Probabilistics Models of Visual Cortex

Homework 2

Feitong Yang

Department of Psychological and Brain Sciences

Question 1)

A Markove Random Field (MRF) is a graphical model of a joint distribution of multiple random variables. The graphic representation is a graph $G = (V, E)$, where each node $v \in V$ is a random variable, and the edges $e \in E$ represents the conditional independence relationships between random variables.

If no edges in a MRF, and all nodes are separated from each other, then all of these random variables are independent of each other. To use MRF to include spatial context, we can define edges that connects each node with its neighboring nodes. For example, in a 2D grid graph, $V(i, j)$ may have edges to $V(i-1, j), V(i, j-1), V(i+1, j), V(i, j+1)$, and such connection denotes that each node and its four neighboring nodes are not independent, such include a spatial interaction between the information of these random variables. The larger and more complicated the connections, the larger the spatial context is included.

In the example of Ising model, defined by a Gibbs Distribution $P(\overrightarrow{S} \mid \overrightarrow{I}) = \frac{1}{Z}\exp\left(-E(\overrightarrow{S}; \overrightarrow{I})\right)$, where the energy $E(\overrightarrow{S}; \overrightarrow{I})$ is defined as

$$E(\overrightarrow{S}; \overrightarrow{I}) = \sum_x (S(x) - I(x))^2 + \lambda \sum_x \sum_{y \in Nbh(x)} (S(x) - S(y))^2$$

In the energy function, the term $\lambda \sum_x \sum_{y \in Nbh(x)} (S(x) - S(y))^2$ specifies a spatial context for each node: each node is related to its four neighboring nodes. This is actually defined by model before any information from the input image. Because such term denotes our prior hypothesis that neighboring pixels should all be either background or foreground, this can be related to the **prior** of the ising model.

In contrast, the term $\sum_x (S(x) - I(x))^2$ is only driven by the image input. To minimize such term is to make the $\overrightarrow{S}$ to be more and more similar to input image $\overrightarrow{I}$, such term can be think of, or related to, the **likelihood** of the ising model.

If only the first term $\sum_x (S(x) - I(x))^2$ is presented, the ising model will get the most likely results based on the input image. If only the second term $\lambda \sum_x \sum_{y \in Nbh(x)} (S(x) - S(y))^2$ is presented, the ising model will drive neighboring pixels to be the same, which follows our prior guess that neighboring pixels are from the same label – either all are background or all are foreground.

The Gibbs Distribution of the Ising model is hard to sample from, especially when we have lots of random variables in $\overrightarrow{S}$. Suppose each random variable has k states, and there are n random variables, there will be k^n total states, making such model really hard to work with. In order to sample or work with such model, we need to estimate the normalizing term Z, which require us to sum up values from all the states. Such summation could be impossible to compute becuse the total number of states could increase exponentially as the number of random variables increases. Thus, we will not be able to know the full joint distribution $P(\overrightarrow{S} \mid \overrightarrow{I}) = \frac{1}{Z} \exp\left(- E(\overrightarrow{S}; \overrightarrow{I})\right)$

Now we use Gibbs Sampling distributions methods to approximate the full joint distribution. The Gibbs sampling method generates lots of samples from a set of conditional distributions to approximate the full distribution. For each random variable, we hold the state of all other random variables the same, and sample the current random variable from

$$\begin{aligned} P(s_i \mid \overrightarrow{I}, \overrightarrow{S}_{-i}) &= P(s_i \mid I_i, S(y) : y \in Nbh(x)) \\ &= \frac{1}{Z_i} \exp\left((s_i - I_i)^2 + \lambda \textstyle\sum_{y \in Nbh(x)} (s_i - S(y))^2 \right) \end{aligned}$$

Each of such distribution is much easier to work with because each distribution only has one random variable of $k, k = 2$ states, we can easily calculate that

$$Z_i = \sum_{s_i=0}^{1} \exp\left((s_i - I_i)^2 + \lambda \sum_{y \in Nbh(x)} (s_i - S(y))^2 \right)$$

In total, we have nk states to sample, and we can generate lots of samples in these nk states to approximate the original full joint distrubtion. Now, the computation is much cheaper and feasible.

