

Day 2

Probabilistic Models with Deep Neural Networks

Andrés Masegosa

*Department of Mathematics
University of Almería
Spain*

Masegosa, A. R., Cabañas, R., Langseth, H., Nielsen, T. D., & Salmern, A. (2019). Probabilistic Models with Deep Neural Networks. arXiv preprint arXiv:1908.03442.

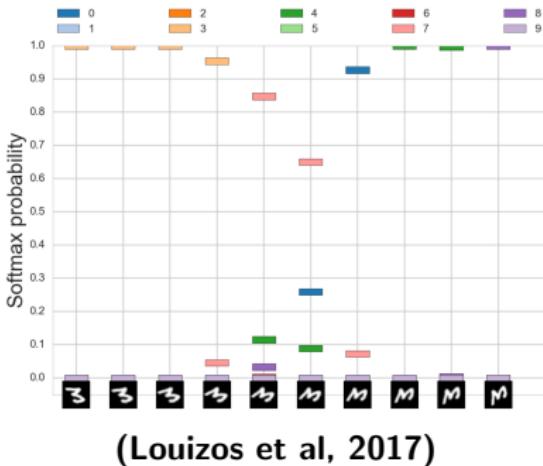
Day 1: Introduction to probabilistic programming languages

- Why do we need PPLs?
- Probabilistic programming in Pyro
- Hand-on exercises:
 - Probability Distributions in Pyro.
 - Probabilistic Models in Pyro.
 - Ice-cream Shop Model.

Day 2: Probabilistic Models with Deep Neural Networks

- Uncertainty in Machine Learning
- Variational Inference
- Supervised/Unsupervised Learning
- Hand-on exercises
 - Bayesian Neural Networks
 - Variational Auto-encoders

Uncertainty with Machine Learning



Why is important to model uncertainty?

- To assess **confidence in the predictions**.
- To know **what I don't know**.
- **Big implications** in many problems.



Why is important to model uncertainty?

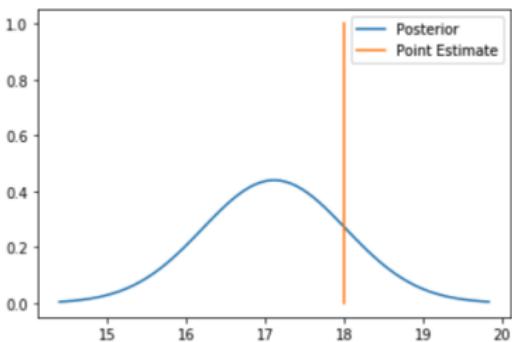
- To assess **confidence in the predictions**.
- To know **what I don't know**.
- **Big implications** in many problems.



Why is important to model uncertainty?

- To assess **confidence in the predictions**.
- To know **what I don't know**.
- **Big implications** in many problems.

Uncertainty with Machine Learning

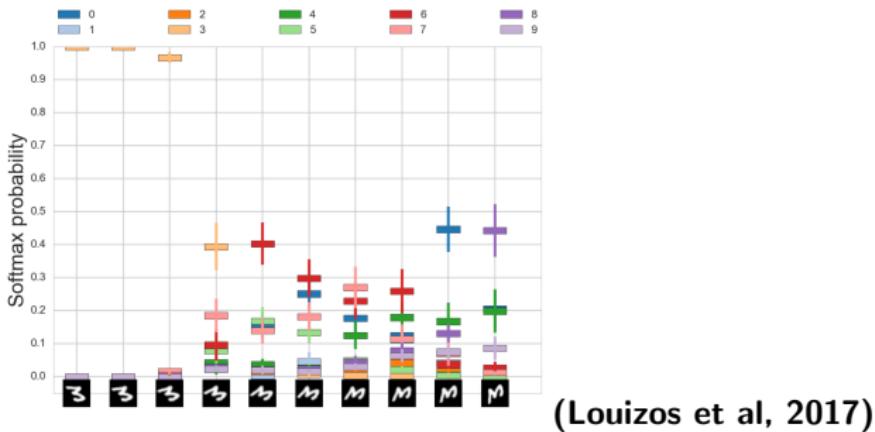


```
P(Temperature | Sensor=18.0) =  
Normal(loc: 17.115509033203125, scale: 0.9070338010787964)
```

Probabilistic Models naturally quantify uncertainty

- Everything is defined in terms of **random variables**.
- **Standard language** to model uncertainty.

