Classification via score-based generative modelling

Yongchao Huang *

June 2022

Abstract

In this work, we investigated the application of score-based gradient learning in discriminative and generative classification settings. Score function can be used to characterize data distribution as an alternative to density. It can be efficiently learned via score matching, and used to flexibly generate credible samples to enhance discriminative classification quality, to recover density and to build generative classifiers. We analysed the decision theories involving score-based representations, and performed experiments on simulated and real-world datasets, demonstrating its effectiveness in achieving and improving binary classification performance, and robustness to perturbations, particularly in high dimensions and imbalanced situations.

Keywords: Score-based modelling, discriminative classification, generative classification, imbalanced learning.

1 Generative and discriminative approaches to classification

Generally, there are two approaches to probabilistic classification: generative and discriminative [17]. Assume we have in total \( c \) classes, i.e. \( y \in \{y_1, y_2, \ldots, y_c\} \), the generative approach models the class-conditional density \( p(x|y) \) in some functional or architectural form using data, and computes the class probability via the Bayes rule \[23]:

\[
p(y_j|x) = \frac{p(x|y_j)p(y_j)}{\sum_{j=1}^{c} p(x|y_j)p(y_j)}
\]

where \( j = 1, 2, \ldots, c \). \( p(y|x) \) is also termed the posterior probability that an observation \( x \) belongs to the \( j \)-th class, \( p(x|y) \) is the generative model, and

---

*Author email: {yongchao.huang@outlook.com}

1 We generally consider a probability mass function (pmf) or probability density function (pdf), denoted by \( p(x) \), as density. Either maps a scalar or vector \( x \) to a non-negative scalar field, i.e. \( p(x) : \mathbb{R}^n \rightarrow \mathbb{R} \), with the property of summing (or integrating) to unity.
the prior probability of class $j$, which can be empirically estimated as the fraction of training samples attached to the $j$-th class. Typical generative methods include naive bayes, linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). Estimating the class-specific density $p(x|y_j)$ from data is not easy, conventional methods such as kernel density estimation (KDE), or modern ones such as generative adversarial networks (GANs) can be used to explicitly or implicitly represent the sampling process, and generate samples from the learned dynamics.

Discriminative approaches, e.g. logistic regression, support vector machine, and decision trees, directly model the posterior $p(y|x)$, or learn a direct map from inputs $x$ to the class labels [17]. A discriminative neural network classifier, for example, may use the softmax activation $\sigma(x)$ in its output layer to positively transform and squash feature embeddings $f(x)$ to obtain the probability of an observation $x$ belonging to class $j$:

$$p(y_j|x) = \sigma(x) = \frac{e^{f_j(x)}}{\sum_{j=1}^{c} e^{f_j(x)}} \quad (2)$$

The exponential response function (i.e. the inverse of a link function, as used in generalized linear models) gives the positive transform. In binary classification (i.e. $c=2$), softmax reduces to the sigmoid function $\lambda(x)$:

$$p(y=1|x) = \lambda(x) = \frac{e^{f(x)}}{1 + e^{f(x)}} \quad (3)$$

If $f(x)$ is linear, $\lambda(x)$ is termed the logistic response function because it models the logit (i.e. log odds ratio) $p(y=1)/p(y=0)$ as a linear combination of features:

$$f(x) = \omega^T x = \log \frac{p(y=1)}{1 - p(y=1)} \quad (4)$$

where $\omega$ hosts the linear coefficients. Another common choice of response function is the cumulative density function (CDF) of a standard normal distribution $\Phi(z) = \int_{-\infty}^{z} N(x|0,1)dx$, which is termed as probit regression.

There is no certain answer to which approach should be chosen [23]. The generative approach provides a principled way to deal with missing values and outliers via access to $p(x)$, and a generative classifier may approach its (higher) asymptotic error faster [17]. Discriminative methods are straightforward and get around the density estimation problem which could be hard in high dimensions due to curse of dimensionality, and inaccurate with presence of small amount of data. Also, these densities could be complex and multi-modal, making learning $p(x|y_j)$ from data challenging. Imagine we have a density of the form [9]:

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta} \quad (5)$$

where $f_\theta(x)$, parameterized by $\theta$, is an arbitrary function of $x$ (linear or non-linear, e.g. GLMs or NNs), and it’s squashed through the exponent to yield
positive values, and then normalised by the constant \((w.r.t. \ x, \ \text{called evidence in a Bayesian posterior density})\) \(Z_\theta = \int e^{-f_\theta(x)} dx\), which gives the regularity requirement for a proper density to integrate to unity. An example is the one-dimensional (1D) Gaussian distribution where \(f_\theta(x) = (x-\mu)^2/2\sigma^2\), \(Z_\theta = \sqrt{2\pi}\sigma\) with \(\theta = (\mu, \sigma)\).

The density estimation task is to estimate the unknown parameters \(\theta\) in \(p_\theta(x)\). If distributional structure on data is assumed, e.g. samples are Gaussian distributed (as in LDA and QDA, with further assumptions such as covariance homoscedasticity), we are able to empirically obtain from samples an educated guess of the population summary statistics (e.g. mean and variance) as functions of \(\theta\). These assumptions, however, are strong and may lead to poor classifier performance when they are violated. If we have i.i.d samples, we can maximize the data likelihood or minimize some distance metric (e.g. KL divergence) to find \(\theta\). However, likelihood-based methods normally require exact computation of the density, which requires knowledge about the normalising constant that may also depend on \(\theta\), and in many cases this is intractable. Maximum likelihood estimation (MLE), for example, finds \(\theta\) by maximizing the overall likelihood (assuming we have in total \(N\) i.i.d samples):

\[
\hat{\theta} = \arg \max_{\theta} L(\theta, x) = \arg \max_{\theta} \prod_{i=1}^{N} p_\theta(x_i) \quad (6)
\]
or equivalently maximizing the sum of log likelihoods:

\[
\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^{N} \log p_\theta(x_i) \quad (6')
\]

where \(L(\theta, x)\) is the likelihood function. \(\hat{\theta}\) can be found theoretically by solving the necessary optimal equation \(\partial L(\theta, x)/\partial \theta = 0\). As the normalising constant \(Z_\theta\) may also be a function of \(\theta\), problem arises when exact likelihood computation is not feasible, i.e. we can only evaluate the density from data up to a multiplicative constant. Eq.5 as an example, very often we have only knowledge about \(f_\theta(x)\) but not \(Z_\theta\). In many cases, particularly high dimensions, the normalising constant \(Z_\theta\) cannot be analytically obtained, e.g. due to intractable or expensive integration. We could, of course, resort to numerical evaluation methods such as Monte-Carlo (MC), which may further raise questions about efficiency, accuracy and convergence. Estimating the gradient of log-density could bypass this.

In the following text, we first introduce the score function as an alternative characterization and learning objective of data, accompanied by the gradient learning method score matching. This learned function can then be used as a generator in sampling to synthesize samples to assist other discriminative classifiers (e.g. nearest neighbours, tree-based, neural network, etc), particularly

\[^2\text{A hat (crown) over a variable denotes estimated value.}\]
\[^3\text{Nearest neighbour classifiers can be generative if class conditional densities are learned, e.g. via KDE [21].}\]
in imbalanced data scenario. We call this type score-assisted\(^4\) discriminative classification. Either, the learned score function can be used to construct the generative density \(p(x|y_j)\) conditioned on an initial (e.g. empirically estimated) density value, which can be plugged into a generative classifier. We call this type score-based generative classification. We focus on binary classification task and test both methods using simulated\(^5\) and real-world data.

## 2 Score-based generative modelling

### Score-based representation of data distribution

Instead of directly estimating the pdf, which may pose challenge in evaluating the normalisation constant, we could estimate the gradient of the log density which is defined as the score function \(s_\theta(x)\)\(^9\):

\[
s_\theta(x) = \nabla_x \log p_\theta(x)
\]  

\(7\)

The score function has same dimension input and output; the resulted score field is a conservative, irrotational vector field (e.g. vanishing curl in three-dimensional Euclidean space), over which line integral is path independent. If the density \(p_\theta(x)\) is in the specific form of Eq.5 we have:

\[
s_\theta(x) = -\nabla_x f_\theta(x) - \nabla_x \log Z_\theta = -\nabla_x f_\theta(x) 
\]  

\(7b\)

Figure 1: Densities and score functions of two 1D Gaussian distributions.

The second term vanishes as the normalising constant \(Z_\theta\) is independent of \(x\). \(s_\theta(x)\) is still a function of \(x\), parameterized by \(\theta\), with the normalisation constant removed, which is desirable in computation. A score function can be used to characterize a random variable without losing information; it can be parameterized explicitly by any admissible formula, or represented by a composite architecture such as neural network. It admits a scalar-valued or vector-valued \(x\) and outputs same-dimension scalar or vector gradient field.

---

\(^4\)score-assisted” is used mostly when score methods are used indirectly, e.g. in data augmentation, while ‘score-based’ is used when directly involved, e.g. in generative classification.

\(^5\)The term ‘simulated’ is used in the scene of new data generation, while ‘synthesized’ or ‘synthetic’ data is employed in the context of augmentation of original samples.
As an example, the score function, \( s(x) = (\mu - x)/\sigma^2 \), of two Gaussian densities with \( \mu_0 = -2, \sigma_0 = 1 \) and \( \mu_1 = 2, \sigma_1 = 1 \), are graphically shown in Fig. 1.

For a general multivariate Gaussian variable \( x \sim \mathcal{N}(\mu, \Sigma) \) with density:

\[
p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right] \tag{8}
\]

its score function \( s(x|\mu, \Sigma) = \Sigma^{-1}(\mu - x) \). A two-dimensional example with \( s_{\theta}(x) : \mathbb{R}^2 \to \mathbb{R}^2 \) and \( \mu = (0, 0), \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \) is shown in Fig. 2.

Figure 2: Density and score field of a 2D Gaussian distribution.

The task of generative modelling is to learn a generative model, e.g. \( p_\theta(x) \) or \( s_\theta(x) \), from data and to generate new samples using the learned model (Fig. 3). The key lies in learning an approximate, sample-based \( s_\theta(x) \). Suppose we have i.i.d samples \( \{x_i \in \mathbb{R}^d\}_{i=1}^N \) drawn from an unknown distribution \( p_D(x) \) supported by \( \chi \), and we want to learn a score function \( s_\theta(x) : \mathbb{R}^d \to \mathbb{R}^d \) from data which approximates \( p_D(x) \) [9]. If we know the density \( p_D(x) \) of the data generating process (DGP), we can of course calculate the population-based scores \( \nabla_x \log p_D(x_i), i = 1, 2, ..., N \), and estimate \( \theta \) by minimizing a certain distributional distance (e.g. KL Divergence) between \( \nabla_x \log p_D(x) \) and \( s_\theta(x) \). By doing this we are essentially performing functional approximation. However, we don’t have access to \( p_D(x) \); in fact, that’s the density we are interested in estimating or approximating. Score matching [15] solves this: it doesn’t require access to \( p_D(x) \) but optimizes \( \theta \) purely based on data.