In such sampling, each step is a Monte Carlo sampling from a conditional distribution. However, successive samples are not independent because each time we hold all other random variables constant and generate a new sample of a specific random variable based on all other random variables. Such process is called the Markov Chain Monte Carlo process. By generating lots of samples from this set of transition conditional probability distributions, the Markov Chain properties suggest such sampling will converges to its stationary distribution that is a joint distribution of all random variables. In the current case, such stationary distribution will be the full Gibbs distribution in the beginning.

If we take a look at a network of neurons, each is modeled by a integrate-to-fire model, we see its probability distribution is

$$P(s_i|\overrightarrow{I}) = \frac{\exp\Big(s_i(\sum_j w_{ij}I_j - T_i)\Big)}{1+\exp\Big(\sum_j w_{ij}I_j - T_i\Big)}$$

where $\overrightarrow{I}$ are input form neurons that are connected to s_i, and T_i is this neuron's firing threshold. To update the state of each neuron, we hold the states of other neurons constant, and take the input from other neurons in the distribution to sample a new state of the current neuron. This process is the same of gibbs sampling update process. Here, each neuron only has two states: fire, or not fire. In contrast, a general Gibbs sampling distribution, each random variable may have multiple states. Thus, the integrate-to-fire model can be treated as a special case of a Gibbs sampling distribution. If a fully connected network of neurons use this integrate-to-fire model for each neuron, and we assume a symmetric matrix of weights, then we can deduce a full joint distribution of such network, which turns out to be a Boltzmann machine model, whose full distribution would be

$$P(\overrightarrow{S}) = \frac{1}{Z}\exp\Bigg(-\Big(\sum_{i<j} w_{ij}s_is_j - \sum_i T_is_i\Big)\Bigg)$$

Question 2)

The *mean field theory approximate* for the Ising model $P(\overrightarrow{S}\mid\overrightarrow{I})$ is to use a family of distributions $Q(\overrightarrow{S}) = \prod_i q_i(s_i)$ to approximate the original distribution. Here, the mean field theory specifically define the structure of the $Q(\overrightarrow{S})$ such that each random variable is independent of the rest random variables. Such distribution is much easier to work with because we can sample each random variable individually.

Since we are using a new distribution to approximate the original distribution, we need to find the best of such new distribution $Q^*(\overrightarrow{S})$. To find this best candidate, we need to define a metric that can measure the similarity of two distributions. The Kullback-Leiber divergence is such measurement that $KL(Q(\overrightarrow{S})||P(\overrightarrow{S}\mid\overrightarrow{I}))\geq 0$. The equality holds only when $P(\overrightarrow{S}\mid\overrightarrow{I}) = Q(\overrightarrow{S})$

$$\begin{aligned}
KL(Q||P) &= \textstyle\sum_{S=s} Q(s)\log\frac{Q(s)}{P(s|\overrightarrow{I})}\\
&= \textstyle\sum_s Q(s)\log Q(s) - \sum_s Q(s)\log P(s|\overrightarrow{I})\\
&= \textstyle\sum_s Q(s)\log Q(s) - \sum_s Q(s)\Big(\log\frac{1}{Z} - E(\overrightarrow{S};\overrightarrow{I})\Big)\\
&= \textstyle\sum_s Q(s)\log Q(s) + \sum_s Q(s)E(\overrightarrow{S};\overrightarrow{I}) + \log Z
\end{aligned}$$

Last step holds because $\sum_s Q(s) = 1$.

Because the constant Z is not concerned in optimization, we need to opimize the rest terms, which we call them *free energy*

$$\begin{aligned} F(Q) &= \textstyle\sum_s Q(s)\log Q(s) + \sum_s Q(s)E(\overrightarrow{S};\overrightarrow{I}) \\ &= \textstyle\sum_{s_i} q_i(s_i)\log q_i(s_i) + \sum_{s_i} q_i(s_i)(s_i - I_i)^2 + \sum_{s_i}\sum_{s_j\in Nbh(s_i)} q_i(s_i)q_j(s_j)(s_i - s_j)^2 \end{aligned}$$

In order to minimize such energy, we can use gradient decent related methods to update the free energy to a lower state

$$q_i^{t+1} = q_i^t - \alpha \cdot \frac{\partial F(Q)}{\partial q_i^t}$$

where

$$\frac{\partial F(Q)}{\partial q_i} = (s_i - I_i)^2 + 2\sum_{s_j\in Nbh(s_i)} q_j(s_j)(s_i - s_j)^2 - [q_i\log q_i + (1-q_i)\log(1-q_i)]$$

where $q_i = q_i(s_i = 1)$.

