Assorted Topics (2)

CS772A: Probabilistic Machine Learning

Piyush Rai

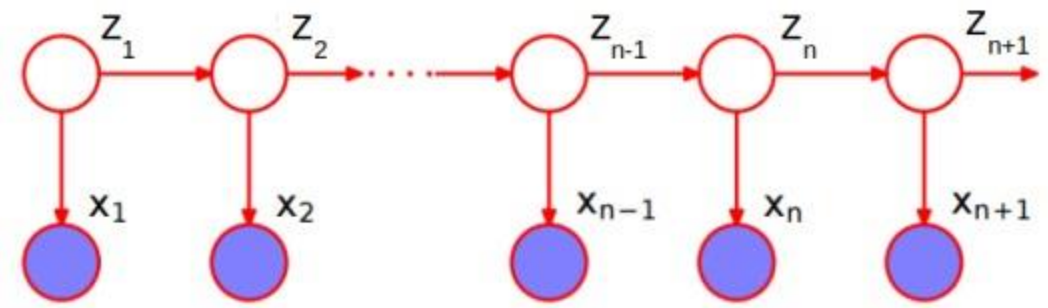
Plan today

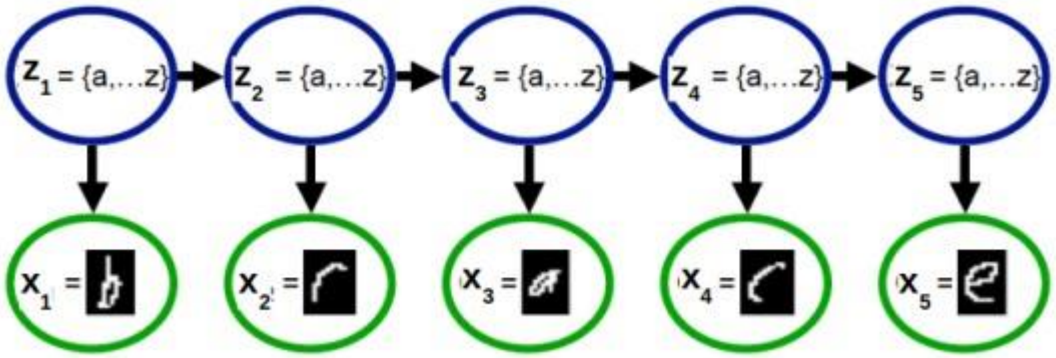
- Probabilistic models for sequential data
  - HMM and State-Space Models (SSM)
- Frequentist approach for estimating uncertainty
- Estimating uncertainty using a single model
  - Evidential Learning

Probabilistic Models for Sequential Data

Latent Variable Models for Sequential Data

- Task: Given a sequence of observations, infer the latent state of each observation





Observation model

$$\boldsymbol{x}_n|\boldsymbol{z}_n \sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) \quad \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)}$$

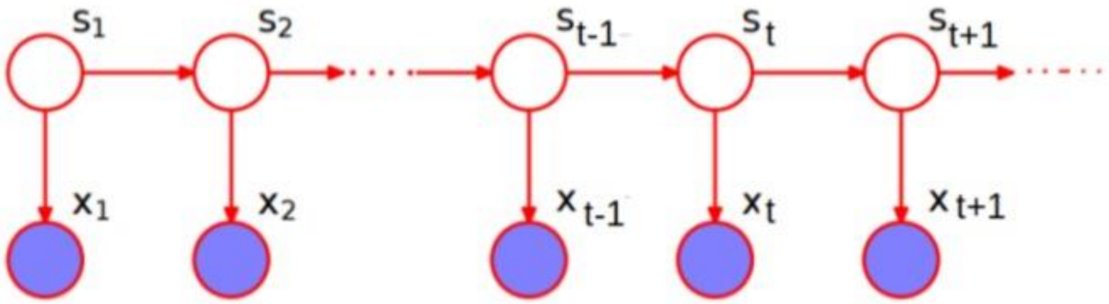
State-transition model

$$\boldsymbol{z}_n|\boldsymbol{z}_{n-1} \sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)}$$

- If z_n's are discrete, we have a hidden Markov model (HMM) $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}=\ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$
- If z_n's are real-valued, we have a state-space model (SSM) $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$

State-Space Models

- In the most general form, the state-transition and observation models of an SSM



Using '**s**' instead of '**z**' to refer to states

Using 't' to denote the 'time-step'