**Score matching (SM)** Hyvärinen [15] proposed that, non-normalized statistical models can be estimated by minimizing the expected squared distance between the gradient of the log-density given by the model and that of data. Formally, the objective is to find the parameters \( \theta \) that minimize the implicit Fisher Divergence between the unknown data density \( p_D \) and approximate density \( p_\theta(x) \) [9]:

\[
\theta = \arg \min_{\theta} D_F(p_D || p_\theta) \tag{9}
\]

with
Figure 3: Samples and score generators.

\[ D_F(p_D \parallel p_\theta) = \frac{1}{2} \int_\chi p_D(x) \| \nabla_x \log p_D(x) - \nabla_x \log p_\theta(x) \|^2 dx \]

\[ = \frac{1}{2} \mathbb{E}_{p_D}[\| \nabla_x \log p_D(x) - s_\theta(x) \|^2] \]

(10)

The normalising constants in \( p_D \) and \( p_\theta \) are eliminated when taking derivative, it’s thus irrelevant during score matching. Under mild conditions such as differentiability \[7\], the implicit Fisher Divergence objective can be transformed into its explicit form which can be conveniently estimated via sampling routines \[7, 9\]:

\[ D_F(p_D \parallel p_\theta) = \int_\chi p_D(x) \sum_{i=1}^d \frac{1}{2} \left( \frac{\partial \log p_\theta(x)}{\partial x_i} \right)^2 + \left( \frac{\partial^2 \log p_\theta(x)}{\partial x_i^2} \right)^2 dx + C \]

\[ \mathbb{E}_{p_D}[\frac{1}{2} || s_\theta(x) ||^2 + tr(\nabla_x s_\theta(x))] + C \]

(10b)

where \( d \) is the dimension of \( x \), \( C \) is a constant. It is proved that \[7\], if \( p_\theta(x) > 0 \) for all \( x \in \chi \), \( D_F(p_\theta \parallel p_\theta) \leftrightarrow p_\theta = p_\theta \).

When computing the explicit Fisher Divergence in SM, substantial cost is induced by the Hessian term. Sliced Score Matching (SSM), which measures the divergence over random projections instead, is proposed \[8, 7\]:

\[ D_{FS}(p_D \| p_\theta) = \frac{1}{2} \int_v q(v) \int_\chi p_D(x) [v^T \nabla_x \log p_D(x) - v^T \nabla_x \log p_\theta(x)]^2 dx dv \]

(11)

where \( v \in \mathcal{V} \subseteq \mathbb{R}^d \) is a noise vector distributed as \( q(v) \). Similar to SM, Eq.\[11\] can be further explicitly developed as \[7\]:

\[ \text{An example implementation can be found at e.g. [3]} \]

\[ \text{Capital C is used in multiple scenes across this work, e.g. later in integrating the score function to recover density. Its meaning should be clear from context.} \]
\[ D_{FS}(p_D||p_\theta) = \int \chi p_D(x) \left\{ \int q(v) [v^T (H_x \log p_\theta(x))v] dv + \frac{1}{2} \| \nabla_x \log p_\theta(x) \|^2 \right\} dx + C \]

\[ = E_{p_D} E_q [v^T \nabla_x s_\theta(x) v + \frac{1}{2} \| s_\theta(x) \|^2] + C \]

where \( H_x \) is the Hessian matrix w.r.t. coordinate \( x \).

Gradient learning approaches such as SM or SSM don’t require evaluating the normalizing constant; we can use any parameterized formula or architecture to represent \( s_\theta(x) \). In this work, we employ a neural network regressor as our score model and train it using Eq.\( 10b \).

**Constructing density from scores** Theoretically, once a score function which provides gradient information about the log density everywhere has been learned, we can recover the density:

\[ p_\theta(x) = C e^{\int x s_\theta(x) dx} \]

where \( \chi \) is the set supporting \( x \), \( C \) can determined by any initial condition. Depending on the dimension of \( x \), the integral could be over a scalar field or a vector filed (e.g. line integral). With examples of Gaussian densities, we start with one-dimensional \( x \), recovering the scalar-valued \( p(x) \) from scalar-valued \( s_\theta(x) \); then move to two dimensions to recover the scalar field \( p(x) \) from a gradient vector field \( s_\theta(x) \) via line integration. For writing convenience, we drop the subscript parameter \( \theta \) wherever it’s clear from the context (e.g. when derivations are parallel for both classes), assuming all functions are parameterized by default (e.g. weights of neural network when representing a score function).

Empirically if we know one point density \( p(x_0) \), starting from it we can incrementally construct the density curve (or surface) by visiting many grid points. For example,

\[ \log p(x_0 + \delta x) - \log p(x_0) = \log \frac{p(x_0 + \delta x)}{p(x_0)} = \int_{x_0}^{x_0+\delta x} s(x) dx \]

For one-dimensional \( x \), \( s(x) \) is also scalar-valued. The integral can be conveniently estimated by numerical integration routines such as Monte Carlo, i.e. \( \int_{x_0}^{x_0+\delta x} s(x) dx \approx \frac{\delta x}{N} \sum_{k=1}^{N} s(x_k) \), where \( x_k \) are sampled from the interval \( [x_0, x_0 + \delta x] \). Essentially, the density at \( x_0 + \delta x \) can be approximated as:

\[ p(x_0 + \delta x) = p(x_0) \times e^{\frac{\delta x}{N} \sum_{k=1}^{N} s(x_k)} \text{, where } x_k \in [x_0, x_0 + \delta x] \]

Alternatively, we can use first-order Taylor expansion to approximate the difference, if the step \( \delta x \) is small:

\[ \log p(x_0 + \delta x) - \log p(x_0) \approx \frac{d \log p(x)}{dx} \bigg|_{x=x_0} \delta x = s(x_0) \delta x \]
The difference between Eq.14 and Eq.15 is that, Monte-Carlo method samples a number of points within the interval, either in a random manner or taking into account the shape of \( s(x) \), and average their scores, while Taylor approximation uses one point. There are other integration approximation methods, e.g. trapezoidal rule, Simpson’s rule or Gaussian quadrature, that can lead to more accurate computation. Here we take the MC view.

As an example, the densities in Fig.1 are reconstructed in Fig.4 from the learned score fields. The density of Class 0 is better recovered with a smaller Jensen–Shannon divergence (JSD, a symmetric measure of dis-similarity between two distributions) value than Class 1; the difference could have been induced by randomness in samples distribution (e.g. different patterns of sample scarcity at far ends). While the score learning is not perfect, we could improve the learning if we make additional assumptions, e.g. assuming the two clusters of observations are Gaussian distributed, this limits our selection of score architecture to be within the space of linear functions (since we know the score function of a 1D Gaussian density is \( s(x) = (\mu - x)/\sigma \), however, there is no reason to prevent ourselves from using more complex functions which might overfit in score matching). This choice of simple score function enables deriving potential analytical closed form of \( f(x) = \int \chi s(x)dx + C \) where \( C \) can be determined by some initial condition (e.g. initial probability at a known point, free to choose the cluster center where \( p(\mu) = 1/\sqrt{2\pi} \hat{\sigma} \), which could further leads to expressive form of a estimated density. Imposing this extra assumption is discussed in Section 4 and results deployed in Fig.23.

![Figure 4: Recovered densities using learned score function. There are 1000 samples for each class, with locations shown at bottom. The score functions use an MLP with size [1, 128, 256, 128, 1] and ReLU activations. Initial probability \( p_{\theta_0}(\mu_0 = [-2.01]) = 0.40, p_{\theta_1}(\mu_1 = [2.04]) = 0.40 \).](image)

Same principles can be applied in high dimensions to recover the vector-input, scalar-output pdf, from a vector-valued input and output score function \( s(x) \). The difference lies in the way we do the integration in Eq.12 for a vector field, the integration in the exponent is interpreted as line integral:

\[
\log \frac{p(x_0 + \delta x)}{p(x_0)} = \int_{t_0}^{t_0 + \delta t} s(x(t)) \cdot x'(t) dt
\]

(16)
where $|\cdot|$ denotes inner product. $l : t_0 \rightarrow t_0 + \delta t$ is a directed, piecewise smooth curve parameterized by $t$, starting at $x_0$ and ending $x_0 + \delta x$ with correspondences $x(t_0) = x_0$ and $x(t_0 + \delta t) = x_0 + \delta x$. Line integrals over the score vector fields reply on the direction of $x(t)$ but are independent of the parametrization $x(t)$ in absolute value. By definition, the line integral in Eq.\ref{eq:10} can be calculated as a Riemann sum of the dot product of the tangent $x'(x)$ and the vector field direction $s[x(t)]$, with infinitesimal partitions over $[t_0, t_0 + \delta t]$:

$$
\int_{t_0}^{t_0+\delta t} s[x(t)] x'(t) dt = \lim_{\Delta t \rightarrow 0} \sum_{k=1}^{N} s[x(t_k)] \cdot x'(t_k) \Delta t \quad (17)
$$

with $[t_1 = t_0, t_N = t_0 + \delta t]$ and increment $\Delta t$. As per its definition Eq.\ref{eq:7}, the score function $s(x)$ is the gradient of a scalar field $p(x)$, i.e. $s(x)$ is a conservative vector field, its line integral is path independent, i.e. while there are infinitely many paths between $[t_0, t_0 + \delta t]$, we can choose the simplest path, i.e. the straight line connecting the starting and end points, then the tangent $x'(t_k) = (x(t_k) - x(t_{k-1})/\Delta t$. Further, when $\Delta t$ is sufficiently small, constant score field (e.g. using central point score) can be assumed within $\Delta t$. Together, we have following simplification for contrasting two densities:

$$
\log \frac{p(x_0 + \delta x)}{p(x_0)} = \lim_{\Delta t \rightarrow 0} \sum_{k=2}^{N} \frac{s[x(t_k)] + s[x(t_{k-1})]}{2} \cdot [x(t_k) - x(t_{k-1})] \quad (18)
$$

An exaggerated illustration of Eq.\ref{eq:18} with one large step contribution, is shown in Fig.\ref{fig:5}. In practice, we start from an initial point $[x_0, p(x_0)]$ with $p(x_0)$ estimated from data (e.g. sampling from the score function using Langevin dynamics\cite{Langevin}), count the frequencies in the neighbourhood of $x_0$ and compute an estimated density), choose a next point $x_0 + \Delta x$, measure the alignment between local score field direction and the trajectory direction, and assign the new point a probability proportional to the exponential cosine similarity. We repeat this incrementally until reaching the target point $x_0 + \Delta x$, at which point we calculate the final probability as the accumulated contribution from each segment of the trajectory. The whole density surface can be constructed by wandering through the feature space. Note that, the density is constructed with only knowledge of the score function and an initial probability guess; it doesn’t require any distributional assumption of data.

As an example, we simulated two Gaussian 2D clusters (Fig.\ref{fig:6}), each with 200 samples labeled 0/1, with the underlying DGP$s: \mu_0 = (0, 0), \Sigma_0 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}, \mu_1 = (4, 4), \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. Two score functions are trained on the simulated samples separately; they are then used to construct the density surfaces in Fig.\ref{fig:7}(a), using estimated probabilities $p_{\theta_0}(\hat{\mu}_0 = [0.07, -0.12]) = 0.18$ and

---

\cite{Langevin} Introduced later in Section 3.2.
Figure 5: Illustration of the path independent property of line integral. Light blue colored area implies high density arena of Class 0; blue arrows suggest score field. $x_0$ is the starting point, $\Delta x$ the step size. The two paths indicated by black solid arrow and dashed arrows yield the same integration value due to the path independent property.

$p_{\hat{u}_1}(\hat{\mu}_1 = [4.06, 4.18]) = 0.18$. These estimates of initial probabilities can be obtained either by sampling from the learned score function and counting class appearances in a specified $\sigma$-neighbourhood, or using $p(\hat{\mu}) = 1/(2\pi)^{d/2}|\Sigma|^{1/2}$ if extra Gaussian distribution of data is assumed. It is seen that, the score-recovered densities are smooth and similar to the original ones, except that blue class density is fatter than its DGP density, which might be induced by inaccurate estimations of initial values and score functions. Shown in Fig[7](b) is the projected 2D densities and equal density boundaries (which is used later in decision theory). It is observed that, the score-recovered densities co-locate well with theoretical contours; the empirical boundary (white dots) doesn’t align fully with the theoretical boundary (green dots) at far ends, this may be a direct result of the excessive power brought by the heavy tail exerted by the over-fat blue class density, or numerical errors at low density areas, as well as incapable extrapolation of the learned score functions in sparse data regions. An improved score-recovered density, with additional assumption of Gaussian distribution of data, is presented in Fig[24].

