

Machine Learning Assignment 11-313652008

313652008 黃睿帆

November 15, 2025

Assignment introduction

In this assignment, we try to answer some "unanswered" questions in our assignment 1 to assignment 10. (Some answers of the following questions were mentioned in the class, and some were generated by Chat-GPT, and some might be discussed in the thesis or article.)

Unanswered questions in assignment 01

- When using **Gradient Descent Algorithm**, how to deal with the situation that f is NOT differentiable (i.e. ∇f cannot be calculated)
- In supervised learning, we commonly use mini-batch gradient descent, and choosing $m < N$, is there the best m to choose?

Answer:

- When f is Not differentiable and f is a convex function, the gradient descent algorithm also worked for such functions by using a so-called sub-gradient (次梯度) of $f(w)$ at $w(t)$, instead of the gradient. In section 14.2 of [1], it claims that **If f is differentiable at w then $\partial f(w)$ contains a single element the gradient of f at w , $\nabla f(w)$.** for example, the sub-gradient of absolute function $f(x) = |x|$, is $\partial f(x) = [-1, 1]$. 接著, 任何一個 subgradient 都能拿來做更新： $x_{k+1} = x_k + \eta g_k, g \in \partial f(x)$
- Unfortunately, there is no universal best for batch size m in mini-batch GD. But common batch size are 32, 64, 128, 256, 512, ..., some articles ([2]) used typically these numbers as batch size. Or recalled that in our class, we used $m = 10$ for $N = 100$.

Unanswered questions in assignment 02

- In supervised learning, we often using classification when target has discrete value, what is the common loss function $L(\theta)$ in classification ? (We mentioned in class that MSE loss is uncommon in classification)
- In supervised learning, can we use classification when target has discontinuous value ? Why in this case we often using regression instead of classification ?

Answer:

The most common loss function used in classification is the **cross-entropy loss**, which arises from the maximum-likelihood formulation of probabilistic classifiers, which we discussed in the class.

MSE is uncommon since MSE assumes a continuous numeric target and does not match the probabilistic nature of classification. Cross-entropy provides better optimization properties and aligns with likelihood-based training. (因為在 MSE 中假設 label 是連續數值，但分類目標是機率分布。Cross entropy 來自 maximum likelihood，與機率模型一致。)

The most common loss function (常用的 Loss function 如下) used in classification is the **cross-entropy loss**, which arises from the maximum-likelihood formulation of probabilistic classifiers. (交叉

熵來自最大概似估計 (Maximum Likelihood)，因此在統計與機率上是自然的選擇。對神經網路來說，交叉熵能提供更「敏感」的梯度，讓訓練收斂更快。而且搭配 softmax (多類) 或 sigmoid (二類) 非常常見 (Chapter 4 of [3])。MSE 在此 Case 下不常是因為：MSE 把類別當成連續數值，但類別其實是「機率分布」問題，不是數值大小問題。而且梯度很小，容易造成訓練緩慢。因此分類問題大多使用 cross-entropy 而不是 MSE。)

Binary classification. For labels $y \in \{0, 1\}$, the binary cross-entropy loss is

$$L(\theta) = -[y \log \hat{p}_\theta(x) + (1 - y) \log (1 - \hat{p}_\theta(x))],$$

where $\hat{p}_\theta(x)$ is the predicted probability of class 1.

Multiclass classification. Let $\hat{p}_\theta(y = k | x)$ be the softmax probability of class k . The multiclass cross-entropy loss is

$$L(\theta) = -\sum_{k=1}^K \mathbf{1}(y = k) \log (\hat{p}_\theta(y = k | x)).$$

For the other question:

Can classification be used? Yes. If the target takes values in a *finite set*, one may formulate the problem as a multiclass classification task. However, this is usually not the most suitable approach. (只要目標有有限個可能值，可以把它當成某種程度的分類問題。)

Why regression is often preferred. (為何實務上還是使用回歸, Chapter 3 of [3].)

1. Numerical distances between targets matter. Regression minimizes a loss such as

$$L(\theta) = |y - f_\theta(x)| \quad \text{or} \quad L(\theta) = (y - f_\theta(x))^2,$$

which reflects how close the prediction is to the true value. Classification, in contrast, treats all misclassifications equally and ignores the numerical structure. (數值之間的「距離」有意義，回歸會找最小化數字之間的差距。分類則是全部錯都算一樣的錯。例如把 0 預測成 3.1415 和預測成 10，在分類中都是「錯誤」，沒有差別。)