HMM is similar to SSM except the state-transition model is a discrete distribution

g_t, h_t can be linear or nonlinear functions

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = g_t(\boldsymbol{s}_{t-1}) + \epsilon_t \qquad \text{(must be a cont. dist. over } \boldsymbol{s}_t\text{)}$$

$$\boldsymbol{x}_t|\boldsymbol{s}_t = h_t(\boldsymbol{s}_t) + \delta_t \qquad \text{(can be any dist. over } \boldsymbol{x}_t\text{)}$$

- Assuming Gaussian noise in the state-transition and observation models

This is a Gaussian SSM

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|g_t(\boldsymbol{s}_{t-1}), \mathbf{Q}_t)$$

$$\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|h_t(\boldsymbol{s}_t), \mathbf{R}_t)$$

If g_t, h_t, Q_t, R_t are independent of t then it is called a stationary model

g_t, h_t, Q_t, R_t may be known or can be learned

State-Space Models: A Simple Example

- Consider the linear Gaussian SSM

$$\boldsymbol{s}_t | \boldsymbol{s}_{t-1} = \mathbf{A}_t \boldsymbol{s}_{t-1} + \epsilon_t$$
$$\boldsymbol{x}_t | \boldsymbol{s}_t = \mathbf{B}_t \boldsymbol{s}_t + \delta_t$$

- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object
- Suppose $\boldsymbol{s}_t \in \mathbb{R}^6$ denotes the "state" vector

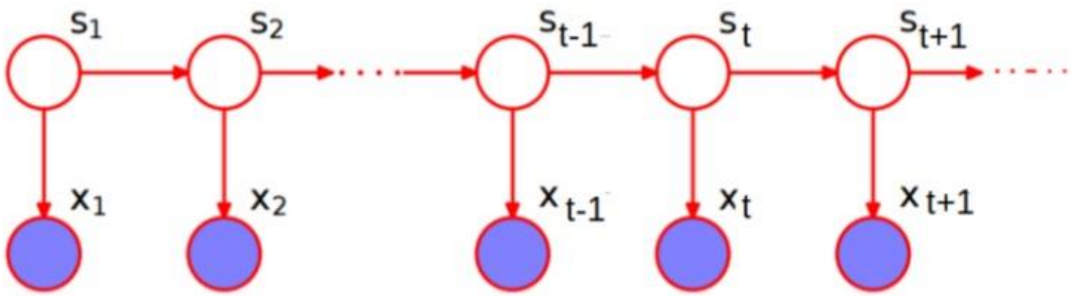
$$\boldsymbol{s}_t = [\text{pos1, vel1, accel1, pos2, vel2, accel2}]$$

- Here is an example SSM for this problem with pre-defined $\mathbf{A_t}$ and $\mathbf{B_t}$ matrices

$$\mathbf{s}_t = \underbrace{\boxed{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha\Delta t} \end{bmatrix}}}_{\mathbf{A}_t} \mathbf{s}_{t-1} + \epsilon_t \qquad \mathbf{x}_t = \underbrace{\boxed{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}}_{\mathbf{B}_t} \mathbf{s}_t + \delta_t$$

Typical Inference Task for Gaussian SSM

- One of the key tasks: Given sequence $x_1, x_2, \ldots, x_T$, infer latent $s_1, s_2, \ldots, s_T$



- Usually two ways of inferring the latent states
  - Infer $p(s_t|x_1, x_2, \ldots, x_t)$: Called the "filtering" problem

$$p(\mathbf{s}_t|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) \propto \underbrace{p(\mathbf{x}_t|\mathbf{s}_t)}_{\mathcal{N}(\mathbf{x}_t|\mathbf{B}\mathbf{s}_t, \mathbf{R})} \int \underbrace{p(\mathbf{s}_t|\mathbf{s}_{t-1})}_{\mathcal{N}(\mathbf{s}_t|\mathbf{A}\mathbf{s}_{t-1}, \mathbf{Q})} p(\mathbf{s}_{t-1}|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{t-1}) d\mathbf{s}_{t-1}$$