It turns out that the dynamic of a neuron in deterministic neural network, given by

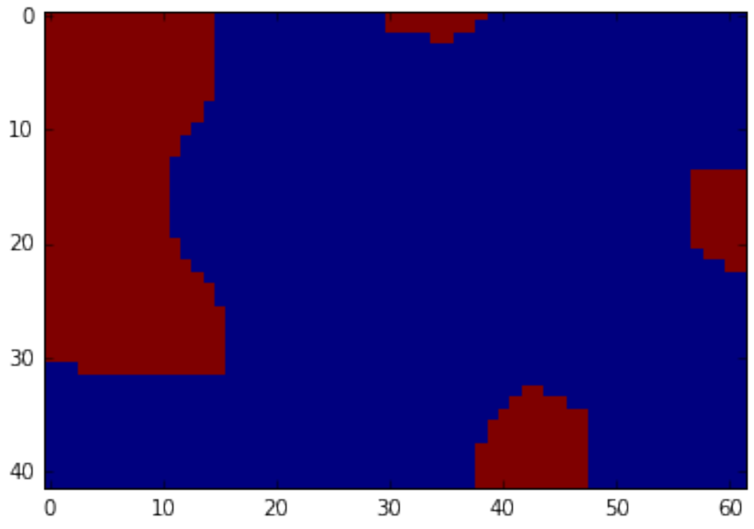
$$\frac{du_i}{dt} = -u_i + \sum_j W_{ij}I_j + \sum_k \theta_{ik}q_k$$

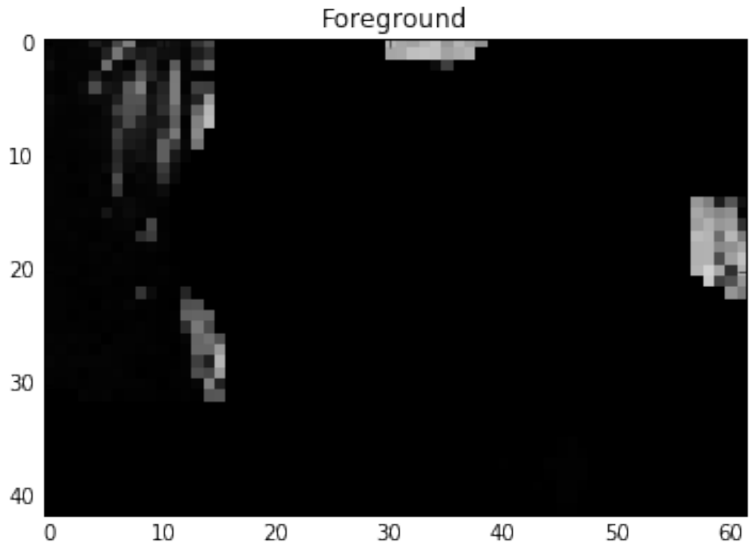
is actually performing the steepest descent during its updates, and minimize the free energy in the mean field theory model. That is, the dynamic system is a mean field approxiamtion to the original stochastic model. Updating the dynamic system is updating the mean fileld approximation.

Question 3)

