# Poisson Regularization in Neural Networks using the Gumbel-Softmax/Concrete Distribution

Shamak Dutta
*University of Waterloo*

**Abstract**

Regularization constrains the parameters of a model in a certain way to reflect our prior knowledge about them. In the context of neural networks, this can be implemented in the form of L1/L2 weight penalties, dropout or batch-normalization. This project proposes Poisson regularization for neural networks, where the units' values determine the rate of an approximate Poisson distribution and samples from this distribution are propagated forward. However, sampling from discrete distributions poses a problem in back-propagation because the process is non-differentiable and stochastic. To solve this issue, the continuous relaxation of a discrete categorical distribution known as the Gumbel-Softmax/Concrete distribution is used to approximate the Poisson distribution. The model is evaluated on the classification of the MNIST dataset.

## 1. Problem Formulation

The goal of this project is to incorporate Poisson noise into neural networks as a regularization technique. This is not as straight-forward as dropout or additive Gaussian noise, because sampling from a Poisson is a non-differentiable procedure and this breaks the back-propagation algorithm which is the standard algorithm to update the parameters of a neural network given an objective function. The parameter updates are guided by the gradient calculation of the objective function with respect to every parameter, which clearly cannot be used when the sampling procedure from a Poisson distribution is used.

*Email address:* `s7dutta@edu.uwaterloo.ca` (Shamak Dutta)

## 2. Problem Solution

*2.0.1. Objective Function*

The objective function used for the classification task is the cross entropy loss,

$$L(\theta) = \frac{-1}{N} \sum_{j}^{N} \sum_{i=1}^{k} y_{ji} log(p_{ji}), \tag{1}$$

where $N$ is the number of examples, $k$ is the number of object categories, $y_{ji}$ is the binary label for the $j^{th}$ example whether it belongs to class $i$ or not and $p_{ji}$ is the predicted label for class $i$ on example $j$.

*2.0.2. Poisson Noise*

The ideal way to add Poisson noise to a layer in a neural network is as follows:

1. Compute the layer's activations as usual. This would be a linear transformation followed by a non-linear function.
2. The activations serve as the rates of a Poisson distribution. Samples from this distribution are propagated forward.

$$f(X) \sim \text{Poisson}(\lambda = X) \tag{2}$$

3. However, since $f(X)$ is not differentiable, during back-propagation, the gradients of $f(X)$ with respect to $X$ do not exist.

To overcome this problem, the Poisson distribution of a given rate is converted to a discrete k-categorical distribution by choosing an appropriate value of k. Once we have an approximate discrete categorical approximation of the Poisson distribution (which is equal to the Poisson distribution as k tends to infinity), we can use the continuous relaxation of a discrete categorical distribution as discovered in [2, 4]. This allows for a differentiable sample from an approximate Poisson distribution and thus back-propagation can continue as normal.

The entire procedure to sample from an approximate Poisson distribution is as follows:

1. Given a rate $\lambda \in \mathbb{R}$, a k-categorical distribution is constructed as:

$$z = [\pi_1 = P(X=0;\lambda), \pi_2 = P(X=1;\lambda), ..., \pi_k = P(X=k;\lambda)] \in \mathbb{R}^{\mathrm{k}}. \quad (3)$$

   This is a crude approximation as we are using the probability mass function of the Poisson distribution of rate $\lambda$ to determine the individual probabilities of a k-categorical distribution. The ideal method would be to determine the pmf of the truncated Poisson distribution $[P(X=0|X \le k;\lambda), P(X=1|X \le k;\lambda), ..., P(X=k|X \le k;\lambda)]$. However, this was not investigated as part of this project.

2. Using the Gumbel-Softmax distribution and a temperature parameter $\tau$, a one-hot sample $y \in \mathbb{R}^{\mathrm{k}}$ (a sample is a k-dimensional one-hot vector) from the categorical distribution $z$ can be generated as follows:

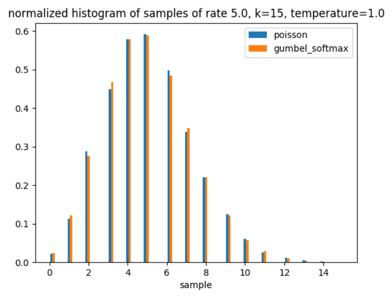
   (a) $y_i = softmax(\frac{g_i + log\pi_i}{\tau})$, where $g_1..g_k$ are i.i.d samples drawn from a Gumbel(0,1) (The Gumbel(0, 1) distribution can be sampled using inverse transform sampling by drawing $u \sim Uniform(0,1)$ and computing $g = log(log(u)))$.

   (b) The straight-through Gumbel-Softmax estimator is used to discretize $y$ using $argmax$ on the forward pass, but use the continuous $softmax$ approximation on the backward pass. This allows for discrete values to propagate through the network but use the continuous relaxation to compute gradients. Thus, $y_i = argmax[softmax(\frac{g_i + log\pi_i}{\tau})]$, which is an approximate sample from a Poisson distribution.

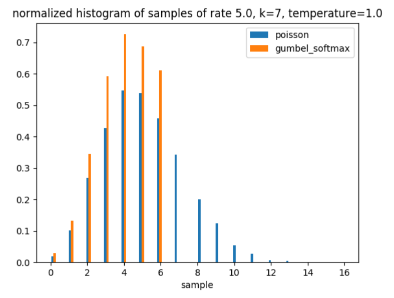
### *2.0.3. Quality of Approximations*

The comparison between the normalized histograms of samples from a true Poisson distribution and the Gumbel-Softmax approximation is shown for different values of $\lambda$.

