Score Based Models
Advanced Topics in Deep Learning

Vahid Tarokh

ECE 590-02

Spring 2022

Partition Function

- In the previous lecture, we discussed energy-based models.
- The main issue with these models was the calculation involving the partition function.
- We considered Monte Carlo approximation methods both for the training and testing of energy models and in particular those based on deep neural networks.
- The results were quite good. We wonder how better it can be if we did not have to approximate.
- In this lecture, we will discuss Fisher Score based approaches to address the partition function.
- First we discuss general score functions.

Score Functions

- How to evaluate the cost/utility of models fit to data?
  - Probability density function p and data observations y
  - Score function $s: (y, p) \mapsto s(y, p)$
  - The smaller, the better
  - Example: logarithmic scoring function $s(y, p) = -\log p(y)$
- Score rule
  - Logarithmic rule
    - Minimize $s(y, p) = -\log p(y)$ (here y can be a vector)
  - MLE
    - $p \equiv p_\theta$, obtain the θ that achieves the minimum of $s(y, p)$
    - Widely used

Overview of Logarithmic Score

- Logarithmic Score
  - Widely adopted for statistical inference, information theory, etc.
  - Examples: maximum likelihood estimation, hypothesis test, Bayesian model comparison
- Why is it so popular?
  - Elementary function of the probability density function
  - Intimate relation with KL-divergence
    - Minimizing
    - $n^{-1} \sum_i s(y_i, p) = n^{-1} \sum_i \{-\log p(y_i)\}$
  - is asymptotically equivalent to minimizing
  - $E_*\{-\log p(y)\} = D_{KL}(p_* || p)$
  - (under i.i.d. or ergodicity assumptions)
  - In a well-specified scenario: optimum $p = p_*$
  - In a mis-specified scenario: optimum p is the closest to p_* in KL sense (under some regularity conditions)

p_*: true distribution

E_*: expectation w.r.t. p_*

Maximum Likelihood Estimator (MLE)

Notice that

$$-\frac{\Sigma_1^n \log p_\theta(y_i)}{n} \rightarrow E_{p^*}\left[-\log\ p_\theta(y)\right]$$

- Minimizing

$$-\frac{\Sigma_1^n \log p_\theta(y_i)}{n}$$

is asymptotically equivalent to minimizing

$$E_*\{-\log p_\theta(y)\} = D_{KL}(p_*||p_\theta) + \mathrm{H}(p_*)$$

or equivalently

$$D_{KL}(p_*||p_\theta).$$

Some Thoughts

- Up to now, we mainly used MLE, cross-entropy, etc. for the training of our models.
- The motivation as to minimize KL divergence between the true model and the postulated modeled (e.g. DNN based).
- This was equivalent to minimizing the logarithmic score (as discussed)
- By doing this, we ran into the curse of partition function.
  - We did Monte-Carlo approximation to address this.
- Not clear that logarithmic score generates the best images perceptually.
- May be we can use another score that does not have the curse of partition function?
- If possible, then we must minimize another distance with the true model

Fisher Divergence and the Fisher Score

- Fisher score from Fisher divergence
  - Consider minimizing the following Fisher divergence instead of KL

$$\frac{1}{2}E_*\|\nabla_y \log p(y) - \nabla_y \log p_*(y)\|^2 \quad (1)$$
$$= E_*\{s_P(y,p)\} + c_*$$

$$s_P(y,p) = \frac{1}{2}\|\nabla_y \log p(y)\|^2 + \Delta_y \log p(y),$$

  - c_* only depends on p_*, the true data-generating distribution. (prove this)
  - The minimum, zero, is achieved if and only if $p(y) = p_{\theta_*}(y)$
    p_{θ_*}: the closest density to p_* in the sense of (1)
    θ_*: true parameter (if well-specified)
- Suppose we minimize $n^{-1}\sum_i s_P(y_i, p_\theta)$ **(e.g. using SGD)**
  - It is approximately minimizing $E_*\{s_P(y, p_\theta)\}$
  - It leads to $\theta \to \theta_*$ in probability