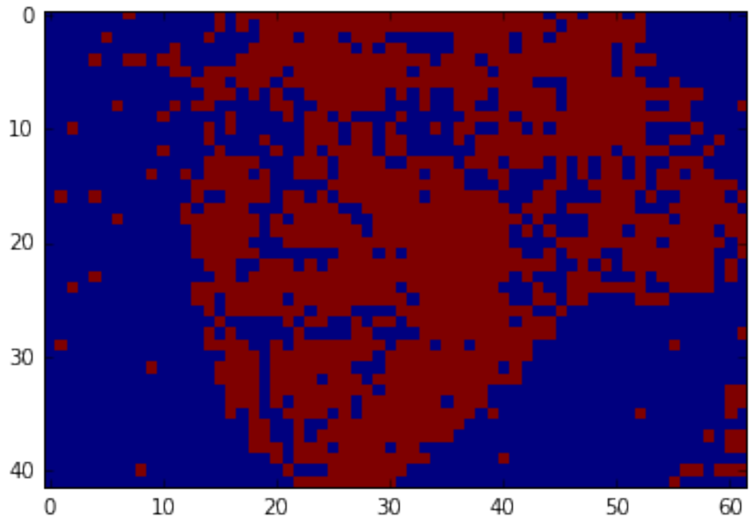
a) try some γ, and η

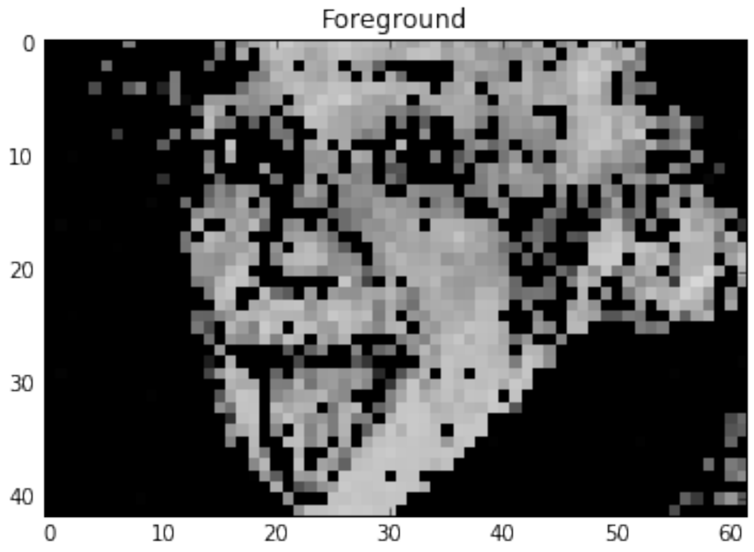
- $\gamma = 20, \eta = 0$: only rely on priors



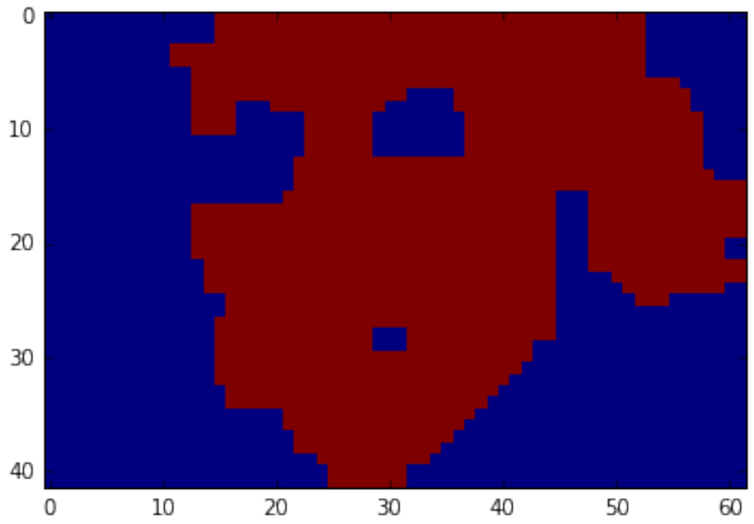


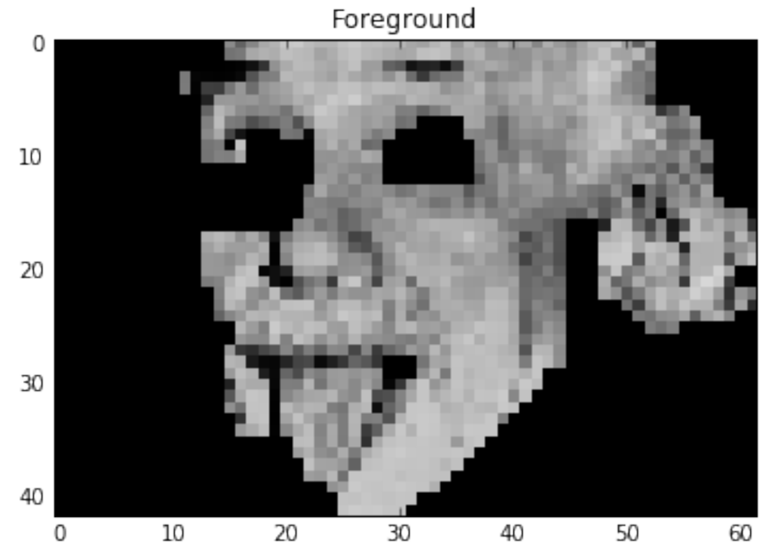
- $\gamma = 0, \eta = 1$: only driven by the image