Uncertainty with Machine Learning



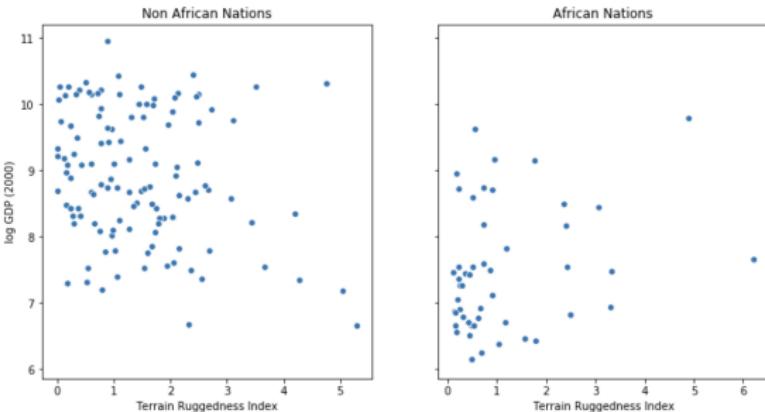
Probabilistic Models naturally quantify uncertainty

- Everything is defined in terms of **random variables**.
- **Standard language** to model uncertainty.

Example

Real Data Example

Nunn, N. & Puga, D., Ruggedness: The blessing of bad geography in Africa,
Review of Economics and Statistics 94(1), Feb. 2012

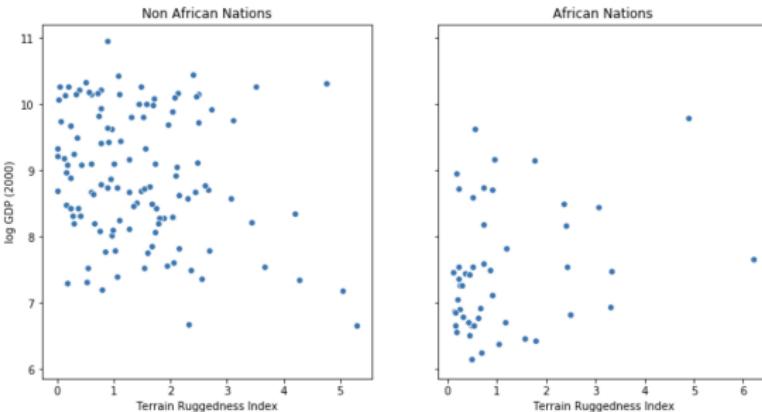


Relationship between topographic heterogeneity and GDP per capita

- Is Terrain ruggedness or **bad geography** is related to poorer economic performance outside of Africa?

Real Data Example

Nunn, N. & Puga, D., Ruggedness: The blessing of bad geography in Africa,
Review of Economics and Statistics 94(1), Feb. 2012



Relationship between topographic heterogeneity and GDP per capita

- Is Terrain ruggedness or **bad geography** is related to poorer economic performance outside of Africa?
- Does **rugged terrains** have had a reverse effect on income for African nations?

④ Linear Regression Model:

$$y = w_0 + w_1 \cdot x$$

① Linear Regression Model:

$$y = w_0 + w_1 \cdot x$$

② Mean Square Error Loss:

$$L(\theta) = \sum_{i=1}^n (y_i - w_0 + w_1 \cdot x_i)^2$$

① Linear Regression Model:

$$y = w_0 + w_1 \cdot x$$

② Mean Square Error Loss:

$$L(\theta) = \sum_{i=1}^n (y_i - w_0 + w_1 \cdot x_i)^2$$

③ Minimization problem (Gradient Descent):

$$\arg \min_{\theta} L(\theta)$$

$$\theta^{t+1} = \theta^t - \nabla_{\theta} L(\theta)$$

① Linear Regression Model:

$$y = w_0 + w_1 \cdot x$$

② Mean Square Error Loss:

$$L(\theta) = \sum_{i=1}^n (y_i - w_0 + w_1 \cdot x_i)^2$$

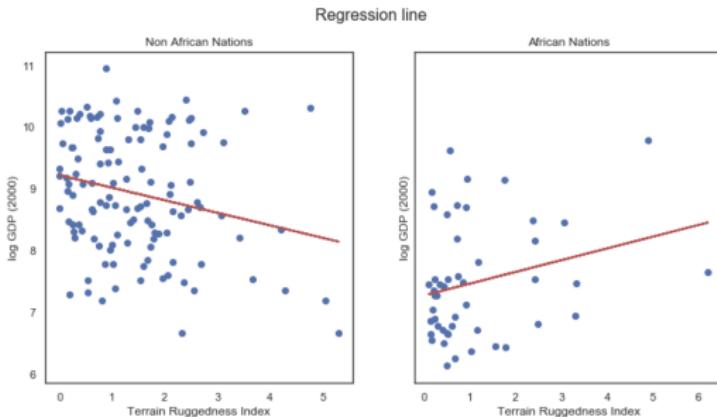
③ Minimization problem (Gradient Descent):

$$\arg \min_{\theta} L(\theta)$$

$$\theta^{t+1} = \theta^t - \nabla_{\theta} L(\theta)$$

[Day2/students_Bayesian_regression.ipynb](#)

