Deep Learning HW3

March 15, 2023

Q1

a) Energy-based models enable such a situation by mapping (x, y) pairs to energies instead of mapping x to y. We want to find the lowest energy mapping.

b) Energy models do not generate a probabilistic distribution, they generate the unnormalized score. Because of this, they can be used for modeling situations where the mapping from x_i to output y_i is one to many, even allowing for cases like text generation and object detection where every input x_i might have infinitely many potential outputs.

c) To calculate probabilistic from energy we can we the Gibbs-Boltzmann distribution

$$P(y \mid x) = \frac{e^{-\beta F(x,y)}}{\int y' e^{-\beta F(x,y')}}$$

- Where B is a positive constant

d) Energy $F(x, y)$ takes low value when y is compatible with x and higher values when y it less compatible with x. Hence, energies are used for inference, not for learning. Loss however is used to define the badness of the parameters (for example weights) and is used while training to shape the energy function.

e) Using only positive examples can make the energy landscape flat. Hence, it can lead to a collapse of the energy function. This can be avoided by 1. Using contrastive methods 2. Using Regularized methods 3. Using Architectural methods

f) The three methods that can be used to shape the energy function are

1. Contrastive methods: Push down the energy of the positive examples and pull -up the energy of the negative examples.
   - There are different ways to pick which point to push up
   - Example: Max likelihood, marked auto-encoder
2. Regularized methods: Use a regularization term that measures the volume of the space that has low energy - Example: sparse auto-encoder, variational auto-encoders
3. Architectural methods: Build the model so that the volume of low-energy space is bounded. - Example: PCA, K-means, Gaussian Mixture Model

g) $l_{nll(x,y,w)} = F_w(x, y) + log \int_{y'} e^{-\beta F(x,y')}$

h)

$$argmin_{\hat{y}} F(x, \hat{y})$$
$$argmin_{\hat{y}\hat{z}} G(x, \hat{y}, \hat{z})$$

Q1.2(i)

$$P(y \mid x) = \frac{e^{-\beta F(x,y)}}{\sum_{y'} e^{-\beta F(x,y')}}$$

(ii)

$$L(x, y, w) = \log P(y \mid x) = \log \left(\frac{e^{-\beta F(x,y)}}{\sum_{y'} e^{-\beta F(x,y')}} \right)$$

$$= -\log \left(e^{-\beta F(x,y)} \right) + \log \left(\sum_{y_1} e^{-\beta F(x,y')} \right) = \beta F(x, y) + \log \left(\sum_{y,} e^{-\beta F(x,y')} \right)$$

(iii) Now,

$$\frac{L(x, w, y)}{\beta} = F_w(x, y) + \frac{1}{\beta} \log \left(\sum_{y'} e^{-\beta F_w(x,y')} \right)$$

from the chain rule, we get

$$\Rightarrow \frac{\partial L(x, w, y)}{\beta \partial w} = \frac{\partial F_w(x, y)}{\partial w} + \frac{1}{\beta} \times \frac{1(-\beta) \times \sum e^{-\beta F_w(x,y')} \frac{\partial f}{\partial w}}{\sum e^{-\beta F_w}(x, y')} (1)$$

$$= \partial F_w(x, y)/\partial w - \sum \frac{e^{-\beta F_w(x,y')}}{\sum e^{-\beta F_w(x,y')}} \frac{\partial F(x, y')}{\partial w}$$

Now, $P(y' \mid x) = \frac{e^{-\beta F_w(x,y')}}{\sum_{y'} e^{-\beta F_W(x,y')}}$ (2)

Substituting (2) in (1), we get

$$\frac{\partial L(x, w, y)}{\beta \partial w} = \frac{\partial F_w(x, y)}{\partial w} - \sum_{w} P_w(y' \mid x) \left(\frac{\partial F_w(x, y')}{\partial w} \right)$$

Calculating this gradient might be intractable due to a large number of values of y', which makes the calculation of the & $\sum e^{-\beta F_w(x,y')}$ term in (1) intractable. We can solve this by using Markov Chain Monte Carlo methods.

(iv) The loss function $L(x, y, w)$ is given by $F(x, y) + \frac{1}{B} \log \left(\sum_{y'} e^{-\beta F(x,y')} \right)$ where $F(x, y)$ is the energy of the correct example and the second term is the energy of the other examples. We note that NLL pushes the energy of a $y\prime$ in the proximity of a y with a force proportional to the probability of that particular $y\prime$ rather than the proportion of the distance between y and $y\prime$. This leads to a huge spike.

$Q1.3$ (a)

$$l_{simple}(x, y, \bar{y}, w) = [F_W(x, y)]^+ + [m - F_w(x, \bar{y})]^+$$

$$\frac{\partial l}{\partial w}$$

$$= \frac{\partial F_W(x,y)}{\partial w} - \frac{\partial F_w(x,\bar{y})}{\partial w} when F_w(x,y) \geq 0 \& F_w(x,\bar{y}) \leq m$$

$$= -\frac{\partial F_w(x,\bar{y})}{\partial w} \quad when F_w(x,y) < 0 \& F_w(x,\bar{y}) \leq m$$

$$= \frac{\partial F_w(x,y)}{\partial w} \quad when F_w(x,y) > 0 \& F_w(x,\bar{y}) > m$$

=0 otherwise

(b)

$$l_{\log}(x, y, \bar{y}, w) = \frac{e^{F_w(x,y) - F_w(x,\bar{y})}}{1 + e^{F_w(x,y) - F_w(x,\bar{y})}} \times \left[\frac{\partial F_w(x,y)}{\partial y} - \frac{\partial F_w}{\partial y}(x,\bar{y}) \right]$$

(c)

$$\frac{\partial l}{\partial w}$$

$$= 2F_W(x,y)\frac{\partial F_W(x,y)}{\partial w} - 2F_w(x,\bar{y})\frac{\partial F_w(x,\bar{y})}{\partial w} when F_w(x,y) \geq 0 \& F_w(x,\bar{y}) \leq m$$

$$= -2F_w(x,\bar{y})\frac{\partial F_w(x,\bar{y})}{\partial w} \quad when F_w(x,y) < 0 \& F_w(x,\bar{y}) \leq m$$

$$= 2F_w(x,y)\frac{\partial F_w(x,y)}{\partial w} \quad when F_w(x,y) > 0 \& F_w(x,\bar{y}) > m$$

=0 otherwise

(d) NLL is different from the simple, log and least-square loss in the sense that it is dependent on $\sum_{y'}$ (expression) term, which makes gradient and loss calculation intractable for a large number of classes (y'). On the other hand, the three losses are not dependent on a $\&_{y'}$ (expression) term.

(ii) The log loss is called soft hinge loss because its underlying function is similar to hinge loss, however, instead of having a hard threshold for the margin, it uses a continuous smooth function to penalize incorrect predictions by using the log (1+ expression) term.

(iii) The simple loss and squared-squared loss are different from hinge or log loss in the sense that simple and squared-squared will keep ensuring a gradient until all the correct examples do not have an average negative energy in a batch, while hinge or log loss supply a gradient only as long as the difference between correct and incorrect examples is not at least m. This means that it will stop supplying a gradient even when the correct examples have very high negative energy, while the incorrect examples do not necessarily have an energy $> m$.

The advantage of using simple loss is that it ensures a high level of accuracy and can be useful when the underlying data is not noisy. However, models like sVM use hinge-loss to make the learning robust to outliers. Hence the hinge/ log loss is useful when we know that our data is noisy. This is because hinge/log loss is okay with allowing certain examples to be misclassified/have on energy less than m, even if they are incorrect.