We could generalize the 1D and 2D (and high-dimensional) cases. Eq[14] and Eq[18] are intrinsically the same, i.e. if we divide the interval $\delta x$ into $N$ equidistant intervals $x(t_k) - x(t_{k-1})$ and evaluate $s(x)$ it the middle point. They differ in how we process the product $\delta x \times s(x)$ in the exponent: for one-dimensional $x$ it’s scalar product, for $N$-dimensional problem it becomes inner product. We summarize the density construction procedure in Algorithm[1].

**Score smoothing** Depending on the parametric form or architecture used to represent a score function, sometimes the learned score surface can be discontinuous. In such cases, we can use for example kernel smoothing technique to smooth the estimated function in a post-estimation stage. A simple strategy is to use an identity kernel, i.e. taking simple average of neighboring scores. This post-estimation step equivalently adds locally smooth constraint to the score
Figure 6: 2D Gaussian DGPs and simulated samples. In both figures, dots represent samples; on the right, green line marks the boundary with equal density.

**Algorithm 1:** Score-based density construction

1 **Initialisation.**
   - obtain an initial probability at arbitrary point \( x_0 \).
   
   e.g. \( p(x_0 = 0) = \frac{1}{\sqrt{2\pi}\hat{\sigma}} e^{-\frac{\hat{\mu}^2}{2\hat{\sigma}^2}} \) for 1D Gaussian data, with \( \hat{\mu} \) and \( \hat{\sigma} \) estimated from data;

2 **Learning a score function** \( s(x) \) via score matching.

3 **Start from an initial point** \([x_0, p(x_0)]\), **visit all desired points** \([x_0 + \delta x, p(x_0 + \delta x)]\) **by varying** \( \delta x \). To calculate \( p(x_0 + \delta x) \):
   - Monte-Carlo sample \( N \) points (or with fixed step size) from \([x_0, x_0 + \delta x]\).
   - compute \( p(x_0 + \delta x) \) using Eq.14

function, which may be useful in sparse data region where limited information can be learned.

Given the coordinates of an arbitrary point \( x^+ \), we query the learned score functions to output scores at surrounding points and compute their average (or weighted average) as the central point score. This has been used in constructing the pdf, where we sample the integration interval \( \delta t \) and take average in Eq.[14]

This can further extended to high dimensions where samples are taken from a neighbourhood space:

\[
\int_{x_0}^{x_0+\delta x} s(x)dx \approx \frac{V}{N} \sum_{i=1}^{N} s_{\theta_1}(x_i), \text{ where } x_i \in \{x_i : dis(x_i, x^+) \leq \sigma\} \tag{19}
\]

where \( x_i \) are drawn from the \( \sigma \)-neighbourhood of \( x^+ \), \( V \) is the volume of the
Figure 7: Recovered densities and decision boundary using learned score function. There are 200 samples for each class. The score functions use a MLP with size \([2, 128, 256, 128, 2]\) and ReLU activation. Initial probability \(p_{\theta_0}(\mu_0 = [0.07, -0.12]) = 0.18, p_{\theta_1}(\mu_1 = [4.06, 4.18]) = 0.18\). On the right, brighter color implies higher recovered density, contour and green dots represent DGP-based (theoretical), white dots indicate the equal density boundary generated by recovered densities.

\(\sigma\)-neighbourhood, \(\text{dis}(\cdot, x^+)\) is a distance metric associated with the space. Geometry-aware sampling methods (e.g. importance sampling) can be used to sample neighborhood points. Eq.19 inspires the use of a patch camera centered at \(x^+\) with radius (or side length) \(\sigma\): we take \(N\) points surrounding \(x^+\), substitute their scores into Eq.19 once \(\sigma\) becomes in infinitesimal while \(N\) becomes large, the approximation would converge as per law of large numbers.

3 Score-assisted discriminative classification

We discuss two applications of gradient learning in discriminative classification: first, in the post-classification stage, we could gain some insights on classification results by examining the score fields of the discriminative density \(p(y|x)\) learned by a classifier (e.g. logistic regression); second, we investigate how score-generated samples can be used to enhance other classifiers’ performance.

3.1 Score field of learned discriminative densities

**Representation of discriminative densities** In discriminative classification, we directly model the discriminative densities \(p(y_j|x) = \frac{e^{-f_{\theta_j}(x)}}{Z_{\theta}}\), where \(\theta = \{\theta_1, \theta_2, ..., \theta_c\}\), \(c\) is the total number of classes. In general, for multi-class classification problem, we use softmax probability where \(Z_{\theta} = \sum_{j=1}^{c} e^{-f_{\theta_j}(x)}\), and \(f_{\theta_j}(x)\) is typically linear in \(\theta_j\); for binary problem, e.g. logistic regression
with labels $y \in \{0, 1\}$, $f_{\theta_0}(x) = \theta^T x$, $f_{\theta_1}(x) = 0$ and $Z_\theta = 1 + e^{-\theta^T x}$. Without loss of generality, we write the binary discriminative densities as $p(y = 0|x) = \frac{e^{-f_{\theta_0}(x)}}{Z_\theta}$ and $p(y = 1|x) = \frac{e^{-f_{\theta_1}(x)}}{Z_\theta}$, where $Z_\theta(x) = e^{-f_{\theta_0}(x)} + e^{-f_{\theta_1}(x)}$ (here we have made $Z_\theta$ a function of $\theta$ and $x$), then the score function can be derived as (derivations see Appendix A):

$$s_{\theta_0}(x) = \frac{[f'_{\theta_1}(x) - f'_{\theta_0}(x)]e^{-f_{\theta_1}(x)}}{e^{-f_{\theta_0}(x)} + e^{-f_{\theta_1}(x)}}, s_{\theta_1}(x) = \frac{[f'_{\theta_0}(x) - f'_{\theta_1}(x)]e^{-f_{\theta_0}(x)}}{e^{-f_{\theta_0}(x)} + e^{-f_{\theta_1}(x)}}$$

(20)

where $f'(x)$ denotes derivative w.r.t. $x$. Logistic density, for example, yields:

$$e_{\theta_0} = - \frac{\theta'}{1 + e^{-\theta^T x}}, e_{\theta_1} = \frac{\theta'e^{-\theta^T x}}{1 + e^{-\theta^T x}}$$

(21)

where $\theta'$ equals $\theta$ but with intercept removed after differentiation.

As a comparison to score function, the gradient of the densities are (Appendix A):

$$\nabla_x p_{\theta_0}(x) = \frac{\nabla_x f_{\theta_1}(x) - \nabla_x f_{\theta_0}(x)}{[1 + e^{f_{\theta_0}(x) - f_{\theta_1}(x)}][1 + e^{f_{\theta_1}(x) - f_{\theta_0}(x)}]}$$

(22)

and $\nabla_x p_{\theta_1}(x) = -\nabla_x p_{\theta_0}(x)$, as $p(y = 1|x) = 1 - p(y = 0|x)$ is satisfied everywhere. Logistic density, for example, gives:

$$\nabla_x p_{\theta_0}(x) = \frac{-\theta}{(1 + e^{\theta^T x})(1 + e^{-\theta^T x})}$$

(23)

**Decision theory for discriminative classification**  Let $L(y, \hat{y})$ be the loss induced by classify $y$ (ground truth) as $\hat{y}$ (prediction), we have $L(y, \hat{y}) = 0$ if $\hat{y} = y$, and $L$ can be symmetric or asymmetric to reflect, e.g. in medical cancer scanning or credit card fraud detection scenarios, the impact or operational cost induced by false positives (FPs) and false negatives (FNs) could be different. A common choice of $L$ is the zero-one loss which simply counts the misclassification (FPs and FNs) numbers. The expected risk of predicting label $\hat{y}$ given $x$ is therefore $R(\hat{y}|x) = \sum_y L(y, \hat{y})p(y|x)$, and we make the optimal prediction $y^* = \arg \min_y R(\hat{y}|x)$. This optimal decision rule essentially leads to $y^* = \arg \max_y p(y|x)$, i.e. choosing the class with highest label probability at $x$, as this minimizes the expected loss at $x$. This optimal classifier is known as the Bayes classifier. Using this construction, we divide the feature space into multiple decision regions, and we are interested in finding the decision boundary where the two probability curves or surfaces meet with equal probability, and we may expect uncertainty to increase near the boundary as the class probabilities approach each other. More detailed treatment of decisions can be found in e.g. [1, 13, 28, 21].

In binary classification, we typically use the following decision rule to assign a label $\hat{y}$ to an observation $x$:
\[
\hat{y} = \begin{cases} 
1, & \text{if } p(y = 1|x) - p(y = 0|x) \geq \gamma_0 \\
0, & \text{otherwise}
\end{cases} \quad (24)
\]

where \( \gamma_0 \geq 0 \) is a soft margin which represents a minimum density gap for the two classes to separate. The intuition behind is, given the feature \( x \) and the estimated probabilistic model \( p(y|x) \), if the probability of being one class is higher than being the other to certain level \( \gamma_0 \), we can safely classify the sample as the higher probability class. \( \gamma_0 \) can be set to zero if we are not hoping for high confidence of separation. Rather than measuring absolute difference, we can also use the ratio \( \frac{p(y = 1|x)}{p(y = 0|x)} \geq e^{\gamma_0} \) as a discriminative criterion:

\[
\hat{y} = \begin{cases} 
1, & \text{if } \frac{e^{-f_{\theta_1}(x)}}{Z_\theta} / \frac{e^{-f_{\theta_0}(x)}}{Z_\theta} \geq e^{\gamma_0} \\
0, & \text{otherwise}
\end{cases} \quad (24b)
\]

A key step in making prediction using Eq. (24b) is to find the decision boundary equation \( e^{-f_{\theta_1}(x)} / e^{-f_{\theta_0}(x)} = e^{\gamma_0} \), which represents a curve or surface. Let's define a distance function \( \gamma(x) \):

\[
\gamma(x) = \log \frac{p(y_1|x)}{p(y_0|x)} - \log e^{\gamma_0} = f_{\theta_0}(x) - f_{\theta_1}(x) - \gamma_0 \quad (25)
\]

By introducing \( \gamma(x) \), the discriminator Eq. (24b) turns into:

\[
\hat{y} = \begin{cases} 
1, & \text{if } \gamma(x) \geq 0 \\
0, & \text{otherwise}
\end{cases} \quad (24c)
\]

The classification task becomes finding the roots of \( \gamma(x) = 0 \), which can be done using a numerical root-finding scheme such as the Newton-Raphson (NR) method:

\[
x_{k+1} = x_k - \frac{\gamma(x_k)}{\nabla_x \gamma(x_k)} \quad (26)
\]

**Score field of learned logistic densities** The use of score fields to characterize post-classification densities is illustrated using a toy example in which Gaussian data is separated by a logistic regression classifier.

We first look at the 1D case. 2000 samples are simulated (1000 for each class) with densities \( N(-2,1) \) and \( N(2,1) \) (same densities as used in Fig.1). Logistic regression (LR) is applied to the data to find the linear discriminator (the logit function in Eq.3) \( f(x) = f_{\theta_0}(x) = \theta_0 + \theta_1 x \) via MLE. The data and classification results are shown in Fig.8, where the learned parameters are \( \theta_0 = -0.1, \theta_1 = 3.5 \). The resulted logistic densities (Eq.3) are \( p(y = 0|x) = e^{0.1 - 3.5 x} / (1 + e^{0.1 - 3.5 x}), p(y = 1|x) = 1 / (1 + e^{0.1 - 3.5 x}) \). Using Eq.25 with \( \gamma_0 = 0 \), we obtain the decision boundary \( x = -\theta_0 / \theta_1 = 0.03 \).

\[\text{Not to be confused with the hyper-parameter discard rate, also denoted by } \gamma, \text{ which is used later in Langevin sampling of imbalanced data.}\]
Figure 8: Logistic regression classifier applied to toy 1D Gaussian data.