Turns out to be another Gaussian

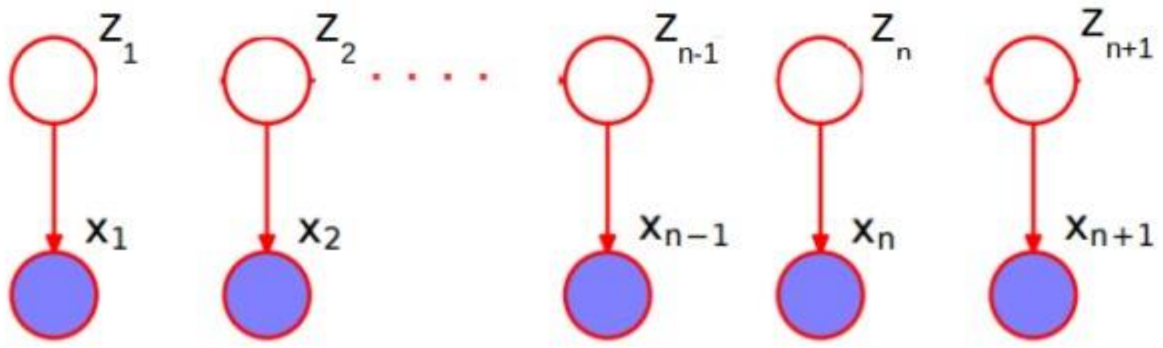
A Gaussian

Kalman Filtering is a popular algorithm for a linear Gaussian SSM

  - Infer $p(s_t|x_1, x_2, \ldots, x_t, \ldots, x_T)$: Called the "smoothing" problem
- Some other tasks one can solve for using an SSM
  - Predicting future states $p(s_{t+h}|x_1, x_2, \ldots, x_t)$ for $h \geq 1$, given observations thus far
  - Predicting future observations $p(x_{t+h}|x_1, x_2, \ldots, x_t)$ for $h \geq 1$, given observations thus far

A Special Case

- What if we have i.i.d. latent states, i.e.,. $p(z_n|z_{n-1}) = p(z_n)$?



- Discrete case (HMM) becomes a simple mixture model $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- Real-valued case (SSM) becomes a PPCA model $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Inference algos for HMM/SSM are thus very similar to that of mixture models/PPCA
  - Only main difference is how the latent variables z_n's are inferred since they aren't i.i.d.
  - E.g., if using EM, only E step needs to change (Bishop Chap 13 has EM for HMM and SSM)

Frequentist Statistics

(vs Bayesian Statistics)

Frequentist Statistics

- The Bayesian approach treats parameters/model unknowns as random variables
- In the Bayesian approach, the posterior over these r.v.'s help capture the uncertainty
- The Frequentist approach is a different way to capture uncertainty
  - Don't treat parameters as r.v. but as fixed unknowns
  - Treat parameters as a function of the dataset, e.g., $\hat{\theta}(\mathcal{D}) = \pi(\mathcal{D})$
  - Variations in param estimates over different datasets represents their uncertainty

This can be some point estimate, e.g., MLE, MAP, method of moments, etc.

A random dataset drawn from the true data distribution

True unknown value of the parameter

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\boldsymbol{\theta}^*) : n = 1 : N\} \qquad (s = 1, 2, \ldots, S)$$

The estimated distribution of the parameters given any randomly drawn dataset from the true data distribution

$$p(\pi(\tilde{\mathcal{D}}) = \theta|\tilde{\mathcal{D}} \sim \theta^*) \approx \frac{1}{S}\sum_{s=1}^{S}\delta(\theta = \pi(\tilde{\mathcal{D}}^{(s)}))$$

Param estimate using the s-th sampled dataset

As $S \to \infty$, this is known as the "sampling distribution" of the estimator

Note that sampling distribution is different from a posterior distribution we infer in Bayesian learning (there, we condition on a fixed training set)

But if the estimator is MLE and Bayesian method's prior is uniform, then both distributions are very similar (sampling distribution is often called "poor man's posterior"

Approximating the sampling distribution

- Since the true $\boldsymbol{\theta}^*$ is not known, we can't compute the sampling distribution exactly

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\boldsymbol{\theta}^*) : n = 1 : N\} \qquad (s = 1, 2, \ldots, S)$$

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\tilde{\mathcal{D}}^{(s)}))$$

- Bootstrap is a popular method to approximate the sampling distribution
- Two types of bootstrap methods: parametric and nonparametric bootstrap

Parametric Bootstrap

- Get a point est. of $\boldsymbol{\theta}$ using training data

$$\hat{\boldsymbol{\theta}} = \pi(\mathcal{D})$$

- Generate multiple datasets using $\hat{\boldsymbol{\theta}}$ as

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\hat{\boldsymbol{\theta}}) : n = 1 : N\} \quad (s = 1, 2, \ldots, S)$$

- Now compute the approximation as

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\tilde{\mathcal{D}}^{(s)}))$$

Nonparametric Bootstrap

- Use sampling with replacement on original training set to generate $\boldsymbol{S}$ datasets with $\boldsymbol{N}$ datapoints in each

Each dataset will contain roughly 63% unique datapoints from original training set

