Energy-based models (EBM), part 2

Aapo Hyvärinen

for the NNDL course, 16 April 2025 (first half)

- Noise-contrastive estimation
- Comparison of estimators

Literature: Murphy's book Chapter 24

From last time: Estimation of energy-based models (EBM)

- Also called estimation of *unnormalized* statistical models
- Density function is known only up to a multiplicative constant

$$p_{\text{norm}}(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} p_{\text{un}}(\mathbf{x};\boldsymbol{\theta})$$

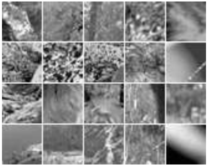
$$Z(\boldsymbol{\theta}) = \int_{\boldsymbol{\xi}\in\mathbb{R}^n} p_{\text{un}}(\boldsymbol{\xi};\boldsymbol{\theta})\, d\boldsymbol{\xi} \tag{1}$$

  with p_{norm}: actual density; p_{un}: unnormalized version
- Functional form of p_{un} is "known" (i.e. can be easily computed)
- Normalization constant (= "partition function") Z "unknown", i.e. *cannot be easily computed*
- Computation of Z based on the above would need numerical integration: very difficult in high dimensions

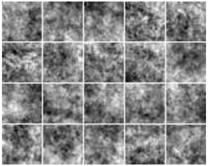
Noise-contrastive estimation (NCE) as already seen in SSL introduction

- Train a nonlinear classifier to discriminate observed data from some artificial noise
- To be successful, the classifier must "discover structure" in the data
- For example, compare natural images with Gaussian noise

Natural images



Gaussian noise



- A nice intuitive idea, but what does this system actually do?
  - ... in addition to learning nice features for supervised learning.

Definition of classifier in NCE

- Observed data set $\mathbf{X} = (\mathbf{x}(1), \ldots, \mathbf{x}(N))$ has *un*known pdf $p_\mathbf{x}$
- Generate "noise" $\mathbf{Y} = (\mathbf{y}(1), \ldots, \mathbf{y}(N))$ with known pdf $p_\mathbf{y}$
- Define a nonlinear function, a NN, $g(\mathbf{u}; \boldsymbol{\theta})$, which models data log-density $\log p_\mathbf{x}(\mathbf{u})$.
  - The function g corresponds to $\log p_{\text{un}}$
  - here we initially focus on general density estimation
- We use standard logistic regression, but with the special nonlinear function

$$G(\mathbf{z}; \boldsymbol{\theta}) = g(\mathbf{z}; \boldsymbol{\theta}) - \log p_\mathbf{y}(\mathbf{z}). \tag{2}$$

  where $\mathbf{z}$ takes values over the whole data (either $\mathbf{x}$ or $\mathbf{y}$).
- Next we prove that such training solves EBM estimation: $\boldsymbol{\theta}$ converges to the true values that generated the data, or: in case of pdf approximation, g converges to log-pdf of $\mathbf{x}$.

Definition of logistic regression and its objective

- Begin by transforming probability p to $p/(1-p)$, "odds ratio"
- Logistic regression uses function r to model *log-odds*

$$r(\mathbf{u};\boldsymbol{\theta}) \approx \log \frac{p(c=1|\mathbf{z}=\mathbf{u})}{1-p(c=1|\mathbf{z}=\mathbf{u})} \tag{3}$$

  where $c \in \{1,2\}$ is class label.
- To derive likelihood, first solve $p(c=1|\mathbf{z}=\mathbf{u})$ as

$$p(c=1|\mathbf{z}=\mathbf{u}) = \sigma(r(\mathbf{u};\boldsymbol{\theta})) \tag{4}$$

  whose solution uses the "logistic function" $\sigma(r) = \frac{1}{1+\exp(-r)}$.
- Likewise we solve: $p(c=2|\mathbf{z}=\mathbf{u}) = 1-\sigma(r(\mathbf{u};\boldsymbol{\theta}))$.
- Take sum of logarithms of the likelihood for the two classes
- This gives the standard log-likelihood objective for training:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log\left[\sigma(r(\mathbf{x}(i);\boldsymbol{\theta})\right] + \log\left[1-\sigma(r(\mathbf{y}(i);\boldsymbol{\theta}))\right] \tag{5}$$

  where $\mathbf{x}$ is data in class 1 and $\mathbf{y}$ in class 2; classes of equal size

What does logistic regression actually do?

