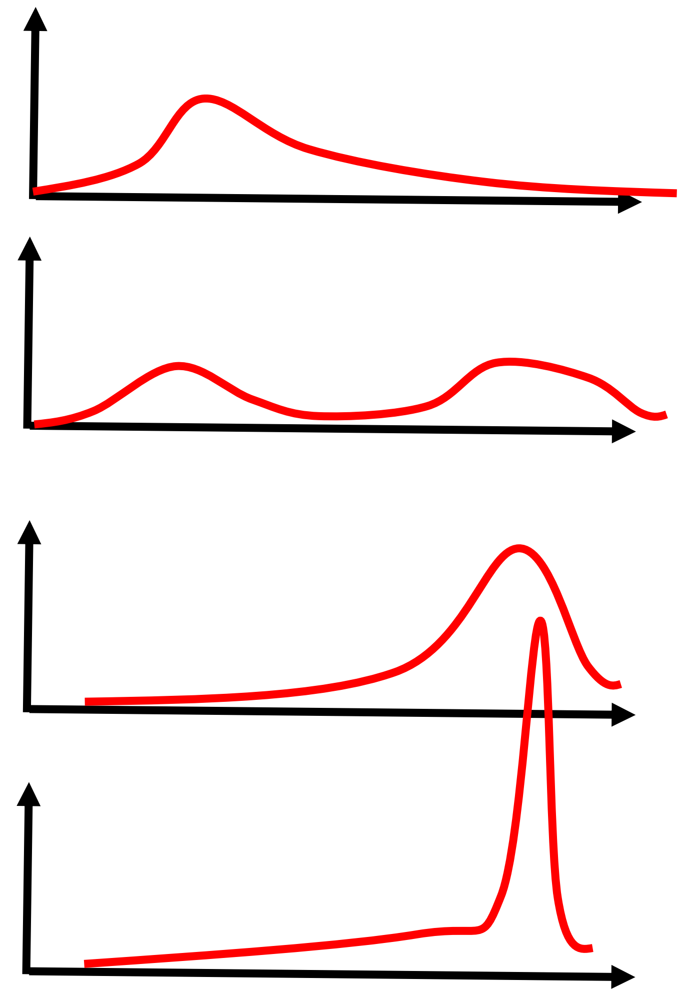
- The second wisdomer touches a leg and uses $p(\theta|x_1)$ as **his new prior**

$$p(\theta|x_1, x_2) = \frac{p_2(x_2|\theta)p(\theta|x_1)}{\int p_2(x_2|\theta)p(\theta|x_1)d\theta}$$



Bayesian inference

- Consider blind wisdomers who try to estimate the mass of an elephant using their tactile measurements.



- They start with common knowledge about animals typical masses $p(\theta)$

- The first wisdomer touches a tail

$$p(\theta|x_1) = \frac{p_1(x_1|\theta)p(\theta)}{\int p_1(x_1|\theta)p(\theta)d\theta}$$

- The second wisdomer touches a leg and uses $p(\theta|x_1)$ as **his new prior**

$$p(\theta|x_1, x_2) = \frac{p_2(x_2|\theta)p(\theta|x_1)}{\int p_2(x_2|\theta)p(\theta|x_1)d\theta}$$

- ...

- At the end they form sharp distribution $p(\theta|x_1, \ldots, x_m)$

What is machine learning?

- ML tries to find regularities within the data
- Data is a set of objects (users, images, signals, RNAs, chemical compounds, credit histories, etc.)
- Each object is described by a set of observed variables X and a set of hidden (latent) variables T
- It is assumed that the values of hidden variables are hard to get and we have only limited number of objects with known hidden variables, so-called training set (X_{tr}, T_{tr})
- The goal is to find the way of predicting the hidden variables for a new object given the values of observed variables by adjusting the weights W of decision rule.

