**CSE6243: Advanced Machine Learning** **Fall 2023**

# Lecture 14: Decision-Focused Learning

*Lecturer: Lingkai Kong* *Scribes: Ronit Arora, Sumanth Kondapalli*

**Note**: *LaTeX template courtesy of UC Berkeley EECS Department.*



## 14.1 New Content

### 14.1.1 Deep Neural Networks

Extremely popular and widespread use due to ability to make predictions. Can be utilized for:

- Asset Allocation
- Vaccine Distribution
- Image Recognition
- Text/Speech/Audio Processing

The power of DNNs lie in their ability to make predictions and decisions. However, they largely exist as black-boxes, and are **over-confident**. We seek to show how uncertainty can be incorporated into these models.

### 14.1.2 Uncertainty

Uncertainty needs to be considered because DNNs are capable of making prediction errors which may errors may have significant impact in risk-sensitive domains. Having a corresponding uncertainty value with a decision allows for more thoughtful consideration of low-uncertainty scenarios in risk-sensitive domains.

Types of uncertainty:

- Epistemic Uncertainty - Ignorance about model caused by lack of observations
  - AKA model uncertainty
  - Insufficient data; can be reduced with **more data**
- Aleatoric Uncertainty - Natural randomness inherent in any task
  - AKA Data uncertainty
  - Overlap between classes, lack of features
  - Can be reduced with **additional features**

To calculate:

- Predictive Uncertainty = $H[\sum_{m=1}^{M} \frac{1}{M} p(y|x^*, \theta_m)]$ ← Entropy of averaged predictions
- Aleatoric Uncertainty = $\frac{1}{M} \sum_{m=1}^{M} H[p(y|x^*, \theta_m)]$ ← Average of predictive entropy
- Epistemic Uncertainty = Predictive Uncertainty − Aleatoric Uncertainty

### 14.1.3 Neural Stochastic Differential Equation (SDE-Net)

Ordinary determistic equation follows the form:

$$x_{t+1} = x_t + f(x_t, t) \rightarrow dx_t = f(x_t, t)dt$$

Neural Stochastic Differential Equation (SDE-Net) adds Brownian motion

$$dx_t = f(x_t, t)dt + g(x_t, t)dW_t$$

The resultant model components are:

- A drift net $f$ for predictive accuracy and aleatoric uncertainty
- A diffusion net $g$ for epistemic uncertainty

Some issues with:

- DNN
  - Lower performance
  - A low predictive loss is not indicative of quality decisions
- DFL
  - Restricted to convex objectives due to KKT conditions
  - Training is inefficient as differantiation needs to occur at every iteration

Instead, we use Stochastic Optimization with Energy-Based Model (SO-EBM). We do this because we can:

- Parematrize non-convex objectives
- Eliminate need to differentiate at every iteration

$$q(a|x;\theta) = \frac{\exp(-\mathbb{E}(x, a; \theta)}{Z(x, \theta)}$$

$$Z(x;\theta) = \int \exp(-\mathbb{E}(x, a; \theta))da$$

$$\mathbb{E}(x;\theta) : \mathbb{R}^D \rightarrow \mathbb{R}$$

Use energy-based parametrization to model probability distribution of decisions based on features. Furthermore, we can parametrize the energy function with expected task loss:

$$\mathbb{E}(x, a; \theta) = \mathbb{E}_{p(y|x;\theta)}[f(y, a)]$$

### 14.1.4 SDE-Net Model Training

The gradient of the training loss (the formal loss function is defined in a subsequent section) can be represented by:

$$\frac{\delta \mathcal{L}_{\text{Total}}}{\delta\theta} = \mathbb{E}_{(x,a^*)\sim\mathcal{D}_a}\left(\frac{\delta\mathbb{E}(a^*,x;\theta)}{\delta\theta} - \mathbb{E}_{q(\tilde{a}|x;\theta)}\frac{\delta\mathbb{E}(\tilde{a},x;\theta)}{\delta\theta}\right) + \lambda\mathbb{E}_{(x,y)\sim\mathcal{D}}\left(\mathbb{E}_{p(\hat{a}|y)}\frac{\delta\mathbb{E}(\hat{a},x;\theta)}{\delta\theta} - \mathbb{E}_{q(\tilde{a}|x;\theta)}\frac{\delta\mathbb{E}(\tilde{a},x;\theta)}{\delta\theta}\right)$$

This allows for the gradient to be estimated by sampling the distribution of the model. No need for differentiation at each iteration.