- Fundamental *Theorem*: logistic regression trained with Eq. (5) learns to approximate difference of log-densities:

$$r(\mathbf{u};\boldsymbol{\theta}) \rightarrow \log p_{\mathbf{x}}(\mathbf{u}) - \log p_{\mathbf{y}}(\mathbf{u}) \tag{6}$$

  which becomes exact with infinite sample size and universal approximation capability of function approximator r
- Heuristic proof: re-interpret the log-odds as *log density ratio*

$$\log \frac{p(c=1|\mathbf{z}=\mathbf{u})}{1-p(c=1|\mathbf{z}=\mathbf{u})} = \log p(c=1|\mathbf{z}=\mathbf{u}) - \log p(c=2|\mathbf{z}=\mathbf{u})$$

  since obviously $p(c=2|\mathbf{z}=\mathbf{u}) = 1 - p(c=1|\mathbf{z}=\mathbf{u})$.
- This is what r tries to model to according to definition
- We can manipulate this by using $p(c=C|\mathbf{z}=\mathbf{u}) = p(\mathbf{z}=\mathbf{u}|c=C)p(c=C)/p(\mathbf{z}=\mathbf{u})$
- $p(c=C)$ and $p(\mathbf{z}=\mathbf{u})$ are equal for $C \in \{1,2\}$ and cancel
- The above equals $\log p(\mathbf{z}=\mathbf{u}|c=1) - \log p(\mathbf{z}=\mathbf{u}|c=2)$
- $\log p_{\mathbf{x}}(\mathbf{u}) - \log p_{\mathbf{y}}(\mathbf{u})$ is the same in different notation

So, what does the classifying system do in NCE?

- Recall: NCE is logistic regression with regression function $r(\mathbf{u};\boldsymbol{\theta}) := G(\mathbf{u};\boldsymbol{\theta}) := g(\mathbf{u};\boldsymbol{\theta}) - \log p_{\mathbf{y}}(\mathbf{u})$
- Fundamental NCE Theorem:
  - Assume $g(\mathbf{u};\boldsymbol{\theta})$, i.e. the NN trained in NCE, can approximate any function, and the sample size is infinite
  - Then, the maximum of classification objective is attained when

$$g(\mathbf{u};\boldsymbol{\theta}) = \log p_{\mathbf{x}}(\mathbf{u}) \tag{7}$$

    where $p_{\mathbf{x}}(\mathbf{u})$ is the pdf of the observed data.
- Proof: We just saw that logistic regression learns to approximate difference of log-densities (here: data vs. noise), in the limit:

$$G(\mathbf{u};\boldsymbol{\theta}) = \log p_{\mathbf{x}}(\mathbf{u}) - \log p_{\mathbf{y}}(\mathbf{u}) \tag{8}$$

  So, comparing with the definition of G, we have Eq. (7).

Connection to EBM: NCE learns normalization by itself

- The maximum of objective function is attained when $g(\mathbf{u};\boldsymbol{\theta}) = \log p_{\mathbf{x}}(\mathbf{u})$, and there is *no constraint* on g in this optimization problem!
  - In particular, no normalization constraint (such as $\int \exp(g(\mathbf{u};\boldsymbol{\theta}))d\mathbf{u} = 1$)
- Even if the family $g(\mathbf{u};\boldsymbol{\theta})$ is not normalized (and a NN is usually not), the maximum is still attained for the properly normalized pdf
- In practice, normalization constant (partition function) will be estimated together with the other parameters
- Such a new normalization parameter is learned in a NN with universal approximation
- In non-NN case where $g(\mathbf{x};\boldsymbol{\theta})$ is a model with a limited number of parameters:
  - Add a new parameter c and use $g(\mathbf{u};\boldsymbol{\theta}) + c$
  - Corollary to NCE theorem: If data generated according to model, i.e. $\log p_{\mathbf{x}}(\mathbf{u}) = g(\mathbf{u};\boldsymbol{\theta}^*)$, this gives a *statistically consistent* estimator for $\boldsymbol{\theta}^*$ in unnormalized case.

