Sampling the “Inverse Set” of a Neuron

Suryabhan Singh Hada    Miguel Á. Carreira-Perpiñán
{shada,mcarreira-perpinan}@ucmerced.edu

Electrical Engineering and Computer Science,
University of California, Merced
http://eecs.ucmerced.edu
Introduction

- Deep neural nets are accurate black-box models. They have shown much success in many applications such as computer vision and natural language processing.
- This makes it necessary to understand the internal working of these networks. What does a given neuron represent?
- We solve this by characterizing the region of input space that excites a given neuron to a certain level; we call this the inverse set.
- This inverse set is a complicated high dimensional object that we explore using an optimization-based sampling approach. Inspection of samples of this set by a human can reveal regularities that help to understand the neuron.
We say an input $x$ is in the inverse set of a given neuron having a real-valued activation function $f$ if it satisfies the following two properties:

$$z_1 \leq f(x) \leq z_2$$ \hspace{1cm} \text{$x$ is a valid input} \hspace{1cm} (1)$$

where $z_1, z_2 \in \mathbb{R}$ are activation values of the neuron.

For example, consider a linear model with weight vector ($w$), bias ($b$), logistic activation function $\sigma(w^T x + b)$ and all valid inputs to have pixel values between $[0,1]$. For $z_2 = 1$ (maximum activation value) and $0 < z_1 < z_2$, the inverse set will be the intersection of the half space $w^T x + c \geq \sigma^{-1}(z_1)$ and the $[0,1]$ hypercube.
For deep neural networks, we approximate the inverse set with a sample that covers it in a representative way.

A simple way to do this is to select all the images in the training set that satisfy eq. (1), but this may rule out all images.

Therefore, we need an efficient algorithm to sample the inverse set.
To create a sample $x_1, \ldots, x_n$ that covers the inverse set, we transform eq. (1) into a constrained optimization problem:

$$\arg \max_{x_1, x_2, \ldots, x_n} \sum_{i, j=1}^{n} \|x_i - x_j\|_2^2 \quad \text{s.t.} \quad z_1 \leq f(x_1), \ldots, f(x_n) \leq z_2.$$

The objective function ensures that the samples are different from each other and satisfy eq. (1).

It has two issues. The generated images are noisy and are very sensitive to small changes in their pixels.
We solve the issues in following way:

- To counter the noisy image issue, we use generator network $G$ to generate images from a code vector $c$.
- For the second issue, we compute distances on a low-dimensional encoding $E(G(c))$ of the generated images constructed by an encoder $E$.

Our final formulation for generating $n$ samples.

$$\arg\max_{c_1,c_2,\ldots,c_n} \sum_{i,j=1}^{n} \|E(G(c_i)) - E(G(c_j))\|_2^2$$

s.t. $z_1 \leq f(G(c_1)), \ldots, f(G(c_n)) \leq z_2$. 
Because of the quadratic complexity of the objective function over the number of samples \( n \), it is computationally expensive to generate many samples.

It involves optimizing all code vectors (\( c \)) together; for larger \( n \), it is not possible to fit all in the GPU memory.

Two approximation:
- Stop the optimization algorithm once the samples enter the feasible set, as, by that time, the samples are already separated.
- Create the samples incrementally, \( K \) samples at a time (with \( K \ll n \)).
Faster sampling approach

- Optimize the objective function for the first $K$ samples, initializing the code vectors $c$ with random values. We stop the optimization once the samples are in the feasible set. These samples are then fixed (called seeds $C_0$).

- The next $K$ samples are generated by the following equation:

$$\arg \max_{c_1, c_2, \ldots, c_K} \sum_{i,j=1}^{K} \left\| E(G(c_i)) - E(G(c_j)) \right\|_2^2 + \sum_{i=1}^{K} \sum_{y=1}^{\left| C_0 \right|} \left\| E(G(c_i)) - E(G(c_y)) \right\|_2^2$$

s.t. $z_1 \leq f(G(c_1)), \ldots, f(G(c_K)) \leq z_2$ and $c_y \in C_0$.

- We initialize them with the previous $K$ samples and take a single gradient step in the feasible region. The resultant samples are the new $K$ samples.
Experiments

- neuron # 981 volcano class.
Inverse set Intersection

neuron #664 (monastery), [50,60]

neuron #862 (toilet seat), [50,60]

Inverse set Intersection
The goal of understanding what a neuron in a deep neural network may be representing is not a well-defined problem. For some neurons, their preferred response does correlate well with intuitive concepts or classes, such as the example of volcano class.

By characterizing a neuron’s preference by a diverse set of examples, we can explain this preference in a more holistic way.

Our sampling method also has more general applicability; just by modifying the constraints, it can also be used for high dimensional sampling in other domains.
Thank You!