We can obtain the scores of the learned logistic densities by plugging the learned $f_{\theta_0}(x)$ and $f_{\theta_1}(x)$ into Eq. 21, as shown in Fig. 8(c). It is observed that, in the region close to the decision boundary $x = 0.03$, the two score functions exhibit abrupt changes, and gradients have sharp peaks at the boundary point.

The same procedure is repeated for the simulated 2D Gaussian data from Fig. 6, in which we each have 200 samples for each class. After applying LR, we have the decision boundary equation $\gamma(x) = f(x) = f_{\theta_0}(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ with coefficients $\theta_0 = -6.17, \theta_1 = 1.73, \theta_2 = 2$ estimated by MLE. The decision boundary is plotted in Fig. 9. Again, we observe large score and gradient values in regions near the boundary line; points far away from the boundary have vanishing scores. The score vector norm may imply some sort of sample weights in making classification decisions (e.g. identifying support vectors).

### 3.2 Discriminative classification with score-augmented data

A straightforward application of score-based generative modelling is to learn a score function from training data (potentially small and sparse), and use it to synthesize credible samples via gradient-based sampling methods (e.g. Langevin dynamics) to to populate the sample space (e.g. augmenting minor class), which can improve the performance of an off-the-shelf discriminative classifiers (e.g. nearest neighbour voting), particularly in imbalanced learning tasks.
3.2.1 A simulated 2D Gaussian data example

We use the 2D Gaussian data from Fig. 6 as an example, and re-draw the two classes, labelled 0 (colored blue) and 1 (colored red), in Fig. 10. As the data are relatively well separated, very few cross-class samples exist (i.e. samples interfering with the other class).

**Score matching** We fit two score functions to the samples, one for each class. The score function is represented using an multi-layer perceptron (MLP, fully connected neural network) trained with the loss function in Eq. 10. The matching between theoretical and predicted scores is shown in Fig. 11, from which we observe better fit in the first class, although marginal deviations between the theoretical and predicted score fields exist in both classes (i.e. gradient flows pointing away at far ends), which implies a tendency for the trained score function to potentially stretch the score field, and biasly draw samples from less populated arenas (which are currently under-represented by the given samples).

We observe from the definition of score function (Eq. 7) that, the reciprocal of the norm of a point score is roughly proportional to its pdf value, i.e.
\[ |s_\theta(x)| = |\nabla_x \log p_\theta(x)| = \frac{1}{p_\theta(x)} |\nabla_x p_\theta(x)| \]  

which hints we could use the score norm as a (very) rough estimate of probability. We thus ask the trained score functions to predict the scores at training samples and plot the samples with point size proportional to the predicted probability hinted by Eq. 27, as shown in Fig. 12. The predictions are consistent with our DGP, with central points given higher predicted probabilities. However, caution should be alerted as this rough approximation is by no means accurate, we may miss significant contribution from \(|\nabla_x p_\theta(x)|\) in Eq. 27.

**Sampling with Langevin dynamics**  The power of generative modelling, either directly modelling the DGP (e.g. GAN) or learning an underlying density distribution, lies in its capacity of generating plausible, synthesized samples at currently unavailable (e.g. due to data collection costs) regimes by sampling from the learned process (e.g. probabilistic extrapolation). However, sampling from arbitrary density is not easy [10]; numerically it can be approached by Markov chain Monte Carlo (MCMC) methods which normally involve a rejection process. As we only have accessible the sample-based score model \(s_\theta(x)\), which is an approximator of the gradient of the log-density, we use a specific MCMC procedure called *Langevin dynamics* [26] to iteratively generate a chain of samples, starting from an initial known sample \(x_0\):

\[
x_{i+1} = x_i + \epsilon \nabla_x \log p_\theta(x_i) + \sqrt{\epsilon} z_i, i = 0, 1, ..., N - 1
\]  

where \(z_i \sim \mathcal{N}(0, I)\). Following the Langevin dynamics, a random initial sample will move gradually to high density regions following the gradient vector field of log-density. Unlike other MCMC methods such as the Metropolis-Hastings sampler [14] which may need access to the (un-normalised) distribution \(p_\theta(x)\), sampling using Langevin dynamics only requires the gradient of log-density. If the perturbation parameter \(\epsilon\) (also step size) is small and the chain length \(N\)
is large to achieve equilibrium, the samples generated by Eq. 28 will converge to the true distribution \( p_\theta(x) \). Empirically, we can approximate \( \nabla_x \log p_\theta(x) \) using \( s_\theta(x) \), and repeat the above sampling process for many times, each time randomly starting from an existing sample \( x_0 \) with probability \( p_\theta(x_0) \propto 1/|s_\theta(x)| \) (Eq. 27).

Originally, we have 200 samples for each class; we ask the trained score functions to repeatedly generate 100 chains, with 1000 samples per chain (\( \epsilon = 0.003 \)) and first 200 samples discarded (ultimately we have in total 80,000 valid samples for each class). The generated samples are shown in Fig. 13. It is seen that, the density shape is preserved, with more samples clustered in the mean arena and sparse samples at margins. It massively extends the given sample profile, and even populates low density arenas, which is desirable when analysing extreme events or expensive datasets. However, although data sparsity is reduced, we should also be cautious about potentially spurious samples in arenas far from the original clusters. As noted by [9], data scarcity in low density regions can invalidate score estimation and Langevin sampling. This is observed from the training process where deviations between theoretical and predicted scores exist and may lead to a stretched valley (i.e. the learned score function may represent a dynamics different from the true DGP). This effect could be alleviated by using annealed Langevin dynamics [9].

With the augmented data, we can apply many convenient classification algorithms, hoping for improved performance compared to just using original sparse data. With plausible volume of augmented samples, a naïve approach is to label a unknown test sample by majority voting within its \( \sigma \)-neighbourhood, i.e. using empirical ratio of labels as an estimate for the discriminative probability \( p(y|x) \). For locating the neighbourhoods, two strategies can be applied: we can either fix the search radius \( \sigma \) and search within a circle centering the test point, or we can search until we find a fixed number of nearest objects (i.e. constant nearest neighbours). Another intuitive method for labelling a test point \( x = (x_1, x_2) \) is contrasting the approximate pdf values (i.e. \( p_\theta(x_0) \propto 1/|s_\theta(x)| \)) of both classes. Results derived using the generated samples are shown in Fig. 14, where

\begin{figure}[h]
  \centering
  \includegraphics[width=0.4\textwidth]{fig12}
  \caption{Probabilities (represented by point size) predicted at original samples.}
\end{figure}

\begin{figure}[h]
  \centering
  \includegraphics[width=0.4\textwidth]{fig13}
  \caption{New samples generated using learned score functions.}
\end{figure}
performances of the three classification methods are compared. It is observed that, all three methods are more confident about points close to class centers (confidence is represented by point size, where for pdfs, confidence level is proportional to its magnitude, and for counting methods, it’s the proportion of points of majority class); points lying at boundary, as evident in the nearest neighbour method, are blurry (indicated by smaller dot size) and thus involve more uncertainties.

The pseudo-pdf method labels a data point by contrasting the two inverse score norms, which may be very inaccurate if the ignored contribution of $|\nabla_x p_\theta(x)|$ is significant in Eq.27. The two counting-based methods, i.e. fixed radius and fixed nearest neighbours majority voting, are based on the newly generated samples with known labels; the choice of fixed radius and number of nearest neighbours reflects a trade-off between bias and variance (i.e. the classifier’s generalisation capacity). It is seen that searching fixed number of nearest neighbours gives the most accurate decision boundary close to the natural boundary given by the DGP in Fig.6.

![Figure 14: Predicted labels at grid points using generated samples](image)

To see the effect of data augmentation, we also present the results built on original small samples in Fig.15. We observe narrower and less dense high-confidence arenas in the fixed radius case, and less accurate inference boundary using the constant neighbours method. In both cases, predictions at the far ends (top left and bottom right) are less credible due to lack of training data in the neighbourhoods. Comparing Fig.14 and Fig.15 demonstrates the power and effectiveness of score-based data augmentation in enhancing classification quality, in the presence of small, sparse data.

**A multi-modal score function** Instead of learning two score functions separately, we could, however, learn a generic, multi-modal generator which produces samples for both classes. This approach could be more efficient in terms
of density estimation, but won’t help with classification because a uniform generator can’t help populate the feature space with labels, neither can we do pdfs contrasting: the score function only gives one unified score for an input $x$, regardless of its class. However, we could potentially utilize the unified score values of the data with known labels as inputs to train a classifier, i.e. classification directly using score features. Note that, this is not a certified approach because there is a risk that, two points from distinct classes may share the same score, e.g. $p_{\theta_1}(x) = C p_{\theta_0}(x)$, which makes them indistinguishable and misleads the classifier in the learning process.

As a toy example, the learned multi-modal score function and its capacity to re-produce the training samples are shown in Fig.16. The scores predicted by the learned multi-modal score function (Fig.16(a)) are reasonable in the sense that, it assigns small score values (i.e. proportionally heavy densities as per Eq.27) to the two cluster centers, which is also evidenced from Fig.16(b). However, it also puts some undesirable masses on margin points in between the two clusters. This could induce skewed, unrealistic behaviour when using the unified score function to represent the original two clusters with different underlying dynamics.

As a trial of classifying points based on scores predicted by the unified score function, the classification boundaries, sketched using the two popular discriminative classifiers, i.e. extreme gradient boosting trees (XGB) and neural network (NN), are shown in Fig.17. Both classifiers are trained to map the relation between the generic scores and labels; however, this mapping is biased: most predicted labels favor Class 0 (colored blue), only those in the near cluster center regimes (where density is large large) are credible for Class 1. We thus conclude that scores on its own are insufficient features for label hinting. That said, any method searching nearest neighbours purely based on scores are invalid as well.
3.2.2 Score-assisted imbalanced learning

In real-world, it’s unlikely to have two well separated classes such as the case in Fig.6; more often we may encounter high dimensional, mixed-classes data (e.g. images), and the amount of available data varies much across classes. This is common in modelling extreme events such as fraud detection (e.g. large claims in insurance, occurrence of fires or floods in natural hazards, etc). A classifier trained on imbalanced data are likely to be biased in decision-making: it may tend to acknowledge the class with more exposure. People have been using upsampling (oversampling) and/or downsampling (undersampling) techniques (Fig.18) to make a dataset balanced. Here we demonstrate the use of score functions to generate more credible or realistically-like samples for the minor class in a principled manner, hoping that score functions could be better gener-
ators in high dimensions, and classifiers trained on score-augmented data could be more skillful in identifying rare cases. This is particularly useful when large volume of data is not available due to, for example, expensive cost in data collection (e.g. clinical trials); instead, we only need to collect a few representative samples over the underlying distribution (i.e. the generative dynamics) and interpolating or extrapolating them as per the learned score function. In this regard, score function may produce more representative synthesized samples by encoding the gradient information of a log-density, and enables more efficient and flexible data generation via Langevin sampling.

A simulated high-dimensional imbalanced data example

In our first imbalanced example, we simulate in total 3000 ten-dimensional samples with small noise (0.01% random label flipping), among which 2830 samples are negative (labelled 0) and 170 are positive (labelled 1). The two classes are mixed and highly imbalanced (positive-to-negative ratio \( \sim 1:95 \)). With more exposure to negative samples, a general classifier trained on this data may tend to label unseen sample as negative.