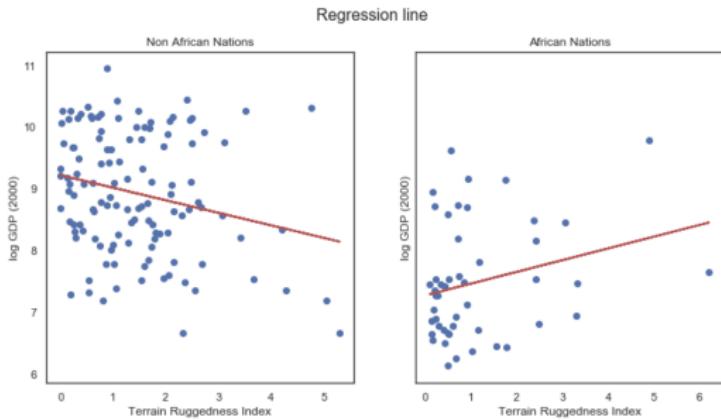
Real Data Example



Linear Regression Model

- Negative slope for Non African Nations.
- Positive slope for African Nations.

Real Data Example



Linear Regression Model

- Negative slope for Non African Nations.
- Positive slope for African Nations.

Are we 100% sure about the conclusion?

Bayesian Machine Learning

① Probabilistic Linear Regression Model:

$$y_i | x_i, \mathbf{w}, \sigma^2 \sim \mathcal{N}(\mu = w_0 + w_1 \cdot x_i, \sigma^2)$$

① Probabilistic Linear Regression Model:

$$y_i | x_i, \mathbf{w}, \sigma^2 \sim \mathcal{N}(\mu = w_0 + w_1 \cdot x_i, \sigma^2)$$

② Model Parameters are Random Variables:

$$w_0, w_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 100)$$

$$\frac{1}{\sigma^2} \sim \mathcal{G}(\alpha = 1, \beta = 1)$$

① Probabilistic Linear Regression Model:

$$y_i | x_i, \mathbf{w}, \sigma^2 \sim \mathcal{N}(\mu = w_0 + w_1 \cdot x_i, \sigma^2)$$

② Model Parameters are Random Variables:

$$w_0, w_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 100)$$

$$\frac{1}{\sigma^2} \sim \mathcal{G}(\alpha = 1, \beta = 1)$$

③ We compute the posterior (Bayes' rule):

$$p(w_0, w_1, \sigma^2 | D) = \frac{\prod_{i=1}^n p(y_i | x_i, w_0, w_1, \sigma^2) p(w_0, w_1, \sigma^2)}{p(y_1, \dots, y_n | x_1, \dots, x_n)}$$

① Probabilistic Linear Regression Model:

$$y_i | x_i, \mathbf{w}, \sigma^2 \sim \mathcal{N}(\mu = w_0 + w_1 \cdot x_i, \sigma^2)$$

② Model Parameters are Random Variables:

$$w_0, w_1 \sim \mathcal{N}(\mu = 0, \sigma^2 = 100)$$

$$\frac{1}{\sigma^2} \sim \mathcal{G}(\alpha = 1, \beta = 1)$$

③ We compute the posterior (Bayes' rule):

$$p(w_0, w_1, \sigma^2 | D) = \frac{\prod_{i=1}^n p(y_i | x_i, w_0, w_1, \sigma^2) p(w_0, w_1, \sigma^2)}{p(y_1, \dots, y_n | x_1, \dots, x_n)}$$

[Day2/students_Bayesian_regression.ipynb](#)

Exercise: Bayesian Logistic Regression

- Predicts whether a country is African or not based on ruggedness and GDP.

`Day2/students_Bayesian_Regression.ipynb`

- A probabilistic model is a joint distribution of hidden variables \mathbf{z} and observed variables \mathbf{x} ,

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- A probabilistic model is a joint distribution of hidden variables \mathbf{z} and observed variables \mathbf{x} ,

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- Inference about the unknowns is through the **posterior**, the conditional distribution of the hidden variables given the observations

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

The posterior quantify our **uncertainty** in the model.

- A probabilistic model is a joint distribution of hidden variables \mathbf{z} and observed variables \mathbf{x} ,

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- Inference about the unknowns is through the **posterior**, the conditional distribution of the hidden variables given the observations

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

The posterior quantify our **uncertainty** in the model.

- For most interesting models, the denominator is not **tractable**.