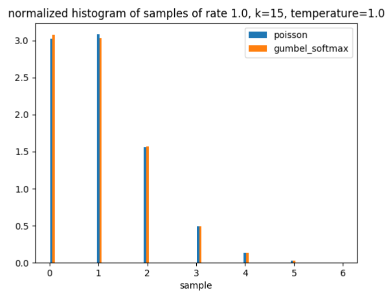
As shown in Figure 1, the approximation using Gumbel-Softmax comes quite close to the true Poisson distribution for the right choice in parameters. It is seen in Figure 1b where the rate is 5 and the k value is 7, the densities look quite different. This emphasizes that the value of k should be chosen wisely. In practice, the value of k is chosen online during training where it is set to the maximum value of the mini-batch plus 4 times the square root of the maximum value. It is better to keep a conservative estimate of k with rather than an exact value.



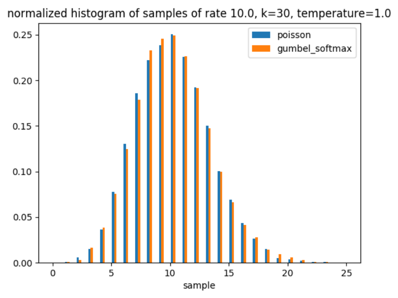
(a)



(b)



(c)



(d)

Figure 1: Normalized histograms of samples from a Poisson distribution and the Gumbel-Softmax approximation

## 3. Data Sources

Training is performed on $28 \times 28$ MNIST [3] images with no additional data augmentation/deformation. The dataset has 60K and 10K images for training and testing respectively. 5K images out of the training set are used for validation during training to ensure minimal overfitting to the training set.

## 4. Solution

*4.0.1. Network Architecture and Setup*

The baseline model is standard convolutional neural network with two convolutional layers (including max-pooling of stride 2) of 32 and 64 filters followed by a fully connected layer of 1024 units and finally the output softmax layer of 10 units which is the final prediction. Each layer is followed by a ReLu non-linearity. Dropout is added to the first fully connected layer with a rate of 0.5. The Poisson noise is introduced in the first convolutional layer (abbreviated as P_1) and in both first and second convolutional layers (abbreviated as P_2) to see if affects the classification or learning performance of the network. To provide another point of comparison, a Poisson noise model with straight-through gradient estimation was evaluated. In this model, the samples on the forward pass are the true samples from the Poisson distribution with the rate equal to the unit values. However, on the backward pass, the sampling function is treated as the identity function and the gradient is set to be the expected value of the distribution, which in this case is the unit's value itself. This is somewhat reminiscent of the Straight-Through Estimator introduced in [1]. This model is abbreviated as P_STE and the noise is added to both convolutional layers.

The TensorFlow framework was used to develop the models while the optimizer used is stochastic gradient descent with momentum. Stochastic gradient descent has trouble navigating through regions where the surface curves more deeply in one direction over another dimension (ravine). This makes SGD oscillate between the slopes of the 'ravine' instead of reaching the local minimum. Momentum is a change to the stochastic gradient update by adding a fraction $\gamma$ of the update vector of the last time step to the current vector,

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \tag{4}$$

$$\theta = \theta - v_t. \tag{5}$$

The mini-batch size used is 100 with a learning rate of 0.0001 for the gradient descent optimizer with $\gamma = 0.9$. The models are trained for 64000 steps, where each step is a weight update over a batch size of 100.

## 5. Results

The results of training the convolutional networks with the incorporation of Poisson noise is listed in Table 1. This is quite encouraging as the Poisson noise models have comparable performance to non-Poisson activity networks.

Table 1: Classification results on test set of MNIST

| Model | Classification Accuracy (%) |
|---|---|
| Baseline | 98.8 |
| P_1 | 98.8 |
| P_2 | 98.0 |
| P_STE | 98.2 |

## 6. Analysis and Conclusions

In this project, Poisson noise was incorporated into convolutional neural networks for the task of image classification. The activation values determine the rate of the Poisson distribution from which samples are propagated forward. There are many choices on how to compute the gradient on the backward pass during back-propagation. This project explored two options: straight-through gradient estimation and the continuous relaxation of a discrete distribution using the Gumbel-Softmax.

From the results in Section 5, it is noted that the performance of the Poisson noise models do not outperform the baseline, but achieve comparable accuracy. This is quite encouraging because the gradient calculations are coarse estimates of the sampling function, in both cases of Poisson noise. There is definitely more work to be investigated such as changing the temperature parameter $\tau$ and also training the network for a longer period because it might take longer to converge when more noise is added. It is also noted that neurons in the visual cortex exhibit Poisson-like statistics with a Fano factor of 1, which serves as the motivation for this project.

The drawbacks of this model are the computational cost of sampling from a Poisson distribution for the straight-through estimator model and the cost of converting to categorical distribution and gradient calculation for the Gumbel-Softmax approximation. It does remain to be investigated with more thorough experiments whether a model with Poisson noise can outperform the traditional convolutional neural networks.

## References

[1] Y. Bengio, N. Léonard, and A. C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[2] E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *ArXiv e-prints*, Nov. 2016.

[3] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[4] C. J. Maddison, A. Mnih, and Y. Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *ArXiv e-prints*, Nov. 2016.