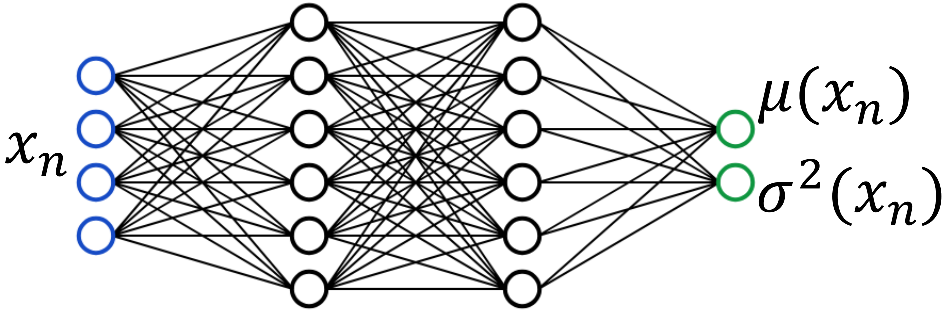
- Now compute the approximation as

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\tilde{\mathcal{D}}^{(s)}))$$

Estimating Model Uncertainty by Training a Single Model

Model Uncertainty by Training a Single Model

- Consider a regression model $p(y_n|x_n, w) = \mathcal{N}(y|\mu(x_n), \sigma^2(x_n))$

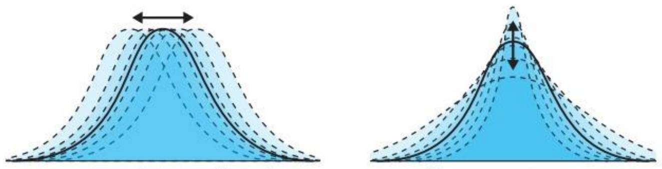


$$\log p(y_n|x_n, w) = \log \frac{1}{\sqrt{2\pi\sigma^2(x_n)}} \exp\left(-\frac{(y_n - \mu(x_n))^2}{2\sigma^2(x_n)}\right)$$

- This model defines the variance in outputs but there no is model uncertainty
- Can do MLE/MAP for $\mu(.)$ and $\sigma^2(.)$ by defining them as functions of the input, e.g.,
  - $\mu(x) = w_1^\top x$ and $\sigma^2(x) = \exp(w_2^\top x)$
  - $\mu(x) = \text{NN}(x, w_1)$ and $\sigma^2(x) = \exp(\text{NN}(x, w_2))$
- Typical ways to compute model uncertainty model uncertainty
  - Do Bayesian inference for the parameters (the network weights w_1, w_2)
  - Train an ensemble of models
- How to get the model uncertainty by training a single model?

Model Uncertainty by Training a Single Model

- Let's not treat $\boldsymbol{\mu(x)}$ and $\boldsymbol{\sigma^2(x)}$ as deterministic but random variables and use a model that estimates the parameters $\boldsymbol{m(x)}$ of their (joint) distribution

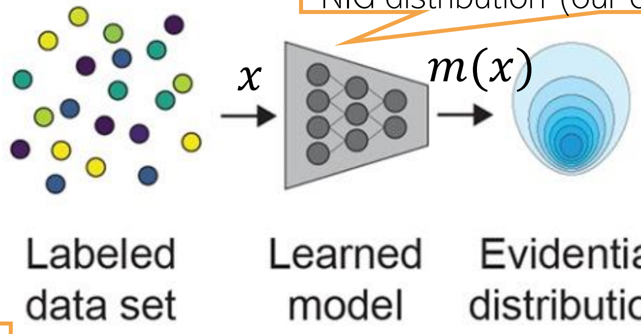


$\mu \sim \mathcal{N}(\gamma, \sigma^2/\lambda)$ $\qquad$ $\sigma^2 \sim \Gamma^{-1}(\alpha, \beta)$

$$p(\underbrace{\mu, \sigma^2}_{\boldsymbol{\theta}} | \underbrace{\gamma, \upsilon, \alpha, \beta}_{\boldsymbol{m}}) = \frac{\beta^\alpha \sqrt{\upsilon}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta + \upsilon(\gamma-\mu)^2}{2\sigma^2}\right\}$$

Normal Inverse-Gamma (NIG) distribution

Network predicts the parameters $\boldsymbol{m}$ of the NIG distribution (our evidential distribution)