Discriminative probabilistic ML model

- Models $p(T, W|X)$ thus not modelling the distribution of observed variables
- Observed variables are assumed to be known for all objects
- Usually assumes that prior over W does not depend on X:

$$p(T, W|X) = p(T|X, W)p(W)$$

- Cannot generate new objects
- Example: classifier of images (T space is much easier than X space)
- More elaborate example: machine translation algorithm (T space has the same complxity as X space)

Generative probabilistic ML model

- Models joint distribution over all variables $p(X,T,W) = p(X,T|W)p(W)$
- Given trained algorithm we may generate new objects, i.e. pairs (x,t)
- May be quite difficult to train since space of X is usually much more complicated than space of T
- Example: generative adversarial network
- More weird example: AlphaGo

Training Bayesian models

- Suppose we are given training data (X_{tr}, T_{tr}) and a discriminative model $p(T, W|X)$

Training Bayesian models

- Suppose we are given training data (X_{tr}, T_{tr}) and a discriminative model $p(T, W|X)$
- At training stage we perform Bayesian inference over W:

$$p(W|X_{tr}, T_{tr}) = \frac{p(T_{tr}|X_{tr}, W)p(W)}{\int p(T_{tr}|X_{tr}, W)p(W)dW},$$

thus obtaining **ensemble** of algorithms rather than a single one

Training Bayesian models

- Suppose we are given training data (X_{tr}, T_{tr}) and a discriminative model $p(T, W|X)$
- At training stage we perform Bayesian inference over W:

$$p(W|X_{tr}, T_{tr}) = \frac{p(T_{tr}|X_{tr}, W)p(W)}{\int p(T_{tr}|X_{tr}, W)p(W)dW},$$

  thus obtaining **ensemble** of algorithms rather than a single one
- At test stage new data x arrives and we need to compute the predictive distribution on its hidden value t
- To do this we perform ensembling w.r.t. posterior over the weights W

$$p(t|x, X_{tr}, T_{tr}) = \int p(t|x, W)p(W|X_{tr}, T_{tr})dW$$

Training Bayesian models

- Ensembling **really** helps and outperforms single best algorithm within the model
- Posterior $p(W|X_{tr}, T_{tr})$ contains **all** information about dependencies between X and T that the model could extract
- If new labeled data (X'_{tr}, T'_{tr}) arrives we may skip the old training data and update our algorithm only on new data using $p(W|X_{tr}, T_{tr})$ as a new prior

Training Bayesian models

- Suppose we are given training data (X_{tr}, T_{tr}) and a discriminative model $p(T, W|X)$
- At training stage we perform Bayesian inference over W:

$$p(W|X_{tr}, T_{tr}) = \frac{p(T_{tr}|X_{tr}, W)p(W)}{\int p(T_{tr}|X_{tr}, W)p(W)dW},$$

  thus obtaining **ensemble** of algorithms rather than a single one

- At test stage new data x arrives and we need to compute the predictive distribution on its hidden value t
- To do this we perform ensembling w.r.t. posterior over the weights W

$$p(t|x, X_{tr}, T_{tr}) = \int p(t|x, W)p(W|X_{tr}, T_{tr})dW$$

Training Bayesian models

- Suppose we are given training data (X_{tr}, T_{tr}) and a discriminative model $p(T, W|X)$
- At training stage we perform Bayesian inference over W:

$$p(W|X_{tr}, T_{tr}) = \frac{p(T_{tr}|X_{tr}, W)p(W)}{\boxed{\int p(T_{tr}|X_{tr}, W)p(W)dW}},$$