Ronald Aylmer Fisher

Desirable properties of Fisher-Score

- A **simple** function of $p(y)$ and its derivatives
- **Proper** in the sense that $\mathrm{E}_*\{s(y,p)\}$ is only minimized at $p = p_*$ a.s. under mild conditions
- **Scale-invariance** in the sense that $\mathrm{E}_*\{s(y,p)\} \equiv \mathrm{E}_*\{s(y,c\,p)\}$

p does not need to be a density function as long as c does not depend on y

- This means that calculations can be done with unnormalized distributions. **Partition function will not be playing a role at all!**
  - This is unlike the logarithmic score.
- May be defined for **improper priors.**
  - The energy model does not even need to be integrable!

Relationship Between Fisher and KL Divergences.

- Relationship between Fisher Divergence and KL

$$D_{\nabla}(p,q) = -2\,\frac{d}{dt} D_{\mathrm{KL}}(p_t, q_t)\big|_{t=0}$$

- $X \sim p(\cdot)$
- $\mathrm{Y} \sim q(\cdot)$
- $X_t = X + \mathcal{N}(0,t)$
- $Y_t = Y + \mathcal{N}(0,t)$ *perturbed by independent* $\mathcal{N}(0,t)$
- $\mathrm{X_t} \sim p_t(\cdot)$
- $\mathrm{Y_t} \sim q_t(\cdot)$
- The proof is similar (and follows from) de Bruijn equality. $J_F(X) = 2\frac{d}{dt}\,[H(X_t)]$

Incorporation into AI systems

- Fisher score can be used for
  - Parameter inference
  - Sequential Monte Carlo
  - Bayesian model comparison
  - Causality Calculations
  - Change detection
  - **Training Energy Models (Topic of current lecture)**
- Fisher score may have some advantages over logarithmic scores in some cases
  - both computational and foundational.

Energy models

- We can think of the the function given by NN_θ as $g_\theta(y)$ and the corresponding value of partition as Z_θ. Thus

$$p_\theta(\mathrm{y}) = \frac{e^{-g_\theta(y)}}{Z_\theta}.$$

- Calculating the Fisher score:

$$\nabla_{\mathrm{y}} \ln p(y) = -\nabla_{\mathrm{y}}\, g(y)$$

$$\Delta_y \ln p(y) = -\Delta_y\, g(y).$$

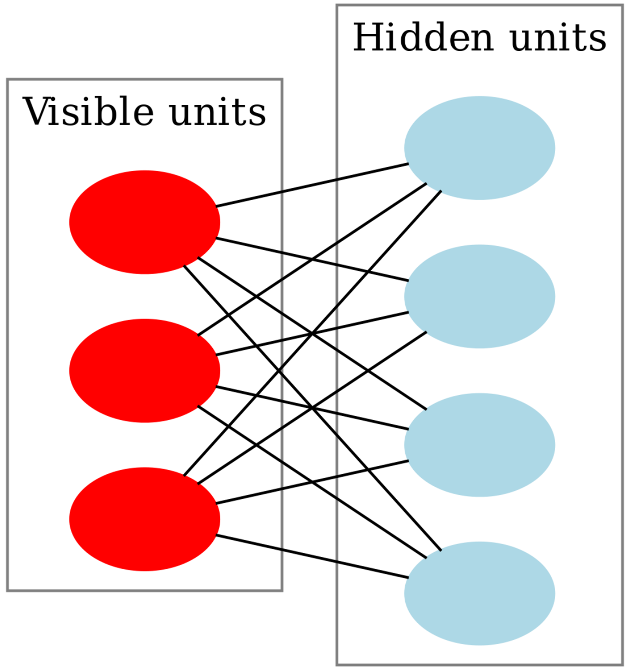
- The Fisher score is given by

$$½\, |\nabla_{\mathrm{y}}\, g(y)|^2 - \Delta_{\mathrm{y}}\, g(y)$$

- This does not require the calculation of the partition function.
- We will present some examples.