2. Regression can generalize to unseen values. If the target arises from an underlying continuous process—even if the outputs appear discontinuous—regression models can predict values not seen in training. Classification models are restricted to a fixed set of labels. (回歸可以預測「沒看過的新值」：若目標其實來自某個（可能不連續的）物理或自然過程的輸出：回歸可以輸出訓練集中未出現過的新值。分類只能輸出固定類別，受限比較多。)

3. Discontinuous targets often come from piecewise functions. For targets generated by a piecewise function (hence discontinuous), regression can still model the underlying structure. Classification ignores the numeric meaning of the outputs and treats each value as an unrelated category. (用 regression 比較自然，可以更好地反映其數值意義，不連續值通常是分段函數的產物)

Example. Consider

$$y = \begin{cases} 0, & x < 0, \\ 5, & x \geq 0. \end{cases}$$

Although this can be treated as a two-class problem, the value 5 may have numeric meaning. Regression preserves this structure, whereas classification discards it.

Unanswered questions in assignment 03

- In the paper we discussed in the class (**On the approximation of functions by tanh neural networks**) ([4]), we drew the neural network. If a network has more layer and more neuron, will it be better, that is, has better accuracy and less loss? (神經網路層數越多、神經元越多，一定比較好嗎？)

Answer:

不一定。更多 layer/neuron 的優點在於更強的表示能力由 Universal Approximation Theorem，可逼近任何連續函數。但同時也存在著缺點，例如：overfitting (模型過大、資料不足), vanishing/exploding gradients (訓練不穩) 等因素使得計算成本上升並具有 generalization gap (模型太複雜反而泛化更差)。

結論：深度網路具更高 approximation power. 但實際 accuracy 也同時取決於數據、正則化、訓練方式、架構。所以：更大不一定更好。通常會使用 cross-validation 找最佳模型大小。(In [5], it also asked similar question.)

Unanswered questions in assignment 04

- When using regression model to approximate periodic functions, will **the Fourier Series** give the best approximation?

Answer:

Yes. If the target function is periodic and sufficiently smooth, the **Fourier Series** is theoretically the optimal approximation in the L^2 sense. (P.S. This semester I take the course: "Fourier Analysis, that's why I asked this question." So some references come lecture notes.)

Specifically, among all functions with the same frequency basis, the truncated Fourier series provides the **best L^2 approximation by the projection theorem. (如果目標函數是週期性的且夠光滑，那麼理論上，傅立葉級數是 L^2 意義下的最優逼近。因為在所有具有相同頻率基底的函數中，根據投影定理，截斷 (Truncated) 傅立葉級數提供了最佳的 L^2 逼近。)

However, in practice: Neural networks do not automatically find Fourier components unless constrained to do so. A standard MLP with ReLU or $tanh$ can approximate periodic functions but may require many neurons because it must "learn" sinusoidal patterns indirectly. (但跑神經網路時，不見得一定會自動找到傅立葉分量。而使用 ReLU 或 $tanh$ 激活函數的標準多層感知器 (MLP) 雖然可以逼近週期函數，但由於它必須間接地「學習」正弦模式的函數，因此還是需要大量的神經元。)

Unanswered questions in assignment 05

- In generative model, we assume, in the class, that $p(x|y = 1), p(x|y = 0)$ are Gaussian, and $p(y = 1), p(y = 0)$ are Bernoulli, is it possible for other cases? (Such as Poisson, exponential, uniformly distribution, etc.), and may we assume all components in $p(x|y)$ are from same distribution?

Answer:

It is absolutely possible to use distributions other than Gaussian. The Gaussian—Bernoulli assumption is the most convenient, that is, the easiest to analyze.

For different choices of $p(x|y)$, it depends on the nature of the data, for instance, Poisson for count data; Exponential / Gamma for positive-valued data; Laplace for heavy-tailed features; Categorical for discrete features; Multinomial for word counts.

For another questions, all components may follow the same distribution. This is exactly the Naive Bayes assumption (see 3.5 of [8] and Chapter 2 of [9]):

$$p(x|y) = \prod_{i=1}^d p(x_i|y).$$

To sum up, we can choose any likelihood model as long as: the density is valid, gradients or closed-form posteriors can be computed, and the model reflects the statistical properties of the data. Gaussian Naive Bayes is common, but Poisson Naive Bayes and Multinomial Naive Bayes are widely used (e.g., for text classification. See [10]). (只要滿足密度有效，可計算梯度或後驗機率，且模型能夠反映資料的統計特性，就是可以選擇的機率模型。)

Unanswered questions in assignment 06

- Continuing the Unanswered questions in assignment 04. How complicated that the neuron network of Fourier Series looks like ?