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

- A probabilistic model is a joint distribution of hidden variables \mathbf{z} and observed variables \mathbf{x} ,

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- Inference about the unknowns is through the **posterior**, the conditional distribution of the hidden variables given the observations

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

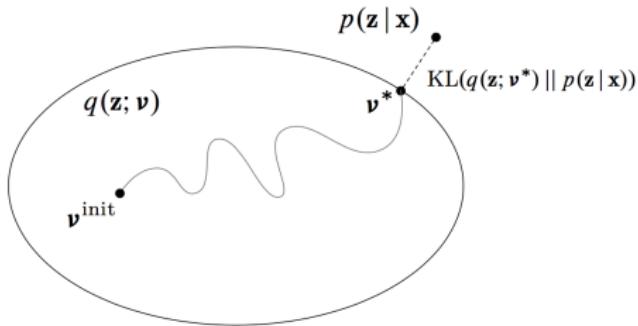
The posterior quantify our **uncertainty** in the model.

- For most interesting models, the denominator is not **tractable**.

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

We have to use **approximate posterior inference**.

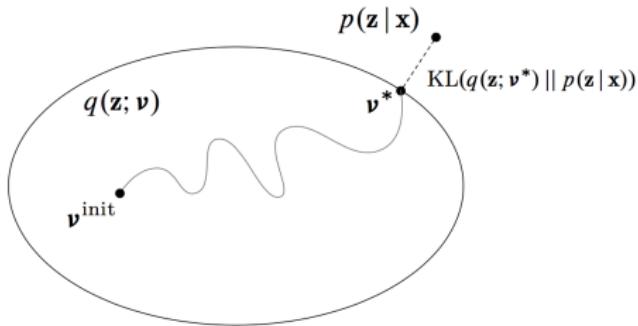
Variational Inference



- **Variational Inference** is an approximate method,

$$q(\mathbf{z}|\boldsymbol{\nu}) \approx p(\mathbf{z}|\mathbf{x})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).

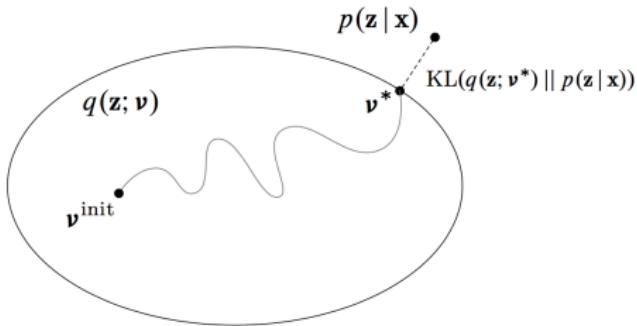


- **Variational Inference** is an approximate method,

$$q(\mathbf{z}|\boldsymbol{\nu}) \approx p(\mathbf{z}|\mathbf{x})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).
- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} \text{KL}\left(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{x})\right)$$



- **Variational Inference** is an approximate method,

$$q(\mathbf{z}|\boldsymbol{\nu}) \approx p(\mathbf{z}|\mathbf{x})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).
- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu}) || p(\mathbf{z}|\mathbf{x}))$$

- Solve using a **optimization algorithm** (i.e. gradient descent).



$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu}) || p(\mathbf{z}|\mathbf{x}))$$

- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu}) || p(\mathbf{z}|\mathbf{x}))$$

- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu}) || p(\mathbf{z}|\mathbf{x}))$$

- After some manipulations,

$$\ln p(\mathbf{x}) = \mathcal{L}(\boldsymbol{\nu}) - KL(q(\mathbf{z}|\boldsymbol{\nu}) || p(\mathbf{z}|\mathbf{x}))$$

- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{x}))$$

- After some manipulations,

$$\ln p(\mathbf{x}) = \mathcal{L}(\boldsymbol{\nu}) - KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{x}))$$

- $\mathcal{L}(\boldsymbol{\nu})$ is known as the **ELBO function**.

$$\ln p(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\nu}) = \mathbb{E}_q[\ln p(\mathbf{x}|\mathbf{z})] - KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}))$$

- Find the **best possible approximation**,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{x}))$$

- After some manipulations,

$$\ln p(\mathbf{x}) = \mathcal{L}(\boldsymbol{\nu}) - KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}|\mathbf{x}))$$

- $\mathcal{L}(\boldsymbol{\nu})$ is known as the **ELBO function**.

$$\ln p(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\nu}) = \mathbb{E}_q[\ln p(\mathbf{x}|\mathbf{z})] - KL(q(\mathbf{z}|\boldsymbol{\nu})||p(\mathbf{z}))$$

- The above minimization problem is **equivalent to maximize the ELBO function**,

$$\boldsymbol{\nu}^* = \arg \max_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu})$$

- ① Define a tractable approximating **variational family**,

$$q(\mathbf{z}|\boldsymbol{\nu})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).

- ① Define a tractable approximating **variational family**,

$$q(\mathbf{z}|\boldsymbol{\nu})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).

- ② Solve this **maximization problem**,

$$\boldsymbol{\nu}^* = \arg \max_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu})$$

- ① Define a tractable approximating **variational family**,

$$q(\mathbf{z}|\boldsymbol{\nu})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).

