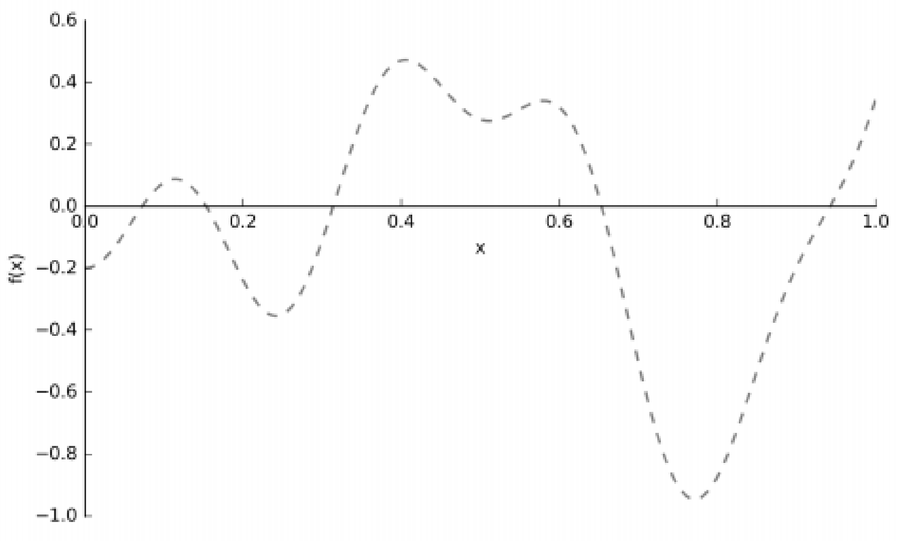
Bayesian Optimization, Linear Generative Models (FA/PPCA)

CS772A: Probabilistic Machine Learning

Piyush Rai

Bayesian Optimization: The Basic Formulation

- Consider finding the optima x_* (say minima) of a function $f(x)$



- Caveat: We don't know the form of the function; can't get its gradient, Hessian, etc
- Can only query the function's values at certain points (i.e., only "black-box" access)
  - The values may or may not be noisy (i.e., we may be given $f(x)$ or $f(x) + \epsilon$)

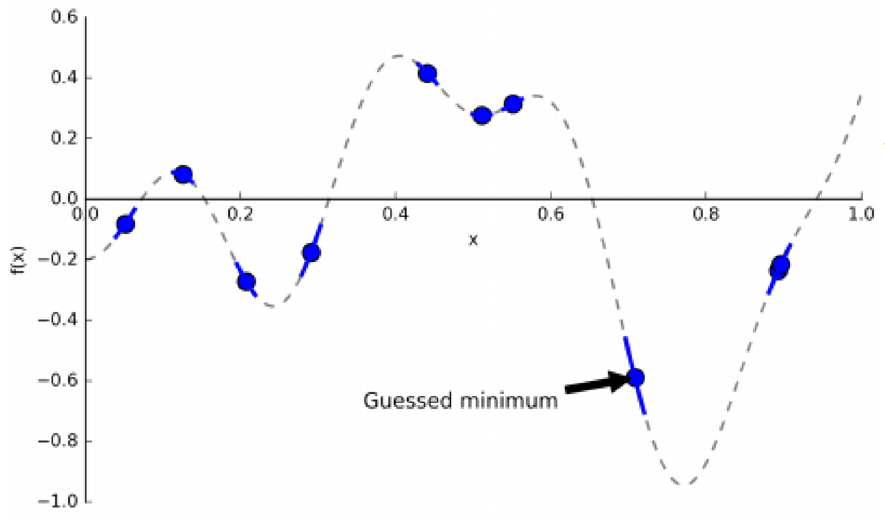
Bayesian Optimization: Some Applications