The data is proportionally split (i.e. stratified) into training (\( \sim 75\% \)) and test (\( \sim 25\% \)) sets, with size of 2259 (positive: 128) and 571 (positive: 42) in each set. While the test set remains fixed, positive samples in the training set are augmented by sampling the learned score function using Langevin dynamics.
The score function is trained on the 128 positive training samples via score matching, and then used to generate 2002 positive samples, making numbers of positives and negatives equal after sampling. Then a discriminative classifier is fit to the newly augmented training data, and its performance is reported on the fixed test set.

| Predicted | Actual |
|-----------|--------|
|           | Negative | Positive |
| Negative  | TN       | FP (α)    |
| Positive  | FN (β)   | TP        |

Table 1: Illustration of the confusion matrix for binary classification.

The full examination table, i.e. the confusion matrix, is illustrated in Table.1 in which TN denotes true negative, FP false positive, FN false negative, and TP true positive. α refers to type I error, β is the type II error. When classifying imbalanced data, rather than focusing on accuracy (which even in worst case could be high, i.e. imagine the classifier just dumbly labels all test samples as negative), we are particularly interested in the sensitivity (true positive rate or recall, a ratio of true positives to all actual positives) and $F_1$ score metrics, the former emphasizes the skill of a classifier to identify all positives, the later is a metric blending precision and recall:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$  \hspace{1cm} (29)

where $\text{precision}=\frac{TP}{TP+FP}$ is the ratio of true positives to number of samples the classifier claims to be positive; $\text{recall}=\frac{TP}{TP+FN}$ is the ratio of true positives to the number of actual positives. They both assess a binary classifier’s ability to identify positives, benchmarking on different denominators.

When generating new samples using the trained score function, apart from the step size parameter $\epsilon$ in Langevin dynamics (Eq.28), we add two extra hyper-parameters: chain length $l$ and discard rate $\gamma$ to allow more flexibilities when sampling. Each time we randomly start from one of the 128 existing positive points in the training set, walk $l$ steps and drop the initial $l \times \gamma$ samples in the Langevin chain. Therefore, we can generate new samples with different characteristics using the same score function, e.g. short Langevin walk around existing points and preserving most points along the trajectory, or walking long while keeping only last few points. The newly generated samples are then input into two discriminative classifiers, namely a random forest (RF) classifier and a neural network (NN) classifier. The results are presented in Table.2. Also reported are classification results based on the original imbalanced data, and results from two two popular minority oversampling methods: the synthetic minority over-sampling technique (SMOTE) [18] and the Adaptive Synthetic (ADASYN) method [20].
It’s observed that, classifiers using the score function generated samples consistently outperform those using SMOTE and ADASYN upsampling methods, reporting higher recall and $F_1$ values (and most precisions). An evident example is the score-based random forest classifier with $l = 10, \gamma = 0.2, \epsilon = 0.01$ (colored red in the middle) which results in similar numbers of TNs and FPs as SMOTE and ADASYN, but with significantly smaller number of FNs and larger number of TPs. In the other two score-based settings (the last two scenarios in Table 2), we observe record high recall and $F_1$ values, respectively. Also notable is the inferior performance of neural network classifiers as compared to the ensembling method in terms of recall (except the first scenario), this might be due to the complexity of the data and the design of neural network architecture. For example, we use a fully connected MLP with layer sizes $[10, 32, 64, 128, 64, 32, 1]$, ReLU activation functions (except output layer) and binary cross entropy loss objective, trained to maturity with risk of overfitting.

| sampling + classification methods | confusion matrix | recall | precision | $F_1$ |
|----------------------------------|------------------|--------|-----------|-------|
| RF (original data)               | 693 6 35 7       | 0.17   | 0.54      | 0.25  |
| NN (original data)               | 675 24 31 11     | 0.26   | 0.31      | 0.29  |
| SMOTE + RF                       | 661 38 21 21     | 0.5    | 0.36      | 0.42  |
| SMOTE + NN                       | 666 33 20 12     | 0.29   | 0.27      | 0.28  |
| ADASYN + RF                      | 662 37 20 22     | 0.52   | 0.37      | 0.44  |
| ADASYN + NN                      | 664 35 32 10     | 0.24   | 0.22      | 0.23  |
| Score-based case 1 ($l=10, \gamma=0.2, \epsilon=0.01$) | Score + RF       | 660 39 11 31 | 0.74 | 0.44 | 0.55 |
| Score + NN                       | 653 46 22 20     | 0.48   | 0.30      | 0.37  |
| Score-based case 2 ($l=20, \gamma=0.9, \epsilon=0.01$) | Score + RF       | 630 69 6 36 | 0.86 | 0.34 | 0.49 |
| Score + NN                       | 621 78 20 22     | 0.52   | 0.22      | 0.31  |
| Score-based case 3 ($l=40, \gamma=0.9, \epsilon=0.0005$) | Score + RF       | 671 28 12 30 | 0.71 | 0.52 | 0.60 |
| Score + NN                       | 659 40 22 20     | 0.48   | 0.33      | 0.39  |

Table 2: Discriminative classifiers performance on test set (synthetic data).

We further investigate the logics behind. A comparison of the combined old and new training data generated by the three sampling methods is shown in Fig 20. We see SMOTE and ADASYN interpolate the existing positive samples, while the learned score function can both interpolate and extrapolate: starting
from any existing point, new samples can be generated by choosing proper step size $\epsilon$, walking distance $l$ (i.e. chain length) and discard rate $\gamma$. As these samples are directly sampled from the learned score function (equivalently sampling from the approximate underlying distribution), they are intrinsically representatives of the underlying dynamics. We have thus seen that the score-based generative method is effective in synthesizing rare events in high dimensions.

![Figure 20: Original and generated samples (showing first two dimensions).](image)

**A real-world fraud detection example**

As our second imbalanced classification example, we analyse an online credit card transaction dataset [5, 2, 6] which contains credit cards transactions over two days in September 2013 by European cardholders, with 492 frauds out of 284,807 transactions. The negative (non-fraudulent) and positive (fraudulent) samples are highly imbalanced, with a fraud rate of 0.173%. The original dataset has 30 anonymized features for each transaction record. To demonstrate the efficacy of classification with minimum input indicators, we choose first 10 dimensions and split the dataset into training and test sets with a train to test ratio of about 3:1. Further, to test the robustness of the score-based generative sampling, we randomly swap the labels of a small number (18 $\times$ 2) of training examples. Details about the data is presented in Table 3. The first two dimensions of the training and test sets are visualised in Fig. 21.

Without further feature engineering (e.g. normalization), we build a plain XGB booster, one of the practically popular and computationally efficient ensemble classifiers, and results shown in Table 4. It is seen that, without data augmentation, the classifier correctly identifies 54% (67 out of 123) of all the positive cases in the test set, making in total 58 mistakes (2 FPs and 56 FNs). We then separately apply SMOTE, ADASYN and the score-based methods to
upsample the minority classes, augmenting the number of positive samples to be 10 folds (i.e. 3690 cases) of the original cases and achieving a constant positive-to-negative ratio of about 1.74:100 (more samples can be generated if computational resource allows). For score-based sampling, a score function is trained using the 369 positive samples (we only need to learn one score function for the minor class), and then used to generate new positive samples which are added to the original training data.

When sampling using the learned score function, there are Langevin hyper-parameters \( (l, \gamma, \epsilon) \) which can be chosen flexibly. Each set of hyper-parameter randomly generates a set of samples, as shown in Fig. 22 and corresponding results in Table 4. Notable observations are: first, comparing the performances using original data and score-generated data (see case 1), the later yields better performance across all metrics. Second, the SMOTE and ADASYN generated data improve the classifier’s ability in identifying positives, but also introduce more FPs. Third, score-generated data show superior performance over SMOTE and ADASYN generated data (e.g. see case 2), which demonstrates the effectiveness of score-based generative sampling. Fourth, score-generated data yield comparable or better performance to all other methods in terms of total mistakes made. Considering noise introduced in the training set (36 flipped labels), the results support the argument that, score-based generative modelling could be more robust to perturbations or outliers. By generating more credible samples, it can migrate the disturbing effects of adversarial samples.

| Set | size | No.positive | No.negative |
|-----|------|-------------|-------------|
| Training | 21232 | 369 | 211963 |
| Test | 72475 | 123 | 72352 |
| Total | 284807 | 492 | 284315 |

Table 3: Specification of fraud detection data.

Figure 21: Fraud detection data (showing first two dimensions). 18 positives (\(~5\%) and 18 negatives have their labels flipped.
Figure 22: Original and generated samples (showing first two dimensions). All sampling methods produce about 3690 (10x original) positive samples, contrasting 211963 negative samples (positive-to-negative ratio 1.74:100).

Table 4: Discriminative classifiers performance on test set (fraud detection).

| sampling + classification methods | confusion matrix recall precision F1 FPs+FNs |
|----------------------------------|------------------------------------------|
| XGB (original data)              | 72350 2 67 0.54 0.97 0.70 58         |
| SMOTE + XGB                      | 72341 11 39 0.68 0.88 0.77 50         |
| ADASYN + XGB                     | 72339 13 38 0.69 0.87 0.77 51         |
| Score-based case 1 \((l=300, \gamma=0.1, \epsilon=0.0001)\) | 72350 2 67 0.65 0.98 0.78 45        |
| Score + XGB                      | 72342 10 34 0.72 0.90 0.80 44        |
| Score-based case 2 \((l=100, \gamma=0.3, \epsilon=0.01)\) | 72332 20 32 0.74 0.82 0.78 52        |

4 Score-based generative classification

Density representation and decision theory  In generative modelling, we focus on \(p(x|y)\) (or just \(p(x)\) when we are referring to a particular class). We have seen examples of Gaussian scores in Fig.1 and Fig.2. If we have multiple DGPs, each class-conditional density \(p(x|y_j)\) can be modelled as \(p(x|y_j) = \frac{e^{-f_{y_j}(x)}}{Z_{y_j}}\).
Two classes as an example, their densities can be written as $p_{\theta_0}(x) = \frac{e^{-f_{\theta_0}(x)}}{Z_{\theta_0}}$ and $p_{\theta_1}(x) = \frac{e^{-f_{\theta_1}(x)}}{Z_{\theta_1}}$, and corresponding score functions derived as $s_{\theta_0} = -\nabla_x f_{\theta_0}(x)$, $s_{\theta_1} = -\nabla_x f_{\theta_1}(x)$. Unlike the discriminative density $p(y|x)$ with property $p_{\theta_0}(y = 0|x) + p_{\theta_1}(y = 1|x) = 1$, which leads to a shared normalising constant, the constants $Z_\theta$ in the generative densities $p(x|y)$ don’t naturally connect to each other, neither are they necessarily the same.

If we encode equal prior probabilities for both classes, the Bayes rule (Eq.[1]) can be simplified:

$$p(y_j|x) = \frac{p(x|y_j)}{\sum_{j=1}^{c} p(x|y_j)}$$

and the decision rule reduces to simply choosing the class label with higher density value, which again gives the Bayes classifier. Binary classification, for example, has the generative decision rule (using a soft margin $\gamma_0 = 0$ in accordance with the discriminative rule Eq.[24]):

$$\hat{y} = \begin{cases} 1, & \text{if } \log \frac{p(x|y=1)}{p(x|y=0)} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

This simple density contrasting rule has previously been exercised in Fig.[7]. To apply this rule, a key step is to construct the class-conditional density $p_{\theta}(x)$ from score function $s_{\theta}(x)$. This can be done using the density construction method described in Section[2] (e.g. Algorithm[1]), which requires an initial probability $p(x_0)$, normally empirically estimated, to be supplied. If we have additional knowledge about the samples, e.g. a cluster is Gaussian distributed, we can make use of this information in constructing the density, and potentially obtain better density estimation and decisions. We describe this using the 1D and 2D Gaussian data examples in the following.