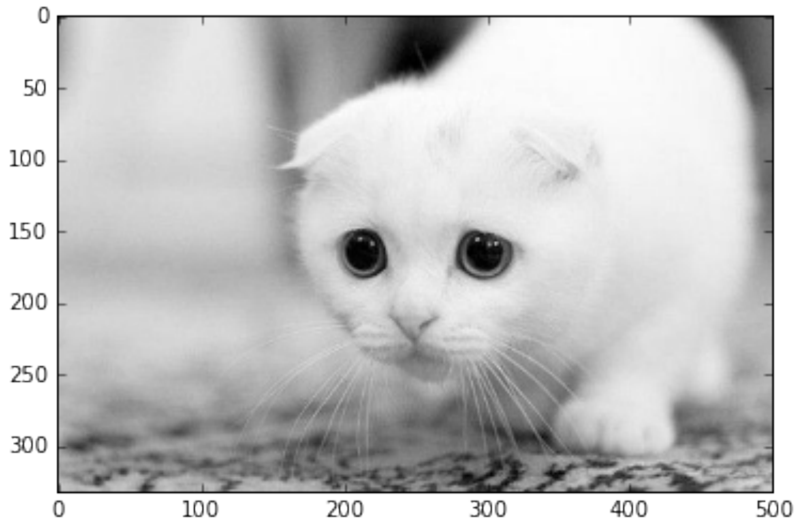
- $\gamma = 20, \eta = 10$: give input larger weight