- Drug Design: Want to find the optimal chemical composition for a drug
  - Optimal composition will be the one that has the best efficacy
  - But we don’t know the efficacy function
  - Can only know the efficacy via doing clinical trials
  - Each trial is expensive; can’t do too many trials
- Hyperparameter Optimization: Want to find the optimal hyperparameters for a model
  - Optimal hyperparam values will be those that give the lowest test error
  - Don’t know the true “test error” function
  - Need to train the model each time with different h.p. values and compute test error
  - Training every time will be expensive (e.g., for deep nets)
  - Note: Hyperparams here can even refer to the structure of a deep net (depth, width, etc)
- Many other applications: Website design via A/B testing, material design, optimizing physics based models (e.g., aircraft design), etc

Bayesian Optimization

- Can use BO to find maxima or minima

- Would like to locate the optima by querying the function's values (say, from an oracle)



Assuming the goal is to find the minima

- We would like to do so using as few queries as possible

- Reason: The function's evaluation may be time-consuming or costly

Bayesian Optimization

Note: Function values can be noisy too, e.g., $f(x_n) + \epsilon_n$

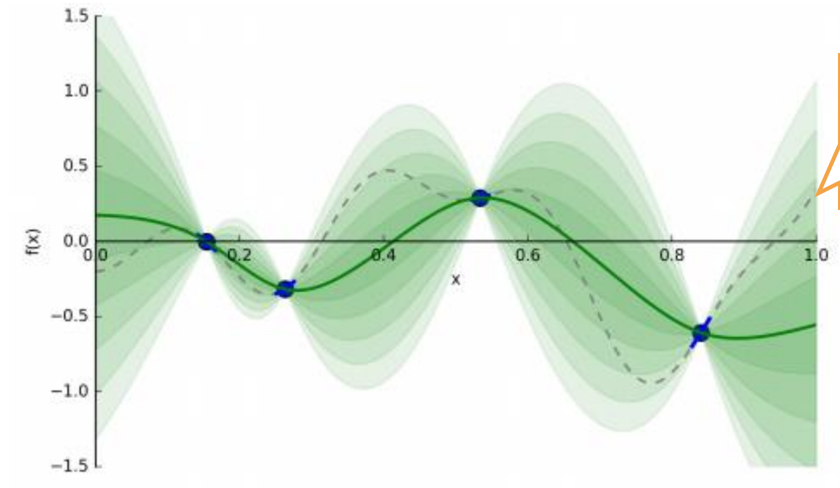
- Suppose we are allowed to make the queries sequentially

$\{(x_n, f(x_n))\}_{n=1}^N$

- This information will be available to us in form of query-function value pairs

By solving a regression problem

- Queries so far can help us estimate the function



Dotted curve: True function
Green curve: Current estimate ("surrogate") of the function
Shaded region: Uncertainty in the function's estimate

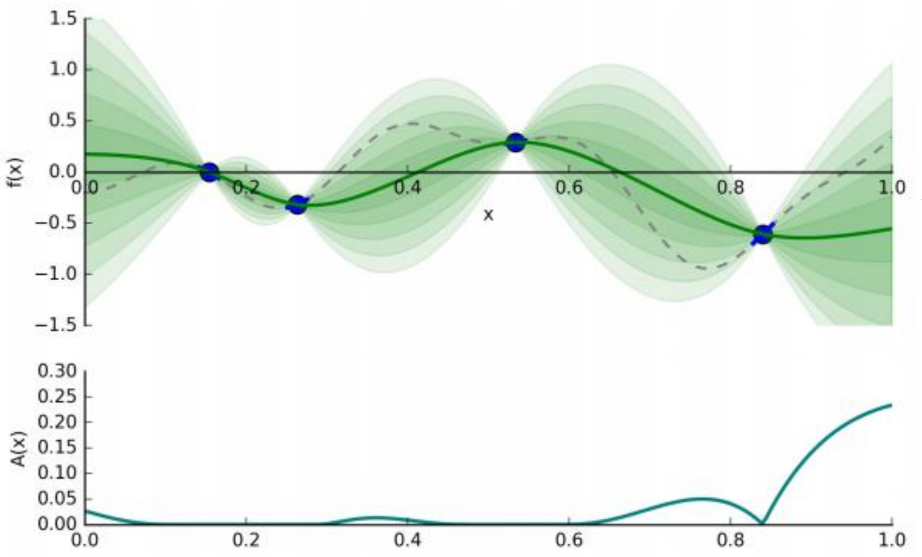
- BO uses past queries + function's estimate+uncertainty to decide where to query next
- Similar to Active Learning but the goal is to learn f as well as finds its optima