- ② Solve this **maximization problem**,

$$\boldsymbol{\nu}^* = \arg \max_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu})$$

- ③ It can be solved by (stochastic) **gradient ascent methods**:

$$\boldsymbol{\nu}^{(t+1)} = \boldsymbol{\nu}^{(t)} + \alpha \nabla_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu}^{(t)})$$

- ① Define a tractable approximating **variational family**,

$$q(\mathbf{z}|\boldsymbol{\nu})$$

- $q(\mathbf{z}|\boldsymbol{\nu})$ is a simpler distribution (e.g. diagonal covariance matrix, unimodal).

- ② Solve this **maximization problem**,

$$\boldsymbol{\nu}^* = \arg \max_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu})$$

- ③ It can be solved by (stochastic) **gradient ascent methods**:

$$\boldsymbol{\nu}^{(t+1)} = \boldsymbol{\nu}^{(t)} + \alpha \nabla_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu}^{(t)})$$

- ④ Use the variational approximation as a **proxy**,

$$q(\mathbf{z}|\boldsymbol{\nu}) \approx p(\mathbf{z}|\mathbf{x})$$

Variational Inference in Pyro

```
1 #The observations
2 obs = {'sensor': torch.tensor(18.0)}
3
4 def model(obs):
5     temp = pyro.sample('temp', dist.Normal(15.0, 2.0))
6     sensor = pyro.sample('sensor', dist.Normal(temp, 1.0), obs=obs['sensor'])
```

Pyro's Model

- We have a **probabilistic model**,

$$p(\text{temp}) = \mathcal{N}(15.0, 2.0)$$

$$p(\text{sensor}|\text{temp}) = \mathcal{N}(\text{sensor}, 2.0)$$

```
1 #The guide
2 def guide(obs):
3     a = pyro.param("mean", torch.tensor(0.0))
4     b = pyro.param("scale", torch.tensor(1.), constraint=constraints.positive)
5     temp = pyro.sample('temp', dist.Normal(a, b))
```

Pyro's Guides

- We want to compute the **posterior**, $p(\text{temp}|\text{sensor} = 18.0)$
- We define, $q(\text{temp}|\boldsymbol{\nu}) = \mathcal{N}(\boldsymbol{\nu}_a, \boldsymbol{\nu}_b)$

```
1 #The guide
2 def guide(obs):
3     a = pyro.param("mean", torch.tensor(0.0))
4     b = pyro.param("scale", torch.tensor(1.), constraint=constraints.positive)
5     temp = pyro.sample('temp', dist.Normal(a, b))
```

Pyro's Guides

- We want to compute the **posterior**, $p(\text{temp}|\text{sensor} = 18.0)$
- We define, $q(\text{temp}|\boldsymbol{\nu}) = \mathcal{N}(\boldsymbol{\nu}_a, \boldsymbol{\nu}_b)$
- **Guide's requirements:**
 - Model and Guide have both **same input signature**.
 - All **unobserved sample statements** in the model must appear in the guide.

```
: pyro.clear_param_store()
svi = pyro.infer.SVI(model=temperature_model,
                     guide=guide,
                     optim=SGD({"lr": 0.001, "momentum":0.1}),
                     loss=Trace_ELBO())

for t in range(num_steps):
    svi.step(obs)
```

Pyro's Variational Inference

- We run **optimization** to solve,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\text{temp}|\boldsymbol{\nu}) || p(\text{temp}|\text{sensor} = 18.0))$$

```
: pyro.clear_param_store()
svi = pyro.infer.SVI(model=temperature_model,
                     guide=guide,
                     optim=SGD({"lr": 0.001, "momentum":0.1}),
                     loss=Trace_ELBO())

for t in range(num_steps):
    svi.step(obs)
```

Pyro's Variational Inference

- We run **optimization** to solve,

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\text{temp}|\boldsymbol{\nu}) || p(\text{temp}|\text{sensor} = 18.0))$$

- It can be solved by (stochastic) **gradient ascent methods**:

$$\boldsymbol{\nu}^{(t+1)} = \boldsymbol{\nu}^{(t)} + \alpha \nabla_{\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{\nu}^{(t)})$$

Exercise: Pyro implementation for a simple Gaussian model

- ① Implement a Pyro's **model** for a Normal distribution with prior over mean and (inverse) of variance. The variable x is always observed.

$$p(\mu) = \text{Normal}(0, 10000.0)$$

$$p(\gamma) = \text{Gamma}(1, 1)$$

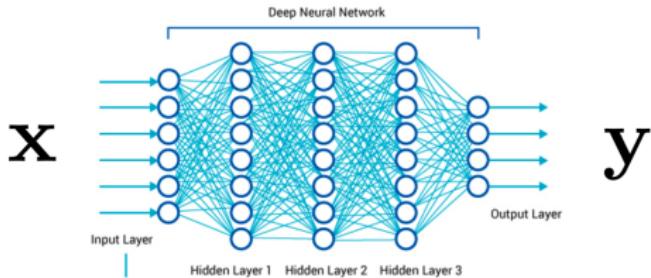
$$p(x | \mu, \gamma) = \text{Norma}(\mu, 1/\gamma)$$