An alternative and costive approach is to numerically solve the decision boundary equation in Eq.[30] via NR which makes use of score function. Using the above specified density representations, boundary condition can be written as:

$$\gamma(x) = \log \frac{p(x|y=1)}{p(x|y=0)} = f_{\theta_0}(x) - f_{\theta_1}(x) + \log Z_{\theta_0} - \log Z_{\theta_1} = 0$$

To numerically find its roots (i.e. points lying on the boundary), the NR updating formula (Eq.[26]) is applied, in which the derivative $\nabla_x f_{\theta_0}(x) = \nabla_x f_{\theta_1}(x) = s_{\theta_0}(x) - s_{\theta_1}(x)$. However, when using Eq.[26] updates, we are blocked by evaluating $\gamma(x)$, which involves assessing $\log Z_\theta$ that could be intractable. If we make some distributional assumption about data, e.g. each class is Gaussian distributed, then we can empirically estimate $\log Z_\theta$ (and other constants associated with the assumed density). However, this raises the question: if we know the distributional properties of the clusters, we could directly construct their
densities using estimated sufficient statistics and draw the decision boundary already, so why bother learning their score functions, performing integration and/or going numerical with extra costs? There are two reasons: first, learning a score function in many cases is cheap and fast, almost marginal with aid of modern ML techniques such as deep learning and automatic differentiation (both are used in score matching, e.g. a shallow multi-layer perceptron can achieve satisfying loss). Second, we will see from following Gaussian data examples that, the decision boundary found by \textit{NR} via scores are very close to theoretical ones, despite the inaccuracies in score learning.

4.1 Two toy Gaussian data examples

We apply generative classification to the simulated 1D (Fig.1) and 2D (Fig.6) Gaussian data. Given the fact that there are equal number of positives and negatives in both scenarios (1000 for each class in the 1D Gaussian case, and 200 in the 2D case), the prior probabilities for both classes are equal, and the decision rule Eq.1b can be directly applied.

Pretending that we have no access to DGPs, we just assume Gaussian distribution for both clusters (by visually inspecting the data), which hints the theoretical normalising constant \( Z = (2\pi)^{d/2}|\Sigma|^{1/2} \) (see Gaussian density in Eq.8), and score function \( s(x|\mu, \Sigma) = \Sigma^{-1}(\mu - x) \). We learn two parametric, linear score functions \( \hat{s}_\theta(x) = \hat{A}x + \hat{b} \) (here \( \theta = [\hat{A}, \hat{b}] \)) via score matching; these learned score functions can be analytically integrated to yield \( \hat{f}_\theta(x) = -\frac{1}{2}x^T\hat{A}x - \hat{b}^T x + \hat{C} \) (Eq.7), where the constant \( \hat{C} \) can be obtained by using \( \hat{f}_\theta(\hat{\mu}) = 0 \) (Gaussian property):

\[
\hat{f}_\theta(x) = -\frac{1}{2}x^T\hat{A}x - \hat{b}^T x + \frac{1}{2}\hat{\mu}^T\hat{A}\hat{\mu} + \hat{b}^T\hat{\mu}
\]

where \( \hat{\mu} \) is the sample mean. We can also derive a sample-based guess \( \hat{Z}_\theta = (2\pi)^{d/2}|\hat{\Sigma}|^{1/2} \), which enables construction of an estimated density:

\[
\hat{p}_\theta(x) = e^{-\hat{f}_\theta(x)}/\hat{Z}_\theta
\]

Note Eq.33 has been derived using the strong assumption of distributional structure of data.

Alternatively, given \( \hat{s}_\theta(x) \), \( \hat{f}_\theta(x) \) and \( \hat{Z}_\theta \), we can numerically find the boundary using the \textit{NR} updating rule (Eq.26) which solves the boundary equation Eq.31:

\[
x_{k+1} = x_k - \frac{\hat{f}_{\theta_a}(x) - \hat{f}_{\theta_0}(x) + \log \hat{Z}_{\theta_a} - \log \hat{Z}_{\theta_0}}{\hat{s}_{\theta_a}(x) - \hat{s}_{\theta_0}(x)}
\]

We have therefore two equivalent choices at hand in seeking the boundary: either directly comparing \( e^{-f_{\theta_a}(x)}/\hat{Z}_{\theta_a} \) and \( e^{-f_{\theta_0}(x)}/\hat{Z}_{\theta_0} \) at grid points with assistance of Eq.32, or using a numerical scheme such as \textit{NR}. We test both methods for the 1D (Fig.1) and 2D (Fig.6) Gaussian data.
We first present the toy 1D Gaussian results. The sample means and standard deviations are $\hat{\mu}_0 = -2.01, \hat{\sigma}_0 = 1.0, \hat{\mu}_1 = 2.04, \hat{\sigma}_1 = 1.0$. The learned score functions, optimized via score matching, are $\hat{s}_{b_0}(x) = -0.99x - 2$ and $\hat{s}_{b_1}(x) = -0.99x + 2$. They are close to the theoretical score function $s(x) = (\mu - x)/\sigma$ with DGP parameters $\mu = \pm 2, \sigma = 1$. Using Eq. 32 and Eq. 33, the two densities, $\hat{\rho}_{b_0}(x)$ and $\hat{\rho}_{b_1}(x)$, are recovered in Fig. 23. We observe improvement over Fig. 4 with the additional distributional assumption, we are able to coin analytical score functions and evaluate density constants from samples, which reduces uncertainties and errors, and rewards a higher recovery accuracy, evidenced by near-zero JSD in both cases.

![Figure 23: Recovered densities using learned (analytical) score function. There are 1000 samples for each class, with locations shown at bottom. The score functions use a two-layer MLP with sizes $[1,1]$, representing a linear function $\hat{s}(x) = \hat{a}x + \hat{b}$. Initial probability $p_{b_0}(\mu_0 = [-2.01]) = 0.40, p_{b_1}(\mu_1 = [2.04]) = 0.40.$](image1)

Direct comparison of the two estimated densities $\hat{\rho}_{b_0}(x)$ and $\hat{\rho}_{b_1}(x)$ gives the decision boundary around $x = 0$. Newton-Raphson finds the decision boundary at $x = 0.017$. Both are close to the density cross-over point $x = 0$ where the generative decision rule (Eq. 11) points to.

For the 2D case, sample means and covariances are $\hat{\mu}_0 = [0.07, -0.12], \hat{\mu}_1 = [4.06, 4.18], \hat{\Sigma}_0 = \begin{bmatrix} 1.08 & -0.56 \\ -0.56 & 1.02 \end{bmatrix}, \hat{\Sigma}_1 = \begin{bmatrix} 1.01 & 0.56 \\ 0.56 & 1.06 \end{bmatrix}$. The learned score functions $\hat{s}(x) = \hat{A}x + \hat{b}$ have coefficients $\hat{A}_0 = \begin{bmatrix} -1.30 & -0.71 \\ -0.71 & -1.38 \end{bmatrix}, \hat{b}_0 = (0.01, -0.11)$, $\hat{A}_1 = \begin{bmatrix} -1.41 & 0.75 \\ 0.75 & -1.34 \end{bmatrix}, \hat{b}_1 = (2.57, 2.57)$. The learning is imperfect though: theoretically we expect $A_0 = -\Sigma_0^{-1} = \begin{bmatrix} -1.33 & -0.67 \\ -0.67 & -1.33 \end{bmatrix}, b_0 = \Sigma_0^{-1}\mu_0 = [0, 0]$ and $A_1 = -\Sigma_1^{-1} = \begin{bmatrix} -1.33 & 0.67 \\ 0.67 & -1.33 \end{bmatrix}, b_1 = \Sigma_1^{-1}\mu_1 = [2.67, 2.67]$. The errors might be induced by finite samples, e.g. the estimated sample mean $\hat{\mu}_0 = [0.07, -0.12]$ deviates from true DGP parameter $\mu_0 = [0, 0]$, etc.

The estimated densities from Eq. 33 and resulted decision boundary are presented in Fig. 24. Compared with Fig. 7, the analytically recovered densities and
boundary are of better quality (i.e. thinner and less heavy tails), and closer to the theoretical ones in Fig.6(a). This is due to the extra assumption of Gaussian distribution of data, which gives linear form of score function and leads to (approximately) accurate estimation of the constants in the density (i.e. $\hat{C}$ in Eq.32 and $\hat{Z}_{\theta}$ in Eq.33).

### Figure 24: Recovered densities and decision boundary using learned (analytical) score function. There are 200 samples for each class. The score functions use a two-layer MLP with sizes $[2,2]$, representing a linear function $\hat{s}(x) = \hat{A}x + \hat{b}$. Initial probability $p_{\theta_0}(\hat{\mu}_0 = [0.07, -0.12]) = 0.18$, $p_{\theta_1}(\hat{\mu}_1 = [4.06, 4.18]) = 0.18$. On the right, brighter color implies higher recovered density, contour and green dots represent DGP-based (theoretical), white dots indicate the equal density boundary generated by recovered densities.

The decision boundary found by the NR method are presented in Fig.25. This numerically derived boundary (blue dots) trembles around the theoretical one (green dots). Despite inaccuracies exist in score function learning, the resulted boundaries are still satisfying, which could be (hesitantly) attributed to the denominator and numerator cancellation effect in the NR updating formula. An extended theoretical treatise of the boundary between two Gaussian densities can be found in Appendix.B.

Compared to the density construction method in Section.2, which requires only the learned score functions $\hat{s}(x)$ and initial point probabilities $\hat{p}(x_0)$, the method used here provides accurate boundary inference. However, when distributional assumption cannot be made, or the learned score function is complex (e.g. deep NN) and cannot be analytically integrated, we may have to stick to the former method. This is illustrated in following two examples.

### 4.2 Generative classification of imbalanced data

The two imbalanced datasets, i.e. the medium-size, high-dimensional synthetic data, and the large-volume, real-world fraud detection data, both used in discriminative classification, are again employed. As the two classes are highly
imbalanced (positive-to-negative ratio of 1:95 in the synthetic case, and fraud rate of 0.173% in the second case), considering data imbalance, the prior probabilities could exert great influence on the posterior, we therefore adopt the original Bayes rule (Eq.1).

As we are contrasting the two posteriors, we need to learn a score function for each class. However, this time we won’t try to construct the whole density surface and find the boundary; instead we query the score function (the ‘oracle’), along with a supplied initial density value, on a come-and-serve basis. The initial density values can be estimated using two methods: the neighbourhood counting method which divides frequencies of class appearance in a $\sigma$-neighbourhood (with radius $\sigma$) of $x_0$ by its total number and volume, and the Gaussian estimation method which gives $\hat{p}(x_0 = \hat{\mu}) = 1/(2\pi)^{d/2}|\hat{\Sigma}|^{1/2}$. The former gives rough estimation, while the later makes a weak, partial Gaussian assumption on data. Unlike the strong Gaussian distributional assumption used in the 1D and 2D Gaussian data examples, the linear structure assumption are not imposed in score function learning, we are free to choose any proper neural network architecture (not limited to two-layer and linear) to fit the gradient field via score matching. However, still lack of evidence of Gaussianity, e.g. density peaks at sample mean, could induce inaccurate score estimation, which is particularly unfair for multimodal data. In the simulated imbalanced data example, we use the Gaussian initial density estimates; we compare both methods in the real-world fraud detection example.