Usually intractable

thus obtaining **ensemble** of algorithms rather than a single one

- At test stage new data x arrives and we need to compute the predictive distribution on its hidden value t
- To do this we perform ensembling w.r.t. posterior over the weights W

$$p(t|x, X_{tr}, T_{tr}) = \boxed{\int p(t|x, W)p(W|X_{tr}, T_{tr})dW}$$

Usually intractable

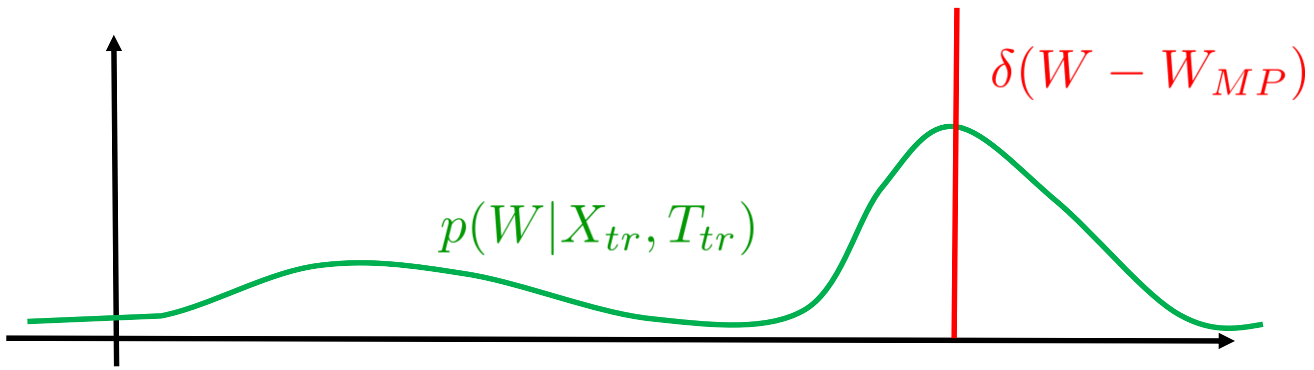
Poor man's Bayes

- Simplified probabilistic modeling
- Approximate posterior $p(W|X_{tr}, T_{tr})$ with a delta function $\delta(W - W_{MP})$
- Corresponds to point estimate of W:

$$W_{MP} = \arg\max p(W|X_{tr}, T_{tr}) = \arg\max p(T_{tr}|X_{tr}, W)p(W)$$

- Inference is more simple

$$p(T|X, X_{tr}, T_{tr}) = \int p(T|X, W)p(W|X_{tr}, T_{tr})dW \approx p(T|X, W_{MP})$$



Advantages of Bayesian framework

- Regularization
- Latent variable modeling (lecture 4)
- Extendability
- Scalability (lecture 5, 10)

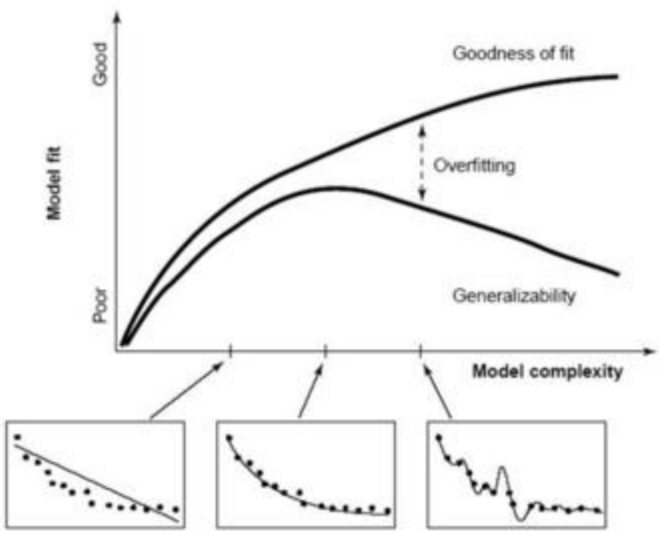
Regularization

- By establishing priors over the weights θ we may **regularize** maximum likelihood estimates