Bayesian Optimization

Note: Function values can be noisy too, e.g., $f(x_n) + \epsilon_n$

- BO requires two ingredients
  - A regression model to learn a surrogate of $f(x)$ given previous queries $\{(x_n, f(x_n))\}_{n=1}^N$
  - An acquisition function $A(x)$ to tell us where to query next

Assumption: $A(x)$ should be easier to optimize than $f(x)$



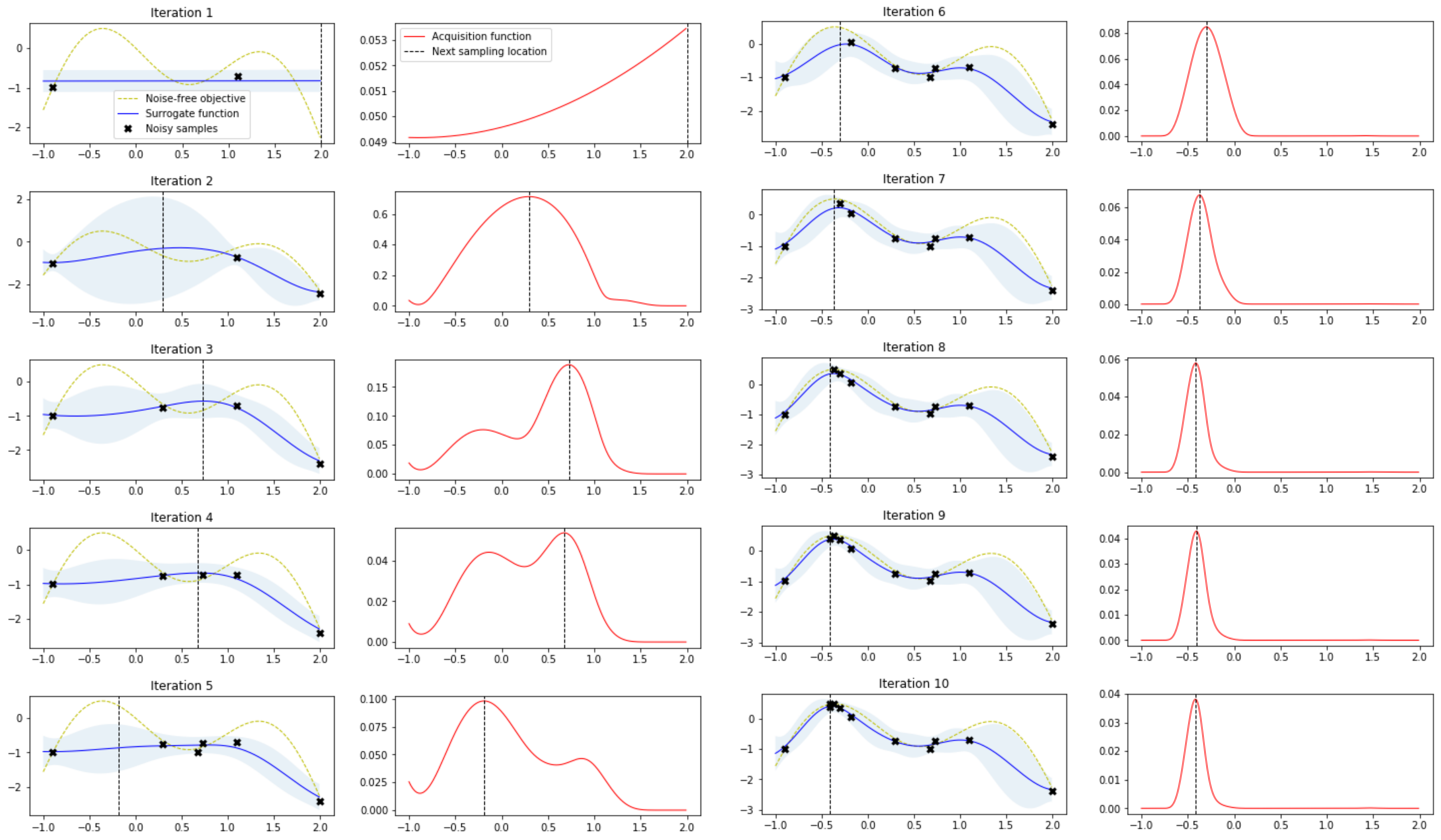
Dotted curve: True function
Green curve: Current surrogate of the function
Shaded region: Uncertainty in the function's estimate