- ② Define a Pyro's **guide** for approximating the posterior,

$$q(\mu) = \text{Norma}(\nu_\mu, \nu_{\sigma^2})$$

$$q(\gamma) = \text{Gammal}(\nu_\alpha, \nu_\beta)$$

Day2/student_simple_model.ipynb



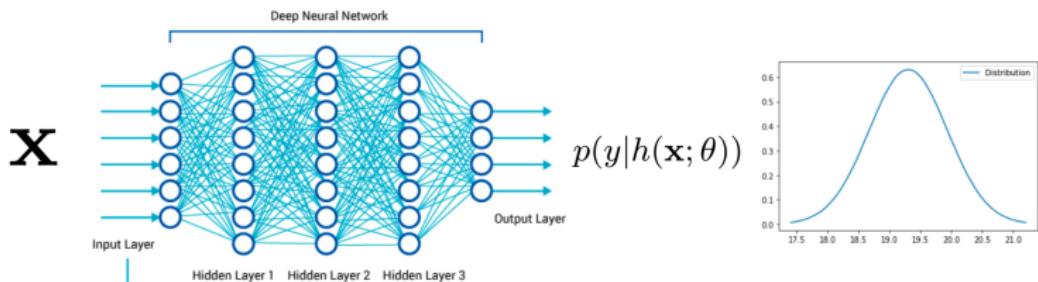
Neural Network

- $h(\cdot; \theta)$ encodes a neural network (i.e. a non-linear function).

$$\mathbf{y} = h(\mathbf{x}; \theta)$$

- Loss minimization,

$$\arg \min_{\theta} \sum_{i=1}^n \ell(h(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

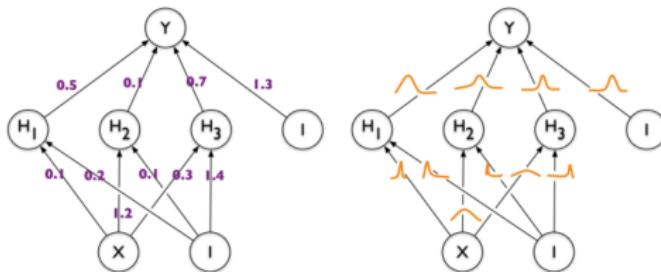


Bayesian Neural Network

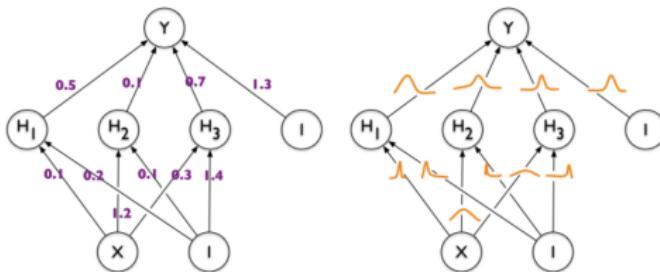
- E.g. Neural regressor,

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mu = h(\mathbf{x}; \theta_h), \sigma^2 = g(\mathbf{x}; \theta_g))$$

$h(\cdot; \theta)$ encodes a neural network.



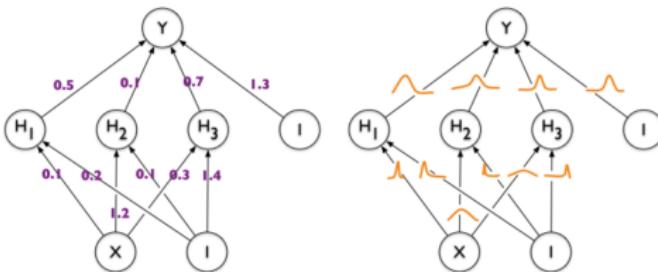
Bayesian Neural Network



Bayesian Neural Network

- **Variational Inference,**

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\theta|\boldsymbol{\nu}) || p(\theta|\mathbf{y}_1, \mathbf{x}_1, \dots, \mathbf{y}_n, \mathbf{x}_n))$$