Gauss-Bernoulli Boltzmann machine

- Continuous data modeled by Restricted Boltzmann machine (RBM) (see ECE685D Notes)



  - An RBM consists of m visible **real valued** units **V** and n binary hidden units **H**
  - The energy function is

$$E_\theta(V,H) = -H^T W V - C^T V - B^T H + \frac{1}{2} V^T V$$

  - The probability density is

$$p_\theta(V,H) = exp(H^T W V + C^T V + B^T H - \frac{1}{2} V^T V)$$

  - $\theta = (\boldsymbol{W}, \boldsymbol{C}, \boldsymbol{B})$ denote all the parameters of the model

Gauss-Bernoulli Boltzmann Machine

- The Fisher score function can be easily calculated.

$$s_\theta(V) = \frac{1}{2}||W^T\rho + C + V||^2 + ||\mathrm{diag}(\rho')W||^2$$

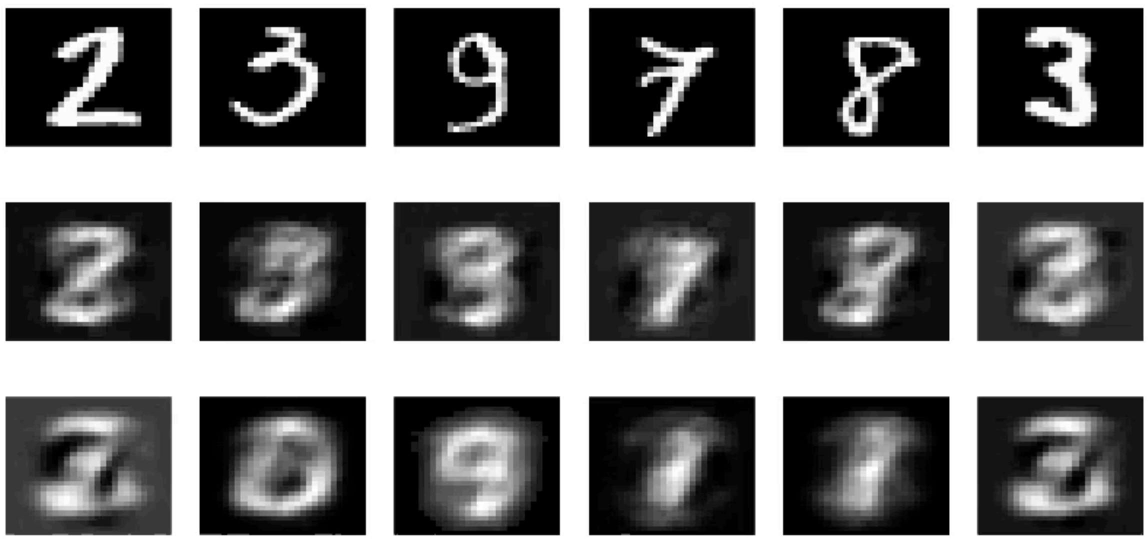
where $\rho = \sigma(WV + B)$, $\rho' = \sigma'(WV + B)$

- We would like to minimize the sum of Fisher score over all training data points. To this end, we use stochastic gradient descent.
- This means that we need to calculate the derivatives of score function with respect to θ.
- This calculation is straightforward but somewhat painful. The results can be put in closed form and involve the sigmoid functions and its derivatives.
  - Exercise: Verify these calculations.
- We use these derivatives for the training with SGD.