A typical example of what $A(x)$ might look like, assuming that the goal is to find the maxima of $f(x)$

- Note: The regression model must also have estimate of function's uncertainty
  - Bayesian nonlinear regression, such as GP, Bayesian Neural network, etc would be ideal

Bayesian Optimization: An Illustration

- Suppose our goal is to find the <u>maxima</u> of $f(x)$ using BO



Pic source: http://krasserm.github.io/2018/03/21/bayesian-optimization/

Some Basic Acquisition Functions for BO

(assuming we are finding the minima)

Acquisition Functions: Probability of Improvement

- Assume past queries $\mathcal{D}_N = (\boldsymbol{X}, \boldsymbol{f}) = \left(x_n, f(x_n)\right)_{n=1}^{N}$ and suppose $f_{min} = \min \boldsymbol{f}$
- Suppose f_{new} denotes the function's value at the next query point x_{new}
- We have an <u>improvement</u> if $f_{new} < f_{min}$ (recall we are doing minimization)
- Assuming the function is real-valued, suppose the posterior predictive for x_{new} is

$$p(f_{new}|x_{new}, \mathcal{D}_N) = \mathcal{N}(f_{new}|\mu(x_{new}), \sigma^2(x_{new}))$$

- We can define a probability of improvement based acquisition function

Exercise: Verify

$$A_{PI}(x_{new}) = p(f_{new} \leq f_{min}) = \int_{-\infty}^{f_{min}} \mathcal{N}(f_{new}|\mu(x_{new}), \sigma^2(x_{new}))df_{new} = \Phi\left(\frac{f_{min} - \mu(x_{new})}{\sigma(x_{new})}\right)$$

$\Phi()$ denotes CDF of $\mathcal{N}(0,1)$

- The optimal query point will be one that maximizes $A_{PI}(x_{new})$

$$x_* = \text{argmax}_{x_{new}} A_{PI}(x_{new})$$

Acquisition Functions: Expected Improvement

- PI doesn't take into account the amount of improvement
- Expected Improvement (EI) takes this into account and is defined as

$$A_{EI}(x_{new}) = \mathbb{E}[f_{min} - f_{new}] = \int_{-\infty}^{f_{min}} (f_{min} - f_{new})\mathcal{N}(f_{new}|\mu(x_{new}), \sigma^2(x_{new}))df_{new}$$

Exercise: Prove this result

$$= (f_{min} - \mu(x_{new}))\Phi\left(\frac{f_{min} - \mu(x_{new})}{\sigma(x_{new})}\right) + \sigma(x_{new})\mathcal{N}\left(\frac{f_{min} - \mu(x_{new})}{\sigma(x_{new})}; 0,1\right)$$

- The optimal query point will be one that maximizes $A_{EI}(x_{new})$

$$x_* = \text{argmax}_{x_{new}} A_{EI}(x_{new})$$

Focus on points where the function has small values (since we are looking for its minima)

Focus on points where the function has high uncertainty (so that including them improves our estimate of the function)

- Note that the above acquisition function trades off exploitation vs exploration
  - Will prefer points with small predictive mean $\mu(x_{new})$: Exploitation
  - Will prefer points with large predictive variance $\sigma(x_{new})$: Exploration

Acquisition Functions: Lower Confidence Bound

- Lower Confidence Bound (LCB) also takes into account exploitation vs exploration
- Used when the regression model is a Gaussian Process (GP)
- Assume the posterior predictive for a new point to be

$$p(f_{new}|x_{new}, \mathcal{D}_N) = \mathcal{N}(f_{new}|\mu(x_{new}), \sigma^2(x_{new}))$$