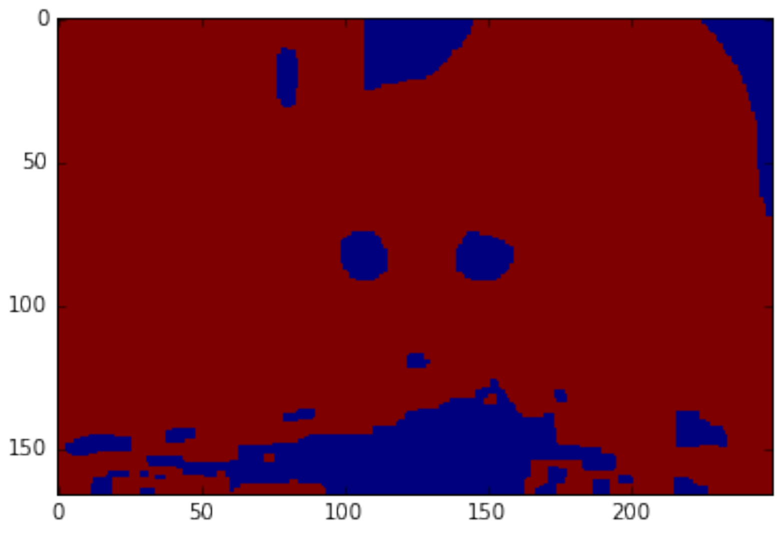


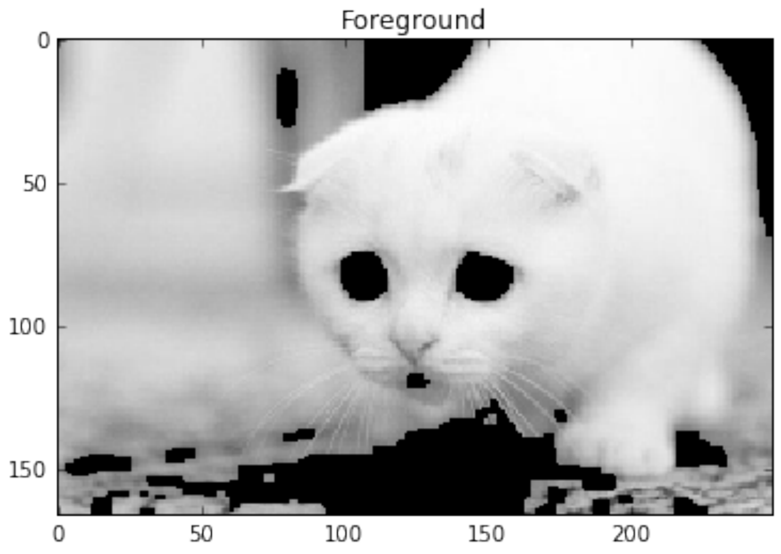


b) try other pictures

- White cat ,$\gamma = 20, \eta = 10$

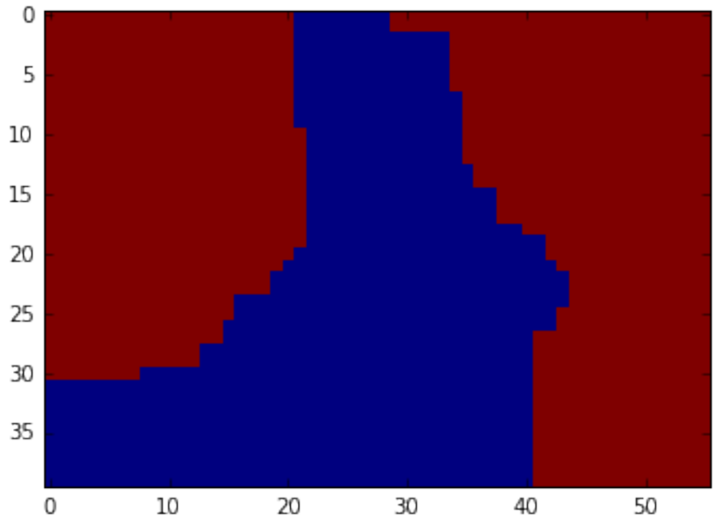


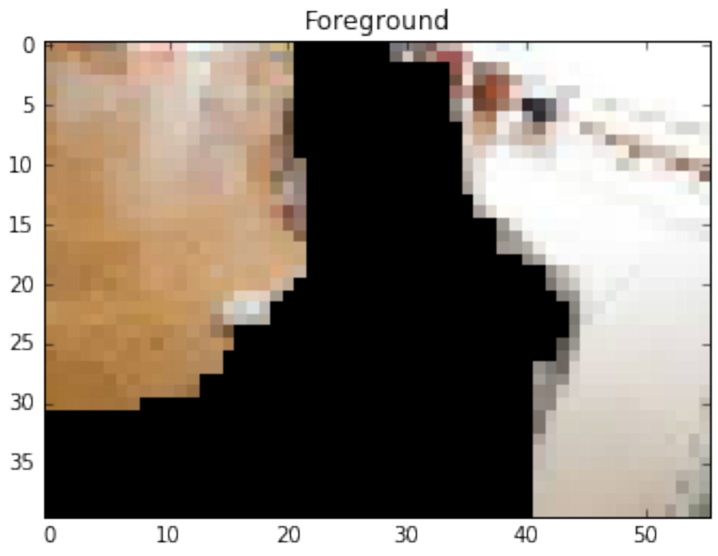




Because the background is quite light, the algorithm cannot segment the cat from the background, back mistakenly treat the cat's eyes to be background.