Training of Restricted Boltzmann Machine

- Continuous data modeled by Restricted Boltzmann machine (RBM)
  - An RBM consists of 784 visible units **x** and 30 hidden units **h**

  At the same computational complexity, our approach creates a generative model that is closer to the underlying truth



Original Images

Generated Images using Fisher

Generated Images using Classical KL Divergence (approximated by Contrastive Divergence)

Video: Training of RBMs using New versus Classical Approaches

Special Energy Models

- Some special energy models are particularly suitable for the application of Fisher score.
- Consider the exponential family (see ECE685D notes)

$$p_\theta(y) = \exp\{\, a(y) + b(\theta) + \theta^T s(y)\}$$

  - By minimizing the Fisher score $\sum_{t=1}^{n} s_P(y_t, p_\theta)$
  - we can have the closed form solution (usually not possible with MLE)

$$\hat{\theta}_H = -\left(\sum_{t=1}^{n} J(y_t)^T J(y_t)\right)^{-1} \left\{\sum_{t=1}^{n} \Delta s(y_t) + J(y_t)^T \nabla a(y_t)\right\}$$

where $J(y)$ denotes the matrix $J(y) = \left[\frac{\partial s_j(y)}{\partial y_i}\right]_{i,j}$ and $\Delta s(y)$ denotes the vector $\left[\Delta s_j(y)\right]_j$

- No need to apply numerical techniques.

Deep Neural Networks as Energy Models

- We can think of the the function given by NN_{θ} as $g_{\theta}(y)$.
- To calculate the Fisher Score, we need to calculate

$$\frac{1}{2}|\nabla_{y}\, g(y)|^{2} - \Delta_{y}\, g(y)$$

- The first part is easy to calculate, but the Laplacian may be computationally complex to calculate.
- Thus it is not immediately clear how to apply the Fisher score for the training of energy models where the energy function is given by a large deep neural network.
- We must think of some other remedies.

Key idea

- Remember that we started from the Fisher divergence with data the true pdf of data $p(x)$:

$$\frac{1}{2} E_* \|\nabla_x \log p_\theta(x) - \nabla_x \log p\,(x)\|^2 = E_*\{s_P(x, p_\theta)\} + c$$

- In a sense all we have been trying to do is to come up with a model $p_\theta(x)$ whose $\nabla_x \log p_\theta(x)$ is close to $\nabla_x \log p(x)$.
  - This led to Fisher score.
- If we estimate $\nabla_x \log p(x)$ well, then we can use MALA to create new images that look like the original images.
  - The estimate $\nabla_x \log p(x)$ is all MALA needs.
- Now suppose we made a deep neural network NN_θ whose output $s_\theta(x)$ estimates $\nabla_x \log p(x)$, then we are in business!
- **This idea applies beyond energy models.**
- It remains to discuss how to construct such a deep neural network whose output

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Sliced Score Matching

- An approach is sliced score matching [Song et al. UAI 2019].
- The ***slicing*** idea is to project a high dimensional optimization objective into random lower dimensional subspaces (in most cases only one dimensional subspaces) and optimize these sums.
- To this end, we generate some unit vectors $\boldsymbol{v}$ according to probability $p_{\boldsymbol{v}}(\boldsymbol{v})$. In most case, we want this distribution to be uniform over all unt directions. This can be achieved by drawing samples from $N(0, I)$ and normalizing the samples to have unit length.
- Now we calculate the sliced Fisher divergence:

$$\frac{1}{2} E_{\mathbf{v}\sim p_{\mathbf{v}}} E_{\mathbf{x}\sim p_{\text{data}}}[(\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^\top \boldsymbol{s}_\theta(\mathbf{x}))^2]$$

- This is the ***sliced*** surrogate of the Fisher divergence that we originally optimized.