Estimating parameters vs. estimating density

- Classical estimation theory considers finding *parameters* $\boldsymbol{\theta}$
  - That was the approach in score matching slides
- But with NCE, we focused on estimating the *density*
- In statistics literature, these are two very different problems: Density estimation is called "non-parametric", actually meaning that the number of parameters is infinite (function spaces such as L^2 have an infinite dimension)
- When you learn a NN, clearly the number of parameters is finite, but often huge (practically infinite?)
- In EBM, we can interchangeably talk about estimation of parameters or density, the methods are the same:
  - Using a big NN as $g \approx$ non-parametric estimation (of density)
  - Using a carefully chosen function as $g \approx$ parametric estimation

Choice of noise distribution in NCE

- The noise distribution $p_\mathbf{y}$ is an important design parameter.
- We would like to have $p_\mathbf{y}$ which fullfills the following:
  1. Easy to sample from
     - But we only need to sample noise once, off-line
  2. Has an analytical properly normalized expression
     - But we only need to, e.g., normalize it once
  3. It leads to (e.g.) a small mean-squared error of the estimator
     - This can be analyzed, but not simple to find good distribution
- Intuitively, noise should be rather similar to data, so that classification not too easy
- In practice, we can take Gaussian noise with the same mean and covariance as the data.
  - Far from optimal, but simple

Comparison between different estimators

- Given two estimators for the same model (e.g. SM and NCE), how can we say one of them is better?
- We have two kinds of properties to compare
  - Statistical performance
  - Computation
- Statistics: Estimation theory compares estimators e.g. by
  1. Consistency: Does the estimator converge to the true $\boldsymbol{\theta}^*$ with infinite sample size?
  2. Mean squared error (MSE): $\mathrm{E}\{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\|^2\}$ where expectation is taken over sample size which is fixed.
  - We prefer a consistent estimator over an inconsistent one
  - We prefer estimator with smaller MSE
- Computation: Typically only computation time considered
  - Could consider energy consumption, memory requirements, etc.
- We see that there are many properties to compare

$\rightarrow$ Usually, some kind of trade-off, arbitrary choices necessary

General comparison between score matching and NCE

- Consider a model $p(\mathbf{x}; \boldsymbol{\theta})$, and its estimation by SM or NCE

Statistics

- Both NCE and basic SM are consistent
- Denoising SM "consistently" estimates noisy pdf, not original
- MSE of NCE depends heavily on choice of noise distribution
- (In denoising SM, must choose noise variance but this is usually easy to optimize by trying out different values.)

Computation

- In a NN, basic SM algebraically/computationally difficult
- Denoising score matching works well (empirically) with NN's
- Not much difference between NCE and DSM

... How about maximum likelihood? Statistically the best, but computationally worst for EBM (needs numerical integration)

Validation and comparison of statistical models

- Above, we compared different estimation principles for a single model
- We might also want to compare two models
  using same estimation principle (and same data)

→ Possible by comparing the values of the objective function for the two models (e.g. better fit to data score function)
  - Importantly, this should be on separate (held-out) test data

- One problem: we don't really know the baseline
  - What kind of value for objective is good at all? Difficult to say.
  - This is a problem even with MLE:
    no clue what value of likelihood is "good" ("enough")
  - So, difficult to say when a statistical model is good or bad
    - In contrast to classification with two classes: it is intuitively clear that 52% accuracy is not good, while 95% accuracy is very good

Different ingredients in probabilistic modelling

- In early lectures, DL was characterized as
  1. Objective function J, based on learning principle
  2. NN as function approximator ($\mathbf{g}$) with specified architecture
  3. Optimization algorithm applied on J
- We have now seen how the construction of objective function happens in EBM. We choose:
  - A Estimation principle e.g. MLE, score matching, NCE, etc.
  - B Architecture, which equivalently defines *Statistical model*
    - since we define e.g. $\log p(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{g}(\mathbf{x}; \boldsymbol{\theta})$ in DL case
- These two together give the *Objective function J*
- We still need
  - C Optimization algorithm applied on J
- This logic applies also to latent variable models (next lecture)

Conclusion (of EBM parts 1 and 2)

- We saw three methods for estimating parameters in unnormalized (energy-based) models; MLE being too difficult
- These methods avoided the very expensive numerical integration needed to compute the normalization constant
- In score matching, match gradients of log-densities
  - normalization constant is completely avoided
- In denoising score matching, train autoencoder to reconstruct data from noisy samples
  - computationally simpler than basic SM in an NN
- In noise-contrastive estimation, learn logistic regression to discriminate data from artificial noise
  - normalization constant can be estimated like any parameter
- Increasingly used in deep learning as general density approximators
- Basis of state-of-the-art image generation methods (see next lecture)