Prior term

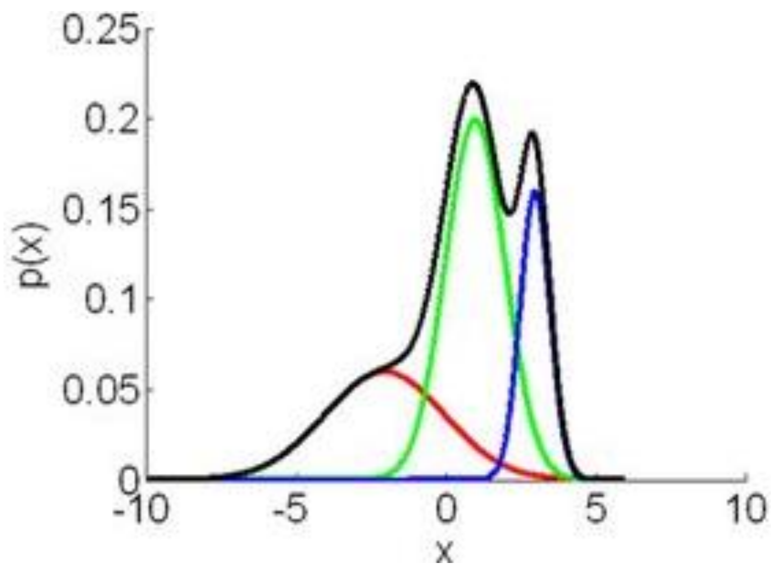
$$\cancel{p(X_{tr}, T_{tr}|\theta) \to \max_{\theta}} \qquad p(\theta|X_{tr}, T_{tr}) = \frac{p(X_{tr}, T_{tr}|\theta)\boxed{p(\theta)}}{\int p(X_{tr}, T_{tr}|\theta)p(\theta)d\theta}$$

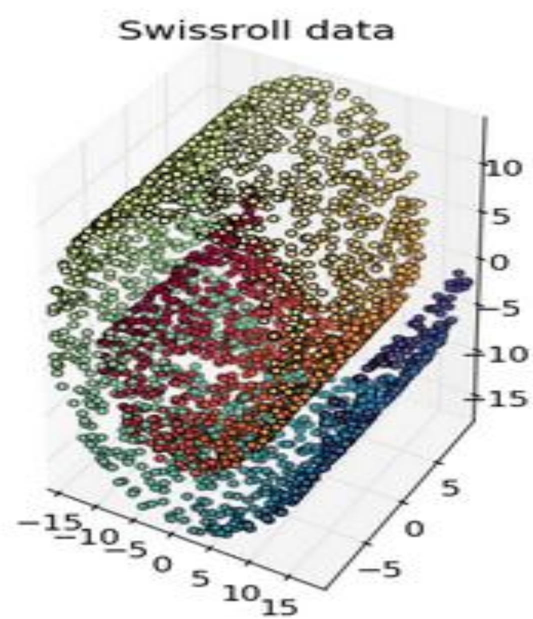
- Prevents overfitting
- We can set the best prior automatically by performing Bayesian **model selection**



Latent variable modeling

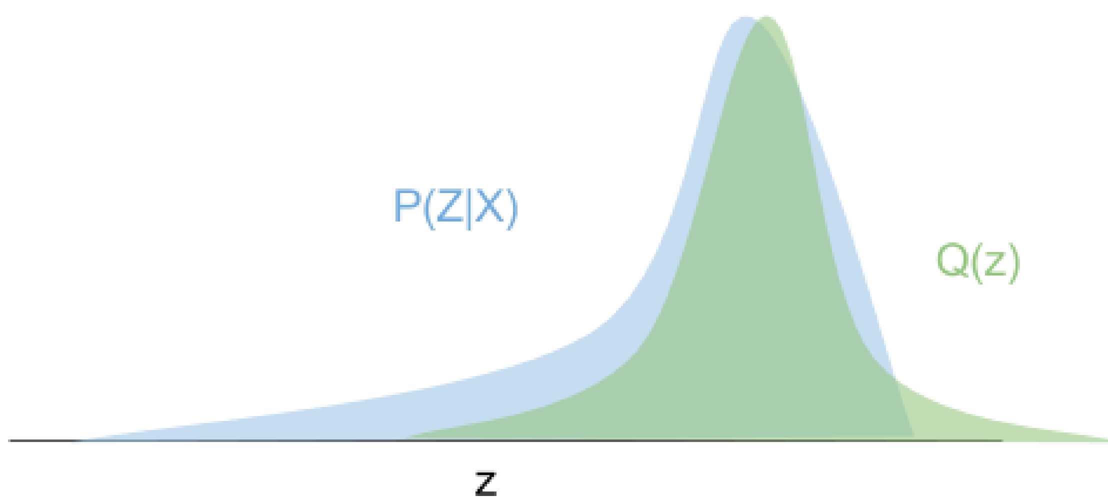
- We may build models with latent variables that are unknown at training stage
- Allows to process missing data
- Allows to build and train much more complicated **mixture models**