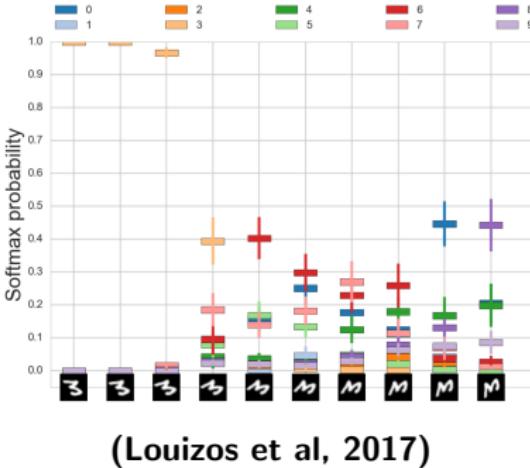
Bayesian Neural Network

- **Variational Inference,**

$$\boldsymbol{\nu}^* = \arg \min_{\boldsymbol{\nu}} KL(q(\theta|\boldsymbol{\nu}) || p(\theta|\mathbf{y}_1, \mathbf{x}_1, \dots, \mathbf{y}_n, \mathbf{x}_n))$$

- Bayesian predictive is an **ensemble of neural networks**,

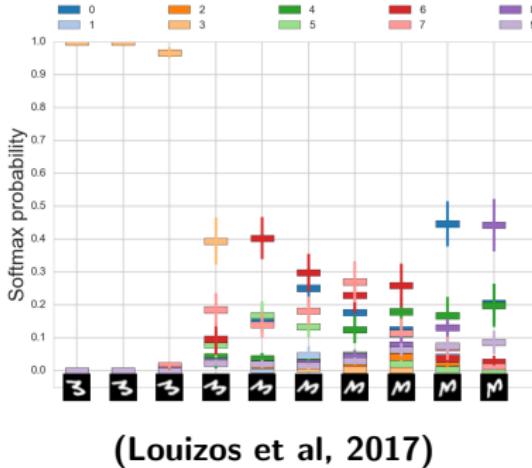
$$\begin{aligned} p(y|\mathbf{x}) &= \int p(y|\mathbf{x}, \theta) q(\theta|\boldsymbol{\nu}^*) d\theta \\ &\approx \frac{1}{M} \sum_{\theta_j \sim q(\theta|\boldsymbol{\nu}^*)} p(y|\mathbf{x}, \theta_j) \end{aligned}$$



(Louizos et al, 2017)

Bayesian Neural Networks

- Capture the uncertainty.
- Better identify low confidence predictions (**They know what they don't know**).

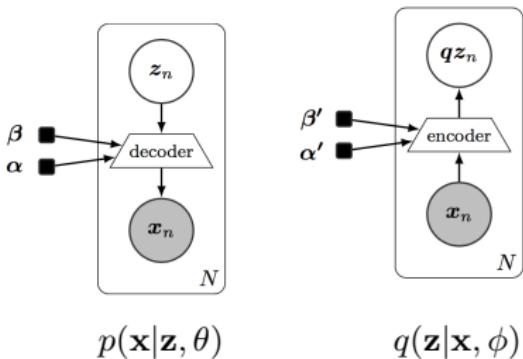


Bayesian Neural Networks

- Capture the uncertainty.
- Better identify low confidence predictions (**They know what they don't know**).

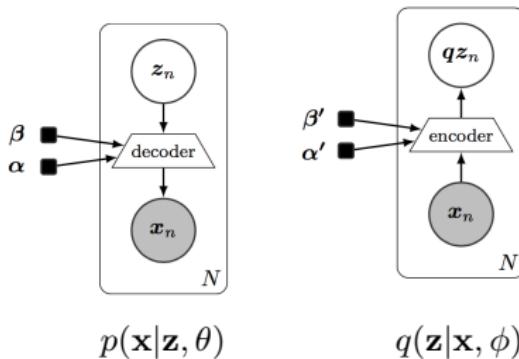
Day2/BayesianNeuralNetworks.ipynb

Unsupervised Probabilistic Models with Deep Neural Networks



Unsupervised Learning with Deep Neural Networks

- **No labeled data** (extract structure from data).