Sliced Fisher Divergence

- Just as in Fisher divergence, we expand and integrate by parts and observe that this is equivalent to minimizing

$$E_{\mathbf{v}\sim p_{\mathbf{v}}} E_{\mathbf{x}\sim p_{\text{data}}}\left[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})\mathbf{v} + \frac{1}{2}(\mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2\right]$$

- We can calculate this empirically by generating various directions (unit vectors) $\boldsymbol{v}$ according to probability $p_{\boldsymbol{v}}(\boldsymbol{v})$ at each step of SGD and for the data point calculate the empirical average. Then we can do descent on θ.
- If $s_\theta(x)$ is the output of a deep neural network $NN_\theta(x)$ then calculating $\nabla_x s_\theta(x)$ is doable (in fact Pytorch does it).
- The whole process is summarized in the algorithm given in the next page.

Sliced Score Matching

Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$

Sample a minibatch of projection directions $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\} \sim p_{\mathbf{v}}$

Estimate the sliced score matching loss with empirical means

$$\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{v}_i^\mathsf{T}\nabla_{\mathbf{x}}\boldsymbol{s}_\theta(\mathbf{x}_i)\mathbf{v}_i + \frac{1}{2}\left(\mathbf{v}_i^\mathsf{T}\boldsymbol{s}_\theta(\mathbf{x}_i)\right)^2\right]$$

- Note: We can use more than one projections per datapoint too.
- Use Stochastic Gradient Descent to minimize the cost

Denoising Score Matching

- Denoising score matching [Vincent 2011] is another way to train DNNs based on matching the score of a noise-perturbed distribution of the data generating distribution $p_{\text{data}}(\mathbf{x})$
- Noise is added to training images x to arrive at the noisy images $\tilde{x}$.

$$p_{\text{data}}(\mathbf{x}) \Longrightarrow q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \Longrightarrow q_\sigma(\tilde{\mathbf{x}})$$

- Typically the noise is Gaussian $N(0, \sigma^2)$.
- Now we try to have $s_\theta(x)$ (the output of DNN) match $q_\sigma(\tilde{\mathbf{x}})$
- We do some calculations next.

Some Calculations

- We have

$$\frac{1}{2}E_{\tilde{\mathbf{x}}\sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] = \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\,\|\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2\,\mathrm{d}\tilde{\mathbf{x}} + \frac{1}{2}\int q_\sigma(\tilde{\mathbf{x}})\,\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2\,\mathrm{d}\tilde{\mathbf{x}}$$
$$- \int q_\sigma(\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}$$

- The first blue term is independent of training of the DNN:

$$= \text{const.} + \frac{1}{2}E_{\tilde{\mathbf{x}}\sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})^\top \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}$$

- Next we manipulate the second (red) term.

$$
\begin{aligned}
& -\int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\int q_\sigma(\tilde{\mathbf{x}}) \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \,\mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \,\mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \,\mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -\iint p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \,\mathrm{d}\mathbf{x} \,\mathrm{d}\tilde{\mathbf{x}} \\
&= -E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]
\end{aligned}
$$

Computation Continued

- This in turn equals to

$$
\begin{aligned}
=&\text{const.} + \frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] \\
&-\frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\| \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2]
\end{aligned}
$$

which is

$$
\begin{aligned}
&= \text{const.} + \frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.} \\
&= \frac{1}{2} E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}
\end{aligned}
$$

Denoising Score Matching

- Thus we proved that

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$
$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

- Now we observe that $\nabla_{\tilde{x}} \log\ q_\sigma(\tilde{x}|x)$ is easy to calculate. In fact assuming Gaussian noise $N(0, \sigma^2)$ this is equal to $\frac{-(\tilde{x} - x)}{\sigma^2}$.
- This means that the above distance can be computed empirically even in very high dimensions using backpropagation.
- The caveat is that we are matching the $\nabla_{\tilde{x}} \log\ q_\sigma(\tilde{x}|x)$ and not $\nabla_x \log p(x)$.
- Let us summarize this algorithm [Vincent 2011] in the next slide.

Denoising Score Matching (pseudocode)

- Choose a Gaussian $N(0, \sigma^2)$ or any other noise $q_\sigma(\tilde{x}|x)$ with small variance σ and conditional mean x.