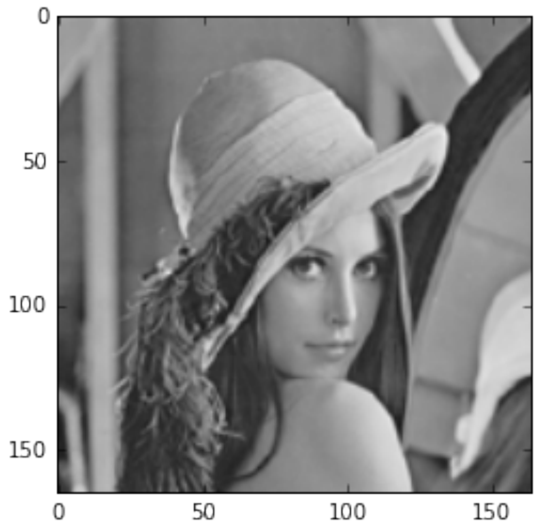
- Dark cat, $\gamma = 20, \eta = 1$

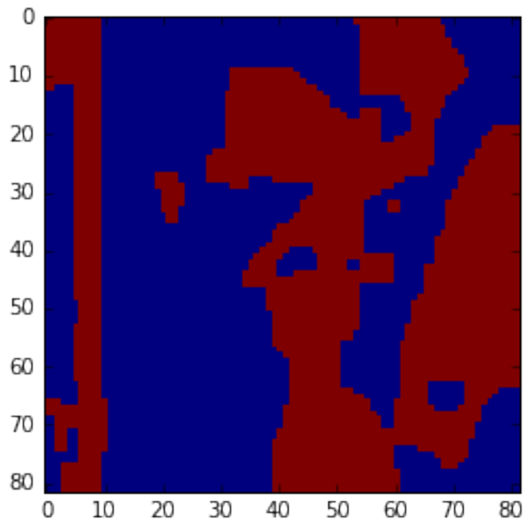


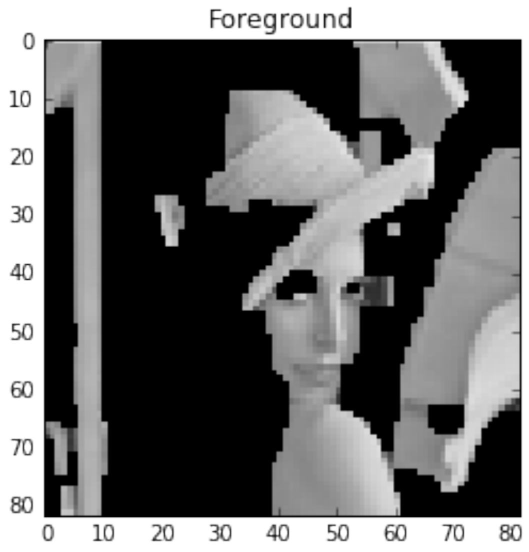


Now the algorithm treat the cat, which should be the figure, as the background, and stripeed it away from the image. The stripped part is largely constrained in the cat's body. If we don't set $s = -1$ as the background by default, and allow $s = -1$ and $s = 1$ just represent two states, then this picture is segmented well.

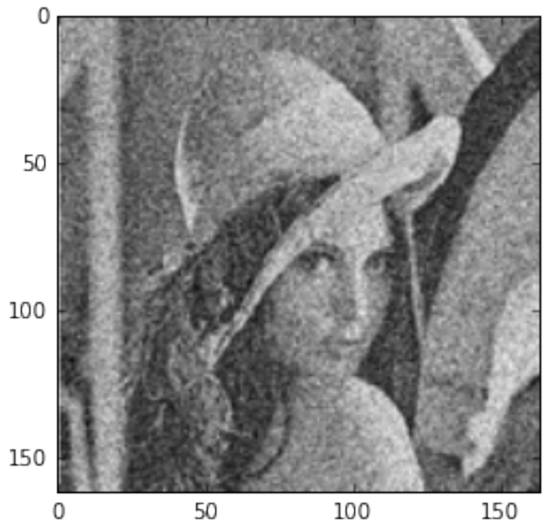
- Classic figure of the lady and its salt-and-pepper-noise version $\gamma = 20, \eta = 10$

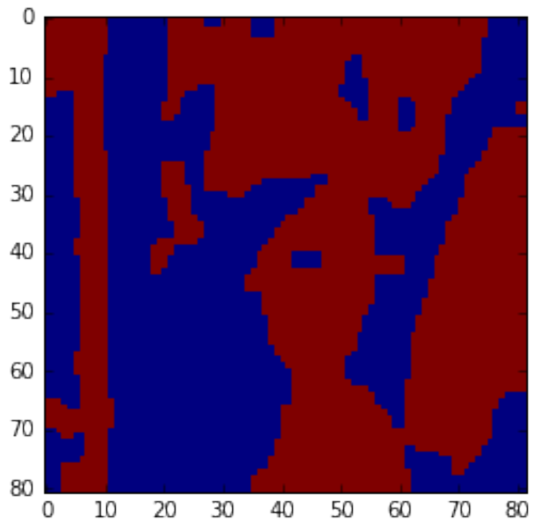


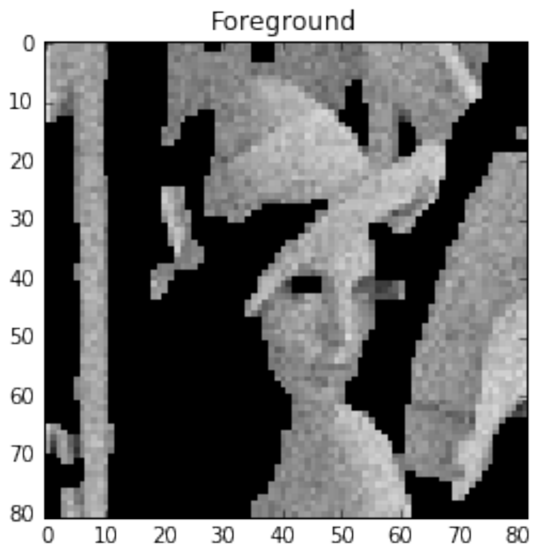




We can see the segmentation is fairly good. What if we add some noise?