When using BO for <u>maximization</u>, we use Upper Confidence Bound (UCB) defined as $A_{UCB}(x_{new}) = \mu(x_{new}) + \kappa\, \sigma(x_{new})$ and $x_* = \text{argmax}_{x_{new}} A_{UCB}(x_{new})$

- The LCB based acquisition function is defined as

$$A_{LCB}(x_{new}) = \mu(x_{new}) - \kappa\, \sigma(x_{new})$$

- Point with the smallest LCB is selected as the next query point

$$x_* = \text{argmin}_{x_{new}} A_{LCB}(x_{new})$$

Thus prefer points at which the function has low mean but high variance

- κ is a parameter to trade-off exploitation (low mean) and exploration (high variance)
- Under certain conditions, the iterative application of this acquisition function will converge to the true global optima of f (Srinivas et al. 2010)

Bayesian Optimization: The Overall Algo

- Initialize $\mathcal{D} = \{\}$
- For $n = 1,2,\ldots,N$ (or until the budget doesn't exhaust)
  - Select the next query point x_n by optimizing the acquisition function

$$x_n = \text{argopt}_x\, A(x)$$

  - Get function's value from the black-box oracle: $f_n = f(x_n)$
  - $\mathcal{D} = \{\mathcal{D} \cup (x_n, f_n)\}$

Can get the function's minima from this set of function's values

  - Update the regression model for f using data $\mathcal{D}$

BO: Some Challenges/Open Problems

- Learning the regression model for the function
  - GPs are flexible but can be expensive as N grows
  - Bayesian neural networks can be an more efficient alternative to GPs (Snoek et al, 2015)
  - Hyperparams of the regression model itself (e.g., GP cov. function, Bayesian NN hyperparam)
- High-dimensional Bayesian Optimization (optimizing functions of many variables)
  - Most existing methods work well only for a moderate-dimensional $\boldsymbol{x}$
  - Number of function evaluations required would be quite large in high dimensions
  - Lot of recent work on this (e.g., based on dimensionality reduction)
- Multitask Bayesian Optimization (joint BO for several related functions)
  - Basic idea: If two functions are similar their optima would also be nearby

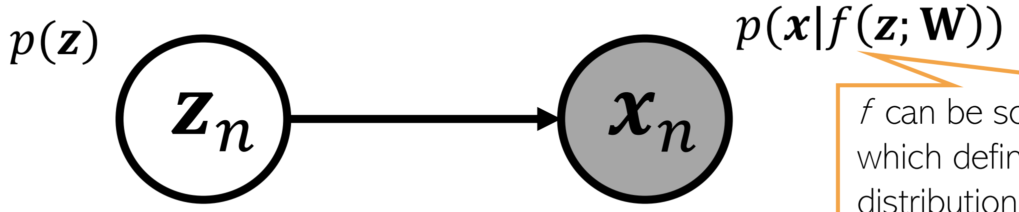
Latent Variable Models for Data Generation

Main goal is to generate data but can also be use to learn low-dimensional features (z_n)

$$\begin{tikzcd}\boldsymbol{z}_n \arrow[r] & \boldsymbol{x}_n\end{tikzcd}$$

Latent Variable Models for Generation Tasks

- Assume a K-dim latent variable $\boldsymbol{z}_n$ is transformed to generate to D-dim observation $\boldsymbol{x}_n$



f can be some linear or nonlinear model which defines the parameters of the distribution of $p(x|z)$ and W denotes the parameters of this model

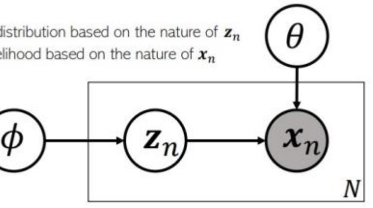
Also possible to use a GP to model f

- It is common to use a Gaussian prior for $\boldsymbol{z}_n$ (though other priors can be used)
- If we use a neural net or GP, such models can generate very high-quality data
  - Take the trained network, generate a random $\boldsymbol{z}$ from prior, pass it through the model to generate $\boldsymbol{x}$