Answer:

If we explicitly convert a Fourier series into a neural network, the network becomes simple: A truncated Fourier series:

$$f(x) = \sum_{k=0}^K a_k \cos(kx) + b_k \sin(kx),$$

it can be written as a 1-hidden-layer network: Hidden layer uses fixed activations $\sin(kx)$, $\cos(kx)$.

Output layer is linear:

Formally:

$$f(x) = W_2\sigma(W_1x),$$

where σ contains basis functions $\sin(kx)$, $\cos(kx)$.

This network is extremely shallow and has only $2K$ hidden units.

But is the Fourier series always the best ? In fact, it is best in L^2 sense for periodic smooth functions. But Not best for highly non-smooth targets (Gibbs phenomenon.(Gibbs 現象: 不連續點處的震盪現象)), and Neural networks can outperform Fourier for local features because Fourier is global.

Unanswered questions in assignment 07

- How does the choice of noise scales (the σ -schedule) and the per-scale weighting in the DSM loss affect consistency and sampling quality?
- What happens to DSM-trained score estimators in low-density regions and near the data manifold —can the estimated score blow up or be poorly behaved?

Answer:

In Denoising Score Matching (DSM) and diffusion models, the choice of noise scales σ is crucial. Effects of the σ -schedule are:

For too small σ such that model must learn extremely sharp score fields near data manifold, it gives unstable gradients, hence poor generalization; For too large σ , such that the model only learns coarse structures, the samples will be blurry or over-smoothed. A good schedules is geometric or cosine. It provides multi-scale structure → stable training → high-quality samples. (如果 σ 太小，會導致模型梯度不穩定，從而泛化能力差。如果 σ 過大，模型只能學習粗略的結構，樣本會變得模糊或過度平滑。幾何或餘弦調度是較好的選擇。它們能夠提供多尺度結構，穩定的訓練，和高品質的樣本。See [11] and [12])

Effects of per-scale weighting in DSM, Recall the DSM objective is:

$$\mathcal{L} = \mathbb{E}_{x,\sigma} w(\sigma) \|s * \theta(x + \sigma\epsilon, \sigma) + \frac{\epsilon}{\sigma}\|^2.$$

Choice of $w(\sigma)$ impacts consistency (Proper weighting ensures that the score approximates the true gradient of $\log p_\sigma(x)$), and sampling quality if small- σ terms are overweighted, it causes instability; if large- σ terms are overweighted, will make samples lack fine detail. Empirically, schedules such as VP, VE, or cosine noise schedules give the best results. ($w(\sigma)$ 的選擇會影響一致性和採樣質量)

For the other question (See [13] and [14]). This is a known behavior. Near the data manifold density changes rapidly. The true score $\nabla_x \log p_\sigma(x)$ has very large magnitude. Thus the learned score can become extremely large, exhibit numerical instability, and overfit noise in the dataset. (This is why sufficiently large values must be included.) (這是已知現象。在資料流形附近，密度變化迅速。因此，學習到的 Score 可能變得極大，出現數值不穩定，並過度擬合資料集中的雜訊。)

In low-density regions (regions far from data): The density is extremely small, and the score (gradient of log-density) becomes ill-defined or tends toward 0. The DSM estimator becomes dominated by noise

and the learned score field may be inconsistent or even divergent. (在低密度區域：Score (對數密度的梯度) 變得難以定義或趨近於 0。DSM 估計器主要受雜訊影響，學習到的得分場可能不一致甚至發散。)

This is related to the fact that $\log p(x)$ has sharp transitions between data and non-data regions. Since DSM gives no samples from true distribution in sparse regions. The model must infer the score field from noisy samples, so with insufficient noise levels, the score estimator becomes unstable. (This is also why **SDE-based training** with a proper schedule produces smoother, more globally consistent score fields.) (這與 $\log p(x)$ 在資料區域和非資料區域之間存在急劇過渡有關。由於 DSM 在稀疏區域中無法提供來自真實分佈的樣本，模型必須從雜訊樣本推斷得分場，因此在雜訊水準不足的情況下，得分估計器會變得不穩定。)

Unanswered questions in assignment 08

- How does numerical stability differ between deterministic methods (like Euler's method) and stochastic methods (like Euler—Maruyama)?
- Is it possible that an SDE approximation diverge even when the drift and diffusion terms are well-behaved?