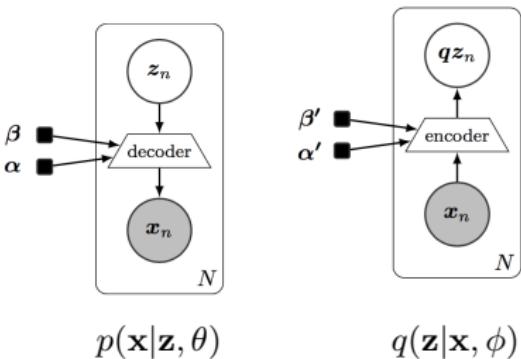


Unsupervised Learning with Deep Neural Networks

- **No labeled data** (extract structure from data).
- **A generative model,**

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \theta)$$



Unsupervised Learning with Deep Neural Networks

- **No labeled data** (extract structure from data).
- A **generative model**,

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \theta)$$

- **Variational Inference**,

$$q(\mathbf{z}|\mathbf{x}, \phi) \approx p(\mathbf{z}|\mathbf{x}, \theta)$$



Unsupervised Learning with Deep Neural Networks

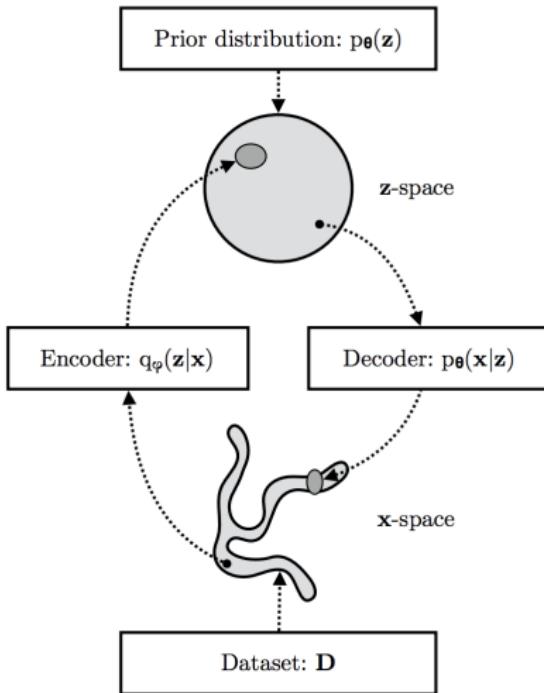
- **No labeled data** (extract structure from data).
- **A generative model,**

$$\mathbf{z} \sim p(\mathbf{z})$$

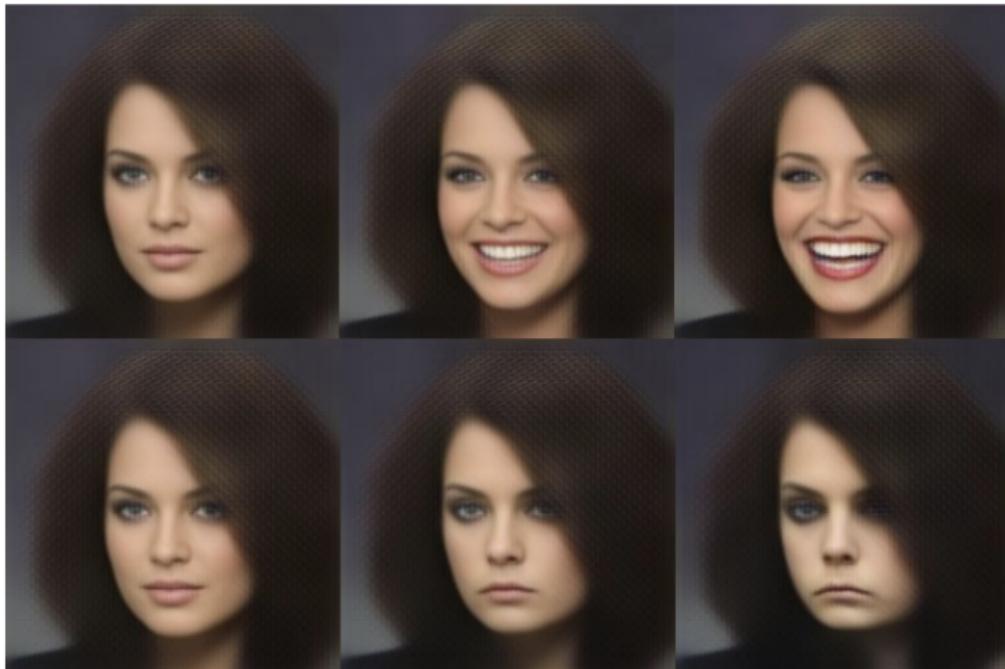
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \theta)$$

- **Variational Inference,**

$$q(\mathbf{z}|\mathbf{x}, \phi) \approx p(\mathbf{z}|\mathbf{x}, \theta)$$



Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. arXiv preprint arXiv:1906.02691.



(White et al. 2016)

Exercise: Bayesian PCA in Pyro

- VAE using linear transformations.
- Applied to MNIST data set.

`Day2/BayesianPCA.ipynb`

Exercise: Bayesian PCA in Pyro

- VAE using linear transformations.
- Applied to MNIST data set.

`Day2/BayesianPCA.ipynb`

Exercise: Variational Auto-Encoder in Pyro

- VAE using simple neural network.
- Applied to MNIST data set.

`Day2/VAE.ipynb`

- **Probabilistic modeling is a key aspect in AI systems**
 - Confidence in model predictions/actions.
 - To know what you do not know.

- **Probabilistic modeling is a key aspect in AI systems**
 - Confidence in model predictions/actions.
 - To know what you do not know.
- **PPLs are the right tool for probabilistic modeling.**
 - Enormous expressibility.
 - Powerful inference engines (BlackBox Variational Inference).



<https://inferpy.readthedocs.io>



<http://www.amidsttoolbox.com/>

Thanks!!!! :)