4.2.1 The simulated high-dimensional imbalanced data example

This time we use different settings for learning the score functions: a slightly shallower neural network architecture with layer sizes [10, 128, 128, 10] is used for both score functions, demonstrating the flexibility of score function representation. We also use different learning rates to update weights during backpropagation, $\alpha_0 = 0.001$ for the major class (labeled 0), and $\alpha_1 = 0.01$ or 0.05 for the minor class (labeled 1), allowing for different levels of details to be learned adapting to data volume.
The empirical prior probabilities for the major and minor classes are 0.943 and 0.057, given by the two class ratios. An (rough) visual inspection of the first two dimensions of the data (Fig.19) hints that we can use a Gaussian guess of the initial density, which gives $\hat{\rho}_0(x_0 = \mu_0) = 0.9 \times 10^{-4}$ and $\hat{\rho}_1(x_0 = \mu_1) = 1.2 \times 10^{-4}$. The two densities $p_\theta(x_0)$ at each test point are calculated incrementally using segment line integral (Eq.18) with the supplied initial density values. The posterior, computed by Eq.1, are compared and data classified as the class with higher posterior. The results yielded by two learned score models are reported in Table.5. The generative classifiers yield comparable (marginally better) performance to the SMOTE and ADASYN based discriminative classifiers in Table.2: classifier 1 identifies the same number of positives (21 TPs out of 42 overall positives) as the random forest (RF) discriminative classifier with SMOTE upsampling, with less FPs (27 vs 38); classifier 2 also shows similar performance (22 TPs correctly identified, 35 FPs made by generative classifier 2 vs 37 FPs made by RF + ADASYN). Compared to the score-assisted discriminative classifiers (last 3 rows in Table.2), the generative classifiers show inferior performance to the random forest classifiers using score-augmented data, but superior to the score-assisted neural network classifier.

| Scenario | confusion matrix | recall | precision | $F_1$ |
|----------|------------------|--------|-----------|-------|
| Score-based generative classifier 1 ($\alpha_0 = 0.001, \alpha_1 = 0.01$) | $\begin{bmatrix} 672 & 27 \\ 21 & 21 \end{bmatrix}$ | 0.50 | 0.44 | 0.47 |
| Score-based generative classifier 2 ($\alpha_0 = 0.001, \alpha_1 = 0.05$) | $\begin{bmatrix} 664 & 35 \\ 20 & 22 \end{bmatrix}$ | 0.52 | 0.39 | 0.44 |

Table 5: Generative classifiers performance on test set (synthetic data, $\alpha$: learning rate of score network).

### 4.2.2 The fraud detection example

We apply generative classification to the highly imbalanced, ten-dimensional fraud detection dataset (with in total 36 flipped class labels). Unlike the discriminative case where we only learn a score function for the minor class to augment data, this time we need to learn separately two score functions and use them to predict class-specific probabilities for each test point. Again we built two generative classifiers with different learning settings, and report the results in Table.6. Features are scaled using z-score standardization before training. The same neural network architecture, with layer sizes [10, 128, 128, 10], are used for representing score functions for both classes. In training the score functions, stochastic gradient descent (SGD) with batch sizes of 128 (major class) and 64 (minor class) is used to accelerate the training process. Note that, in contrast to the case of synthetic example, we apply smaller learning rate for the minor class with sufficiently large number of epochs, which allows the score network to learn more details from the data while evolving slowly.

33
Two initial density estimation methods are used: the neighbourhood counting method and the Gaussian density estimation method. Both methods estimate an initial density value $\hat{p}_\theta(x_0 = \hat{\mu})$ at the sample mean location. A visual inspection of the first two dimensions (Fig.21) suggests that the data may not be Gaussian distributed, which invalidates the Gaussian initial density method. The $\sigma$-neighbourhood counting method calculates the ratio of samples within a circle of radius $\sigma = 1.0$ centering $x_0 = \hat{\mu}$ (i.e. the area under pdf in the neighbourhood of $x_0$). These two methods may give different estimation values, however, as we are contrasting the two densities, the absolute values may be irrelevant once they give similar scale class density ratios. The Gaussian method yields an initial density ratio $\hat{p}_\theta(x_0 = \hat{\mu}) / \hat{p}_\theta(x_0)$ of 140 while the counting method gives $\sim 70$. It is observed that, the neighbourhood counting method gives slightly better results in terms of recall (and similar $F_1$ values), at the expense of worse precision (i.e. more FPs).

A fair comparison can be made between the test performances of the generative classifier (Table.6) and the discriminative XGB classifier (first row in Table.4), both built on original data. The generative classifiers identify marginally more positives (indicated by higher recalls) at the expense of introducing more FPs. We are cautious to compare the generative results to other data-augmented discriminative cases, neither can it be compared to other state-of-the-art hand crafted methods, as the data (only 10 features features and original samples are used), feature engineering (only standardization is used), and noise levels (36 labels swapped) used could be different.

| $p(x_0)$ estimation method | Scenario | confusion matrix | recall | precision | $F_1$ |
|---------------------------|----------|------------------|--------|-----------|-------|
| Gaussian                  | Score-based generative classifier 1 $(\alpha_0 = 0.05, \alpha_1 = 0.001)$ | 72166 | 186 | 51 | 72 | **0.59** | 0.28 | 0.38 |
| $\hat{p}_\theta(x_0 = \hat{\mu}_0) = 7.0 \times 10^{-6}$ | Score-based generative classifier 2 $(\alpha_0 = 0.05, \alpha_1 = 0.0005)$ | 72269 | 83 | 52 | 71 | 0.58 | 0.46 | 0.51 |
| $\hat{p}_\theta(x_0 = \hat{\mu}_1) = 5.0 \times 10^{-5}$ | Score-based generative classifier 3 $(\alpha_0 = 0.01, \alpha_1 = 0.0005)$ | 72299 | 53 | 50 | 64 | 0.52 | 0.55 | **0.53** |
| Neighbourhood counting    | Score-based generative classifier 1 $(\alpha_0 = 0.05, \alpha_1 = 0.001)$ | 72148 | 204 | 47 | 76 | **0.62** | 0.27 | 0.38 |
| $\hat{p}_\theta(x_0 = \hat{\mu}_0) = 0.548$ | Score-based generative classifier 2 $(\alpha_0 = 0.05, \alpha_1 = 0.0005)$ | 72261 | 91 | 48 | 75 | 0.61 | 0.45 | 0.52 |
| $\hat{p}_\theta(x_0 = \hat{\mu}_1) = 0.008$ | Score-based generative classifier 2 $(\alpha_0 = 0.01, \alpha_1 = 0.0005)$ | 72285 | 67 | 55 | 68 | 0.55 | 0.50 | **0.53** |

Table 6: Generative classifiers performance on test set (fraud detection, $\alpha$: learning rate of score network).
5 Discussions

In this work, we investigated the applications of score-based generative modelling in discriminative and generative classification settings. Specifically,

- We illustrated the fundamental principles of score-based generative modelling, including score function as an alternative to density for characterizing data distribution, learning individual and generic score functions via score matching, constructing densities from scores in low and high dimensions, and sample generation via Langevin dynamics.

- We discussed the basic decision theories for both discriminative and generative classification problems, and how score functions can be involved in making decisions. For generative classification, learned score functions can be used to construct class-conditional densities which is used directly in the Bayes rule; for discriminative classification, it can be used to augment sample space, or to characterize learned densities in the post-classification stage. In both tasks, scores can be used to find the decision boundary with Newton-Raphson method.

- We performed discriminative and generative classification experiments on three datasets: a simulated 1D and 2D Gaussian dataset of small size, a simulated imbalanced dataset of medium size and high dimension, and a large volume, high-dimensional, highly imbalanced real-world fraud detection dataset. The score-based data augmentation methodology was benchmarked by SMOTE and ADASYN oversampling methods, and exhibited superior performance in imbalanced classification tasks. The score-based generative classifiers yield comparable performance to discriminative classifiers.

Across this work, these techniques are explained with analytical, numerical and experimental examples. Various classifiers are applied in different settings, i.e. nearest neighbours, logistic regression, random forest, extreme gradient booster, neural network for discriminative classification, and (implicitly) Bayes classifier for generative classification. Gaussian densities are employed in many of our analytical and simulation examples, due to its analytical tractability and numerical verifiability. To foster further discussions, we address some interesting issues identified in our work.

Learning the score network  We can write a score function in parametric form (e.g. Gaussian scores), or represent it using a neural network, which is of advantage as neural network can model complex, high-dimensional, non-linear relations. Learning a score function, in many cases, is cheap and fast (e.g. optimization via first-order methods such as gradient descent). Also, training a small to medium size neural network is made fast with aid of modern learning techniques (e.g. SGD). At inference time, however, it could be slow if the size of test samples is large (as in the fraud detection example). The computational
intensity gets worse if Monte Carlo integral is evaluated incrementally when inferring densities.

When learning a score network, it might be beneficial to standardize or normalize the input features for two reasons: first, to assign (roughly) equal weight to each feature, because features with large values could exert more influence on loss. Second, to accelerate the learning process. Standardization can help reshape the skewed loss surface to an standard ellipse, which facilitates gradient descent updating. We exercised this in generative classification of fraud detection data, but have not benchmarked it. With or without feature scaling might lead to different learned score fields, and therefore influence subsequent density construction and classification outcomes. Also, in the presence of large volume data (e.g. the major class in fraud detection), stochastic gradient descent can be used for training the score network, although bias and inaccuracies may be introduced.

Trustful data augmentation Score function can be used to generate synthetic data with high fidelity. The data generation process makes use of the learned gradient information and the efficient Langevin sampling method. It could be particularly useful for synthesizing minority class in imbalanced learning scenario. The generated samples are representative, credible and well-behaved: they interpolate and extrapolate existing data, mimic the behaviour of the underlying distribution (e.g. Fig.13 and Fig.20). Data generation is made successful given a well-learned score function and proper sampling settings.

Generative classifier for imbalanced data The generative approach is principled and explainable, particularly in classifying imbalanced data: it acknowledges the fact of imbalance by encoding the imbalance ratio into priors. Therefore, it may be the natural way to model imbalance data. With score-based modelling, the task of generative classification becomes learning class-conditional score functions, construct class-conditional densities and compare posteriors. Scores are also involved in solving the decision boundary equation via Newton-Raphson method (e.g. Fig.7). If additional distributional assumption is added, we may be able to coin the learned score functions and densities in analytical form (Fig.24).

Tuning the (hyper)parameters It remains as an open question how to choose a proper neural architecture and its hyper-parameters (e.g. learning rate) to sufficiently, and not overly, extract gradient information from data. Shallow or deep, narrow or wide, high or low learning rates, etc, could lead to underfitting or overfitting behaviours; they are general topics rooted in deep learning. We acknowledge that different networks can approximate the same score function with different (computational) complexities and accuracies. The image of neural architecture search is beyond this paper.

As an initial density value $\hat{p}_0(x_0)$ is required for score-based density construction, challenge lies in how to supply an educated guess of $\hat{p}_0(x_0)$ for each
class. If the data is reasonably Gaussian clustered, we can use the Gaussian central probability \( p(\mu) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \). However, misuse of the Gaussian guess could induce error when data distribution is not Gaussian (e.g. the fraud detection data) and if it’s multi-modal (we might be supplying an badly estimated, non-representative point). In such case we have the purely empirical neighbourhood counting method.

There are also hyper-parameters in sampling. For example, in Langevin dynamics we have step size \( \epsilon \), chain length \( l \) and discard rate \( \gamma \). The aim of sampling is to generate samples that follow the underlying distribution. To achieve this we perturb existing samples and produce new samples following the learned gradient fields, using proper hyper-parameter settings. However, we may still have problems of slow mixing of Langevin dynamics and ineffective learning in data scarcity regimes [9].

### 6 Future work

This work builds some basic blocks and serves as an intro to discriminative and generative classification with score-based generative modelling. Future work may include extending the current framework to many other applications where learning-based data generation is demanded (e.g. rare event modelling, adversarial training), or density estimation is difficult (e.g. high dimensions). Gradient learning has the advantage that the learning outcome (i.e. the score function) is not constrained by a unit sum, and it can be represented by any input-output mapping (e.g. a parametric formula or an architecture). We could, for example, look into other task-specific learning representations for score functions, directly learning the score function of a discriminative density, making decisions by contrasting individual and global scores, or using scores as features in other machine learning tasks. Opportunities also exist in exploring other gradient-based sampling routines (e.g. HMC), utilising scores as an approximation; or comparing other generative modelling methods. Some automation process may be devised to train a proper score architecture to match a desired density profile, and tune the hyper-parameters in the sampling procedure. Generative modelling in general could be a cheap surrogate for active learning wherever data labelling is expensive. Although this work focuses on binary classification tasks, it can be extended to multi-class classification problems, using e.g. one-vs-one or one-vs-all contrasting strategies [28].