We found that even with salt-and-pepper noise, the algorithm has a good segment result.Actually, the ising model is used in de-noise applications in computer vision.

c) when converges?

Note: there seems to be an error in the original

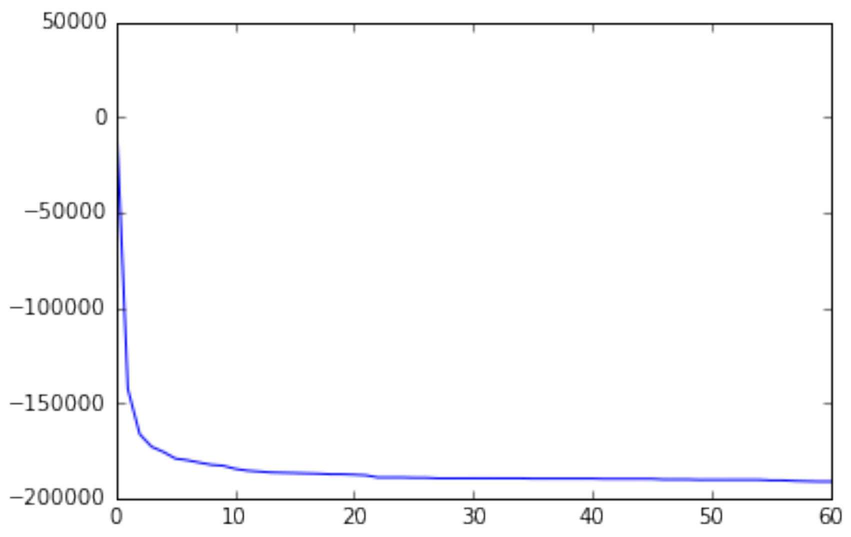
```
def energy_data(state, data, eta):
    E = eta * sum((data - state)**2)
    return E
```

It does not return a single value. So i changed into

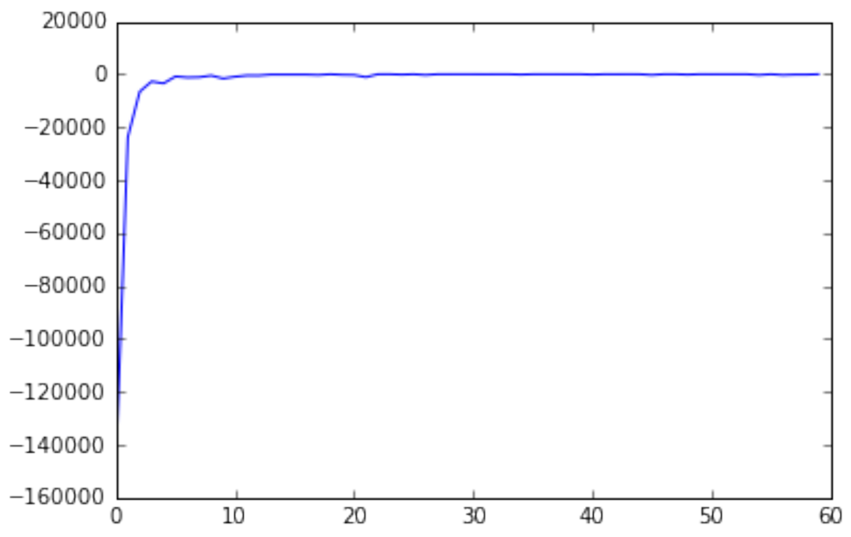
```
def energy_data(state, data, eta):
    E = eta * sum(sum((data - state)**2)) #fixed an error
    return E
```

Now we plot the enery of the 60 iterations and the change of the energy of these iterations, we have :

Energy



Change of Energy



When look at the value, after 5 iterations, the absolute change value is never larger than 1500. After 10 iterations, the absolute change is usually less than 1000. There is one peaky change at 22nd iteration, whose absolute value is about 1100.

Considering the first iteration, energy drops 1.21×10^5, after 10 iterations, the energy changes less than 1000, we can safely say the sampling converges after 10 iterations.

> For all code, please refer to the supplementary materials