Answer:

- Deterministic methods such as Euler's method study numerical stability relative to the underlying ODE, typically through the Dahlquist test equation $y' = \lambda y$. Explicit Euler is conditionally stable and sensitive to stiffness. For SDEs, Euler—Maruyama (EM) must satisfy *mean-square* and *weak* stability. Under global Lipschitz and linear-growth conditions on the drift $b(x, t)$ and diffusion $\sigma(x, t)$, EM converges (strong order 1/2, weak order 1). However, EM may become unstable or diverge for superlinear coefficients even if the true SDE is well-posed. (See [15] and [16]) [Euler 法的穩定性通常用 Dahlquist 測試方程 $y' = \lambda y$ 研究。顯式 Euler 只有在步長小、問題不剛性時才穩定；SDE (Euler—Maruyama)：除了弱誤差、強誤差外還需要考慮均方穩定 (mean-square stability)。在漂移 $b(x, t)$ 和擴散 $\sigma(x, t)$ 滿足全域 Lipschitz + 線性增長條件時，EM 方法強收斂階 $\frac{1}{2}$ ，弱收斂階為 1。若係數成長超線性，即使 SDE 本身良好，EM 也可能不穩定或發散。]
- Yes —numerical approximations (especially explicit ones like Euler—Maruyama) can diverge even when the true SDE has a well-defined unique solution, if the coefficients do not satisfy global Lipschitz and linear growth conditions. Examples show EM may blow up for SDEs with superlinear coefficients. Conversely, if coefficients satisfy global Lipschitz + linear growth, EM is stable and convergent. (See [17])(Hutzenthaler、Jentzen、Kloeden ([17]) 證明：若漂移或擴散具有超線性成長，Euler—Maruyama 會發散 (包括爆掉的期望值)，即使真實 SDE 有唯一強解且矩存在。若係數滿足全域 Lipschitz + 線性增長，則 EM 不會發散。)

Unanswered questions in assignment 10

- How exactly do the probability-flow ODE and the score PDE relate: under what assumptions does the deterministic PF-ODE reproduce the same time-evolving marginals as the SDE, and how does the score function $\nabla_x \log p(x, t)$ enter and determine that equivalence (especially when the diffusion coefficient $g(x, t)$ is state-dependent) ?

Answer:

Relation between the probability-flow ODE and the score PDE; role of the score function. The Fokker–Planck equation may be rewritten as a pure continuity equation. Let $D(x, t) = \sigma\sigma^\top$. Using $\nabla p = p\nabla \log p$,

$$\partial_t p = -\nabla \cdot ((b - \frac{1}{2}D\nabla \log p)p).$$

Thus the *probability-flow velocity* is

$$v(x, t) = b(x, t) - \frac{1}{2}D(x, t)\nabla_x \log p(x, t).$$

The deterministic probability-flow ODE

$$\frac{dx}{dt} = v(x, t)$$

pushes forward the density $p(x, 0)$ to the same marginal $p(x, t)$ as the SDE, provided p is smooth and v is Lipschitz. The term involving $\nabla \log p$ is the *score*, and fully determines the correction needed to convert the diffusive FP operator into a deterministic transport equation. The result holds even when $D(x, t)$ depends on x (state-dependent diffusion)(See [18]). (Fokker—Planck 可寫成純流體連續方程。使用若 $p(x, t)$ 足夠平滑且 v 局部 Lipschitz，則 PF-ODE 推動密度的結果與 SDE 有相同的 marginal distributions。Score $\nabla \log p$ 是將擴散 PDE 轉成純輸運 (ODE) 所需的校正項。)

References

- [1] Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, 2014.
- [2] Keskar et al. On Large-Batch Training for Deep Learning (ICLR 2017).
- [3] Christopher Bishop, Pattern Recognition and Machine Learning.
- [4] On the approximation of functions by tanh neural networks, Tim DeRyck, Samuel Lanthaler, Siddhartha Mishra, Seminar for Applied Mathematics, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland.
- [5] Cybenko, Approximation by superpositions of a sigmoidal function.
- [6] Mallat, A Wavelet Tour of Signal Processing.
- [7] Boyd, Chebyshev and Fourier Spectral Methods.
- [8] Murphy, Machine Learning: A Probabilistic Perspective.
- [9] Bishop, Pattern Recognition and Machine Learning.
- [10] <https://vocus.cc/article/676ba6e4fd89780001aaffd8>.
- [11] Song and Ermon (2019), Denoising Score Matching with Annealed Noise.
- [12] Song et al. (2021), Score-Based Generative Modeling through SDEs.
- [13] Song, Ermon (2019) —DSM theoretical properties.
- [14] Song et al. (2021) —Discussion on score behavior near low-density regions.
- [15] Kloeden, Platen (Numerical Solution of SDEs)
- [16] Higham (2001) survey on numerical methods for SDEs.
- [17] Hutzenthaler, Jentzen, Kloeden (2011/2012) on divergence of EM.
- [18] Song et al., Score-Based Generative Modeling through SDEs (2021).