Sample a minibatch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$

Sample a minibatch of perturbed datapoints $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_n\} \sim q_\sigma(\tilde{\mathbf{x}})$

$$\tilde{\mathbf{x}}_i \sim q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)$$

Estimate the denoising score matching loss with empirical means

$$\frac{1}{2n} \sum_{i=1}^{n} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i) \|_2^2]$$

If Gaussian perturbation

$$\frac{1}{2n} \sum_{i=1}^{n} \left[\left\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i) + \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i}{\sigma^2} \right\|_2^2 \right]$$

- Use Stochastic Gradient Decent (SGD) to minimize over θ.

Fine-tuning σ

- Let us calculate the effect of σ on the loss function using reparameterization trick (please see ECE685D Lecture notes).

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[\left\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})+\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right] = \frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\mathbf{z}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})+\frac{\mathbf{z}}{\sigma}\right\|_2^2\right]$$

$$= \frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}E_{\mathbf{z}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})\|_2^2 + 2\boldsymbol{s}_\theta(\mathbf{x}+\sigma\mathbf{z})^\intercal\frac{\mathbf{z}}{\sigma}+\frac{\|\mathbf{z}\|_2^2}{\sigma^2}\right]$$

- Now if $\sigma \to 0$ we can easily see that the variances of $\frac{z}{\sigma}$ and $\frac{||z||^2}{\sigma^2}$ both goes to ∞.
- So we can not let $\sigma \to 0$ as we will encounter computational instability in training.
  - It is important to fine-tune σ.

Fine-tuning σ

- One way to do this [Song and Ermon 2019] is to choose a decreasing sequence of positive numbers $\sigma_1 > \sigma_2 > \cdots > \sigma_L$.
- Then use the Denoising Score Matching (or the sliced score matching algorithm applied to noisy data) with Gaussian noise variance $N\left(0, \sigma_i^2\right)$ to train score function $s_\theta(x, \sigma_i)$ at i-th step.
- Then run MALA with appropriately selected noise variance to sample.
- Use the samples to train for the next $(i+1)$-th step.
- Details are given in the next page.

Annealed Score Matching and MALA

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright$ α_i is the step size.
    for $t \leftarrow 1$ to T **do**
        Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
        $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \dfrac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
    end for
    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Selection of sequence of noise variances

- σ_1 can be selected as the maximum Euclidean distance taken over pairs of data points.
- It has been heuristically observed that $\sigma_1 > \sigma_2 > \cdots > \sigma_L$ can be selected to form a geometrically decaying sequence with the decay rate and L to be fine-tuned.
- Yet another heuristic method is to minimize the weight loss function (recall the reparameterization trick)

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{s}_\theta(\mathbf{x}+\sigma_i\mathbf{z},\sigma_i)+\frac{\mathbf{z}}{\sigma_i}\right\|_2^2\right]$$

where the weight $\lambda(\sigma_i) = \sigma_i^2$

- Combining all these heuristics, we arrive at the following algorithm [Song and Ermon 2019].

Noise conditional score networks Training Algorithm

Sample a mini-batch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}$
Sample a mini-batch of noise scale indices

$$\{i_1, i_2, \cdots, i_n\} \sim \mathcal{U}\{1, 2, \cdots, L\}$$

Sample a mini-batch of Gaussian noise $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Estimate the weighted mixture of score matching losses

$$\frac{1}{n} \sum_{k=1}^{n} \left[\|\sigma_{i_k} \boldsymbol{s}_\theta(\mathbf{x}_k + \sigma_{i_k} \mathbf{z}_k, \sigma_{i_k}) + \mathbf{z}_k\|_2^2 \right]$$

- Use Stochastic Gradient descent to minimize.

Experiments [Song and Ermon 2019]

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32

Experiments: High Resolution Image Generation [Song and Ermon 2019]