Breaking down this loss and the gradient of it, our loss is essentially a combination of a MLE with KL Divergence. This is devised for 2 reasons:

1. Finding Optimum Location - With MLE, by definition we can find an optimal point for the model. This is provided in the derived first term of the gradient.
2. We qualify this with KL divergence, as we are modeling the probability distribution as part of the exponential family. Thus, to ensure the loss is representative of this distribution overall (distribution regularization), we employ KL Divergence. This is provided in the derived second term of the gradient.

We can also use a self-normalized importance sampler. We still sample from:

$$q(a|x;\theta) = \frac{\exp(-\mathbb{E}(x,a;\theta))}{Z(x;\theta)}$$

The gradient above is large and intractable to calculate (similar to other energy based model losses) with a $q$ that has such a large search space for $a$. Therefore, we estimate the gradient by sampling from the above distribution.

In essence, we sample a set of $M$ particle locations, each $a$ from some proposal distribution. This proposal distribution is composed of a mixture of Gaussians:

$$\pi(a|x) = \frac{1}{K}\sum_{i=1}^{K}\mathcal{N}(a^*;\sigma_k)$$

After doing this, we can represent $q(a|x;\theta)$ as a weighted distribution with weights determined as follows:

$$w^m = \frac{\frac{\exp(-\mathbb{E}(a^m|x;\theta))}{\pi(a^m|x)}}{\sum_{m=1}^{M}\frac{\exp(-\mathbb{E}(a^m|x;\theta))}{\pi(a^m|x)}}$$

The benefit of this is that we can sample extremely quickly from a mixture of Gaussians, and make the gradient calculation feasible through estimation from sampling.

### 14.1.5 Distribution-Free Training Objective

Since the Brownian motion term $g(x,a) = \mathbb{E}_{p(y|x)}[f(y,a)]$ is a function of $x$ and $a$, similar to $f$, it should be possible to train without using the distribution as was shown previously. Our objective function becomes:

$$g^*(x,a) = \text{argmin}_g\mathbb{E}_a\mathbb{E}_{(x,y)\sim\mathcal{D}}[g(x,a) - f(y,a)]^2$$

To calculate MSE, we use:

$$\text{MSE}_{\text{test}} = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(x,a) - \mathbb{E}_{p(y|x)}[f(y,a)]\right)^2\right]}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(x,a) - \mathbb{E}_{\mathcal{D}}[g^*_{\mathcal{D}}(x,a)]\right)^2\right]}_{\text{Variance}}$$

Worth noting that the variance term can be reduced with more data, whereas reducing the bias term is less straightforward. Briefly, the proposition to reduce this bias term for a lower MSE is through using attention. The reason for this is that we inherently seek parameterization restrictions of $p(y|x)$. This technique is derived from conditional mean embedding. Given $\hat{p}_{\mathbf{R}}(y|x)$ which represents our parameterization restriction, we can represent: $g(x,a) = \mathbb{E}_{\hat{p}_{\mathbf{R}}}[f(y,a)]$, which is essentially our attention mechanism (sum of functions with real weight factors applied, equal to our expectation).

### 14.1.6 SDE-Net Loss Function

Objective function for training SDE-Net

$$\underbrace{\min_{\theta_f}\mathbb{E}_{x_0\sim P_{\text{train}}}\mathbb{E}(L(x_T))}_{\text{Task-dependent loss function}} + \underbrace{\min_{\theta_g}\mathbb{E}_{x_0\sim P_{\text{train}}}g(x_0;\theta_g)}_{\text{Minimize diffusion for in-distribution}} + \underbrace{\max_{\theta_g}\mathbb{E}_{x_0\sim P_{\text{OOD}}}g(x_0;\theta_g)}_{\text{Maximize diffusoin for out-of-distribution}}$$

$$s.t. \qquad dx_t = f(x_t,t;\theta_f)dt + g(x_0;\theta_g)dW_t$$

We can simulate out-of-distribution data by adding noise in the form: $\tilde{x}_0 = x_0 + \epsilon$.