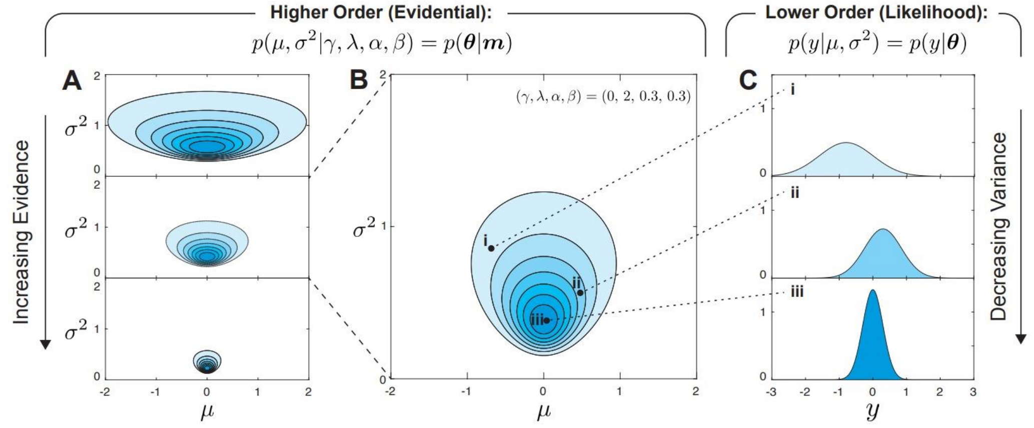
"evidential learning"

Marginal likelihood or predictive distribution

$$p(y|\mathbf{m}) = \int_{\sigma^2=0}^{\sigma^2=\infty} \int_{\mu=-\infty}^{\mu=\infty} p(y|\mu, \sigma^2) p(\mu, \sigma^2|\mathbf{m}) d\mu d\sigma^2$$

$$= \mathrm{t}(y; \gamma, \frac{\beta(1+\nu)}{\nu\alpha}; 2\alpha)$$

Maximize the log-marginal likelihood to estimate evidential distribution params $\boldsymbol{m}$



Mean prediction

(Expected) aleatoric (data) uncertainty

Epistemic (model) uncertainty

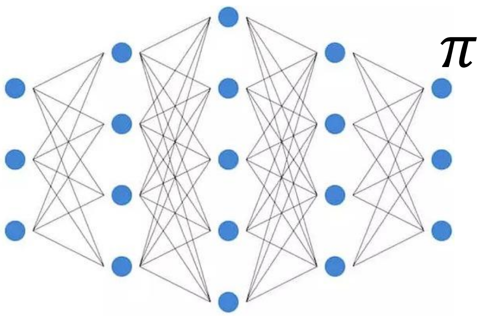
$$\mathbb{E}[\mu] = \gamma, \quad \mathbb{E}[\sigma^2] = \frac{\beta}{\alpha - 1}, \quad Var[\mu] = \frac{\beta}{\nu(\alpha-1)}$$

Fig source: Deep Evidential Regression (Amini et al, 2021), Evidential Deep Learning for Guided Molecular Property Prediction and Discovery (Soleimani et al, 2021)

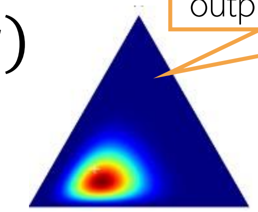
Model Uncertainty by Training a Single Model

- Consider K class classification: $p(y|x, \boldsymbol{W}) = \text{multinoulli}(y|\pi_1, \pi_2, \dots, \pi_K)$
- Assume distributions parameters $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_K]$ to be functions of the input, e.g.,
  - $\boldsymbol{\pi}(\boldsymbol{x}) = \text{softmax}(\boldsymbol{W}x)$
  - $\boldsymbol{\pi}(\boldsymbol{x}) = \text{softmax}(\text{NN}(x, \boldsymbol{W}))$

x



$\pi(x)$



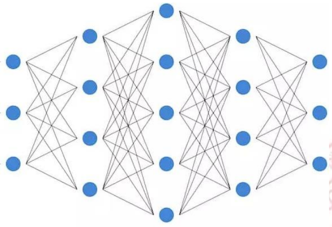
Dirichlet distribution over the output probability vector $\pi(x)$

In evidential learning, the network won't compute $\pi(x)$ but will give us a distribution over $\pi(x)$ by computing the Dirichlet distribution's concentration parameters $\alpha(x)$

- We can assume a distribution over $\boldsymbol{\pi}$ and learn params of this distribution*

$$p(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \text{Dirichlet}(\boldsymbol{\pi}|\boldsymbol{\alpha})$$

x



$\alpha(x)$

- We can maximize the marginal likelihood to estimate $\boldsymbol{\alpha}$

The neural network will be trained to compute $\alpha(x)$ for any given input $\boldsymbol{x}$

Maximize the marginal likelihood to estimate $\boldsymbol{\alpha}$

$$p(y|\alpha) = \int p(y|\pi)p(\pi|\alpha)d\pi$$

*Evidential Deep Learning to Quantify Classification Uncertainty (Sensoy et al, 2018)