Some sample images generated by Vector Quantized Variational Auto-Encoder (VQ-VAE), a state-of-the-art latent variable model for generation

Factor Analysis and Probabilistic PCA

$p(\boldsymbol{z}_n|\phi)$: A suitable prior distribution based on the nature of $\boldsymbol{z}_n$
$p(\boldsymbol{x}_n|\boldsymbol{z}_n,\theta)$: A suitable likelihood based on the nature of $\boldsymbol{x}_n$



- FA and PPCA assume $\boldsymbol{f}$ to be a linear model
- In FA/PPCA, latent variables $\boldsymbol{z}_n \in \mathbb{R}^K$ typically assumed to have a Gaussian prior
  - If we want sparse latent variabled, can use Laplace or spike-and-slab prior on $\boldsymbol{z}_n$
  - More complex extensions of FA/PPCA use a mixture of Gaussians prior on $\boldsymbol{z}_n$
- Assumption: Observations $\boldsymbol{x}_n \in \mathbb{R}^D$ typically assumed to have a Gaussian likelihood
  - Other likelihood models (e.g., exp-family) can also be used if data not real-valued
- Relationship between $\boldsymbol{z}_n$ and $\boldsymbol{x}_n$ modeled by a noisy linear mapping

$$\boldsymbol{x}_n = \boldsymbol{W}\boldsymbol{z}_n + \epsilon_n = \sum_{k=1}^{K} \boldsymbol{w}_k z_{nk} + \epsilon_n$$

Zero-mean and diagonal or spherical Gaussian noise

Linear combination of the columns of $\boldsymbol{W}$

$$p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|0,\mathbf{I})$$
$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\boldsymbol{x}_n|\mathbf{W}\mathbf{z}_n,\Psi)$$

Diagonal for FA, spherical for PPCA

- Linear Gaussian Model. $\boldsymbol{W}$, $\boldsymbol{z}_n$'s, and Ψ can be learned (e.g, using EM, VI, MCMC)

Some Variants of FA/PPCA

Non-negative priors often give a nice interpretability to such latent variable models (will see some more examples of such models shortly)

- Gamma-Poisson latent factor model

Popular for modeling count-valued data (in text analysis, recommender systems, etc)

  - Assumes K-dim non-negative latent variable $\mathbf{z_n}$ and D-dim count-valued observations $\mathbf{x_n}$
  - An example: Each $\mathbf{x_n}$ is the word-count vector representing a document

$$p(\mathbf{z_n}) = \prod_{k=1}^{K} \text{Gamma}(\mathrm{z_{nk}}|\mathrm{a_k}, \mathrm{b_k})$$
$$p(\mathbf{x_n}|\mathbf{z_n}) = \prod_{d=1}^{D} \text{Poisson}(\mathrm{x_{nd}}|f(\mathbf{w_d}, \mathbf{z_n}))$$

This is the rate of the Poisson. It should be non-negative, $\exp(\mathbf{w}_d^\top \boldsymbol{z}_n)$, or simply $\mathbf{w}_d^\top \boldsymbol{z}_n$ if $\boldsymbol{w}_d$ is also non-negative (e.g., using a gamma/Dirichlet prior on it)

  - This can be thought of as a probabilistic non-negative matrix factorization model

- Dirichlet-Multinomial/Multinoulli PCA
  - Assumes K-dim non-negative latent variable $\mathbf{z_n}$ and D categorical obs $\mathbf{x_n} = \{\boldsymbol{x}_{nd}\}_{d=1}^{D}$
  - An example: Each $\mathbf{x_n}$ is a document with D words in it (each word is a categorical value)

Also sums to 1

$$p(\mathbf{z_n}) = \text{Dirichlet}(\mathbf{z_n}|\boldsymbol{\alpha})$$
$$p(\mathbf{x_n}|\mathbf{z_n}) = \prod_{d=1}^{D} \text{Multinoulli}(\mathrm{x_{nd}}|f(\mathbf{w_d}, \mathbf{z_n}))$$

This should give the probability vector of the multinoulli over x_{nd}. It should be non-negative and should sums to 1