Scalability

- Bayesian methods were traditionally considered as computationally expensive
- Recently the situation has changed dramatically
- New mathematical tools for scalable variational approximations and MCMC algorithms
- Now applicable to large datasets and high dimensions



Kullback-Leibler divergence

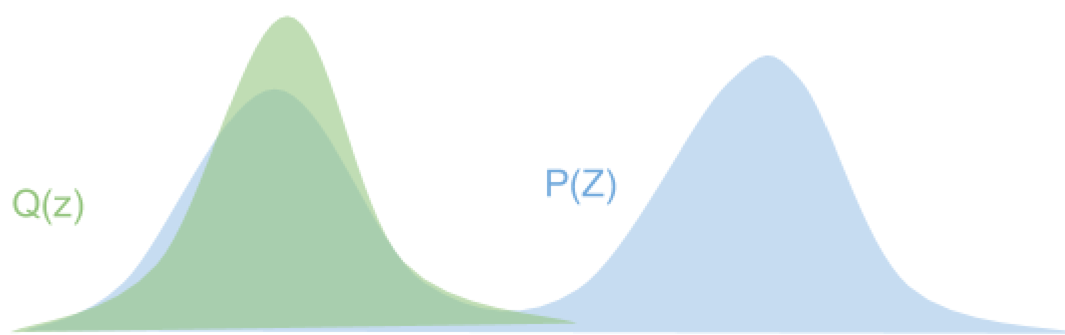
- A good mismatch measure between two distributions over **the same domain**

$$KL(q(x)||p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_q \log \frac{q(x)}{p(x)} \geq 0$$

- Information-theoretic interpretation

$$\texttt{KL} = \texttt{CrossEntropy} - \texttt{Entropy}$$

- If we minimize KL w.r.t. $q(.)$ the approximation should be good where $q(x)$ has large values



Kullback-Leibler divergence

- Let us prove non-negativity of KL. Consider

$$-KL(q(x)||p(x)) = \int q(x) \log \frac{p(x)}{q(x)} dx$$

Kullback-Leibler divergence

- Let us prove non-negativity of KL. Consider

$$-KL(q(x)||p(x)) = \int q(x) \log \frac{p(x)}{q(x)} dx$$

- Recall that logarithm is a concave function and apply Jensen inequality

$$\int q(x) \log \frac{p(x)}{q(x)} dx \leq \log \int q(x) \frac{p(x)}{q(x)} dx = \log \int p(x) dx = \log 1 = 0$$

Kullback-Leibler divergence

- Let us prove non-negativity of KL. Consider

$$-KL(q(x)||p(x)) = \int q(x) \log \frac{p(x)}{q(x)} dx$$

- Recall that logarithm is a concave function and apply Jensen inequality

$$\int q(x) \log \frac{p(x)}{q(x)} dx \leq \log \int q(x) \frac{p(x)}{q(x)} dx = \log \int p(x) dx = \log 1 = 0$$

- Any concave function $f(.)$ such that $f(1) = 0$ defines its own divergence
- KL is a particular case of a more general family of divergences

Conclusion

- Bayesian framework is an alternative approach to building probabilistic models
- Bayesian ML has several advantages over traditional models
- It DOES NOT contradict or deny frequentist framework – this is just another tool for data scientist