### 7 Related work

On general classification, Friedman et al. [13], James et al. [28], Murphy [21] and Berger [1] gives comprehensive treatments of the classification decision theory. Ng and Jordan [17] compared logistic regression and naive Bayes classifiers,

---

[10]The related work list is by no means complete due to page limit.
and showed that while discriminative learning has lower asymptotic error, a generative classifier may approach its (higher) asymptotic error much faster. Murphy [21] described discriminative and generative classifiers in various settings. Rasmussen [23] described the decision theory for both types of classification in Bayesian context. James et al. [28] also have a treatise on different classifiers. For classification in the imbalanced domain, Chawla et al. [18] proposed the SMOTE oversampling approach for classifying imbalanced datasets and showed that a combination of oversampling and undersampling can achieve better classifier performance (measured by ROC) using C4.5, Ripper and a Naive Bayes classifiers. Moniz et al. [4] proposed variants of SMOTEBoost which combine boosting technique and SMOTE resampling, and demonstrated its use in imbalanced regression tasks such as extreme values prediction. He et al. [20] presented the ADASYN approach for generating samples based on their level of difficulty in learning, which reduces the bias introduced by class imbalance and adaptively shifts the decision boundary. Pozzolo et al. [2] argued that the bias due to undersampling significantly impacts classification accuracy and probability calibration, and tested the argument on the credit card transaction dataset which is also used in this work.

On sampling methods, Metropolis et al. [27] proposed the Metropolis algorithm, Hastings [14] introduced the Metropolis-Hastings sampler, Geman [12] developed the Gibbs sampler, Kloek et al. [19] introduced importance sampling, Simon et al. [24] proposed the Hamiltonian Monte Carlo (HMC) algorithm, which mixes molecular dynamics and Langevin to guide MC simulation. In recent decades, sequential MC methods [11, 22] are also advanced. While some MC methods may suffer from random walk behaviour, Welling and Teh [26] combined stochastic gradient optimization (with Langevin dynamics for noise injection) and Bayesian posterior sampling to enable efficient MCMC sampling and generate samples which converge to the full posterior distribution.

On score methods, Hyvärinen [15] proposed the score matching method for estimating non-normalized statistical models, derived the simplified objective function, and validated it on multivariate Gaussian and independent component analysis models and image data. Hyvärinen [16] extended score matching for binary variables and the non-negative real domain, and obtained in closed form for some exponential families. Yu et al. [25] described a generalized form of score matching for non-negative data with improved estimation efficiency, and improved theoretical guarantees of the regularized score matching method. Song and Ermon [9] proposed the combined use of score function and Langevin dynamics for generative sampling, and introduced perturbation in score estimation, which gives comparable performance to GANs. Song et al. [8] proposed sliced score matching for complex models and higher dimensional data by projecting the scores onto random vectors, and applied it to deep energy-based models, variational inference and Wasserstein Auto-Encoders. Pacchiardi and Dutta [7] used score matching for training a neural conditional exponential family to approximate the ABC likelihood, and applied it in MCMC sampling for intractable distributions and to large-dimensional time-series model. Generative models have been used as adversarially robust classifiers for complex
datasets, particularly in the image classification domain [29]. Zimmermann et al. [29] investigated score-based generative classification of natural images, and found marginal advantage over discriminative classifiers in terms of adversarial robustness, yet it provides a different approach to classification.

8 Conclusions

Score-based generative method is efficient in terms of learning and sampling, robust to perturbations, effective in high dimensions and imbalanced situations. With sample-based score matching, learning complex (e.g. multi-modal) score functions are enabled by modern deep learning techniques. Sampling from a score function is convenient via Langevin dynamics; the sampling process is flexible, yields better distributional properties compared to interpolation methods, and is particularly useful in the presence of small, sparse, or disturbed data. Score-based generative classification, with comparable performance and marginal advantage, provides a novel method to classification; discriminative classification with score-generated data gives better performance over other data augmentation methods across metrics, as evidenced by simulated and real-world experiments.

9 Code availability

All codes are available on https://github.com/YongchaoHuang.

References

[1] James O. Berger. “Statistical Decision Theory and Bayesian Analysis”. In: Springer, 1985.
[2] Andrea Dal Pozzolo; Olivier Caelen; Reid A. Johnson; Gianluca Bontempi. “Calibrating Probability with Undersampling for Unbalanced Classifica- tion.” In: Symposium on Computational Intelligence and Data Mining (CIDM), IEEE (2015).
[3] Mykola Bordyuh. Implementation of the score matching algorithm. 2021. URL: https://github.com/mbordyuh/score_matching
[4] Nuno Moniz; Rita Ribeiro; Vitor Cerqueira; Nitesh Chawla. “SMOTEBoost for Regression: Improving the Prediction of Extreme Values”. In: 5th IEEE International Conference on Data Science and Advanced Analytics (2018).
[5] Credit Card Fraud Detection. 2021. URL: https://www.kaggle.com/mlg- ulb/creditcardfraud
[6] Credit Card Fraud Prediction with XGBoost. 2020. URL: https://github.com/wmlba/Fraud_Detection_Techniques
[7] Lorenzo Pacchiardi; Ritabrata Dutta. “Score Matched Neural Exponential Families for Likelihood-Free Inference.” In: Journal of Machine Learning Research 23 (2022), pp. 1–71.

[8] Yang Song; Sahaj Garg; Jiaxin Shi; Stefano Ermon. “Sliced score matching: A scalable approach to density and score estimation”. In: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference 115 (2020), pp. 574–584.

[9] Yang Song; Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) (2019).

[10] Joseph J.K. O Ruaanaidh; William J. Fitzgerald. “Numerical Bayesian Methods Applied to Signal Processing”. In: 1st. Springer, 1996, p. 244.

[11] Arnaud Doucet; Nando de Freitas; Neil Gordon. “Sequential Monte Carlo Methods in Practice”. In: Springer, 2016.

[12] S. Geman and D. Geman. “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”. In: IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-6 (1984), pp. 721–741.

[13] Jerome H. Friedman; Robert Tibshirani; Trevor Hastie. “The Elements of Statistical Learning”. In: Springer, 2001.

[14] W.K. Hastings. “Monte-Carlo sampling methods using Markov chains and their applications”. In: Biometrika 57 (1970), pp. 97–109.

[15] Aapo Hyvärinen. “Estimation of non-normalized statistical models by score matching.” In: Journal of Machine Learning Research 6(Apr) (2005), pp. 695–709.

[16] Aapo Hyvärinen. “Some extensions of score matching.” In: Computational Statistics and Data Analysis 51 (2007), pp. 2499–2512.

[17] Andrew Ng; Michael Jordan. “On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes”. In: Advances in Neural Information Processing Systems (2001) 14 (2001).

[18] N. V. Chawla; K. W. Bowyer; L. O.Hall; W. P. Kegelmeyer. “SMOTE: synthetic minority over-sampling technique”. In: Journal of artificial intelligence research (2002), pp. 321–357.

[19] H. K. Kloek T.; van Dijk. “Bayesian Estimates of Equation System Parameters: An Application of Integration by Monte Carlo”. In: Econometrica 46(1) (1978), pp. 1–19.

[20] Haibo He; Yang Bai; Eduardo A Garcia; and Shutao Li. “Adasyn: adaptive synthetic sampling approach for imbalanced learning.” In: 2008 IEEE International Joint Conference on Neural Networks (2008), pp. 1322–1328.

[21] Kevin P. Murphy. “Machine Learning: A Probabilistic Perspective”. In: 1st. The MIT Press, 2012, p. 1067.
[22] Nicolas Chopin; Omiros Papaspiliopoulos. “An introduction to Sequential Monte Carlo”. In: Springer, 2020.
[23] Christopher K. I. Williams; Carl Edward Rasmussen. “Gaussian Processes for Machine Learning”. In: MIT Press, 2005.
[24] Simon Duane; Anthony D.Kennedy; Brian J.Pendleton; Duncan Roweth. “Hybrid Monte Carlo”. In: Physics Letters B 195(2) (1987), pp. 216–222.
[25] Shiqing Yu; Mathias Drton; Ali Shojaie. “Generalized Score Matching for Non-Negative Data.” In: Journal of Machine Learning Research 20 (2019), pp. 1–70.
[26] M. Welling; Y. W. Teh. “Bayesian learning via stochastic gradient langevin dynamics.” In: Proceedings of the 28th international conference on machine learning (ICML-11) (2011), pp. 681–688.
[27] Nicholas Metropolis; Arianna W. Rosenbluth; Marshall N. Rosenbluth; Augusta H. Teller. “Equation of State Calculations by Fast Computing Machines”. In: Journal of Chemical Physics 21 (1953), pp. 1087–1092.
[28] Gareth M. James; Daniela Witten; Trevor Hastie; Robert Tibshirani. “An Introduction to Statistical Learning”. In: Springer, 2013.
[29] Roland S. Zimmermann et al. Score-Based Generative Classifiers. 2021. doi: [10.48550/ARXIV.2110.00473](https://arxiv.org/abs/2110.00473).
Appendix A  Derivation of discriminative densities

Here we provide details of deriving the score functions and gradients for the binary case in Section 3. Using the specified representations of discriminative densities \( p(y=0|x) = \frac{e^{-f_0(x)}}{Z_0} \) and \( p(y=1|x) = \frac{e^{-f_1(x)}}{Z_0} \), where \( Z_0(x) = e^{-f_0(x)} + e^{-f_1(x)} \), we have:

\[
s_{\theta_0}(x) = \nabla_x \log p(y=0|x) = \frac{\partial[-f_{\theta_0}(x) - \log Z_0]}{\partial x} = -f'_{\theta_0}(x) - \frac{1}{Z_0} Z'_0 = -f'_{\theta_0}(x) - \frac{1}{Z_0} e^{-f_0(x)} + e^{-f_1(x)} \]

where \( f'(x) \) denotes derivative w.r.t. \( x \). Similarly, \( s_{\theta_1} \) can be derived as:

\[
s_{\theta_1}(x) = \frac{[f'_{\theta_0}(x) - f'_{\theta_1}(x)]e^{-f_0(x)}}{e^{-f_0(x)} + e^{-f_1(x)}} \tag{36}\]

We see that:

\[
s_{\theta_0}(x) + s_{\theta_0}(x) = \frac{[f'_{\theta_0}(x) - f'_{\theta_1}(x)][e^{-f_0(x)} - e^{-f_1(x)}]}{e^{-f_0(x)} + e^{-f_1(x)}} \tag{37}\]

\[
s_{\theta_0}(x) - s_{\theta_0}(x) = f'_{\theta_1}(x) - f'_{\theta_0}(x) \tag{38}\]

\[
s_{\theta_0}(x) \quad s_{\theta_1}(x) = e^{f_{\theta_1}(x)} - f_{\theta_0}(x) \tag{39}\]

For logistic densities, we have \( f_{\theta_0}(x) = \theta^T x, \quad f_{\theta_1}(x) = 0 \) and \( Z_0 = 1 + e^{-\theta^T x} \).

\[
s_{\theta_0} = -\frac{\theta'}{1 + e^{-\theta^T x}} \quad s_{\theta_1} = \frac{\theta' e^{-\theta^T x}}{1 + e^{-\theta^T x}} \tag{40}\]

and

\[
s_{\theta_0}(x) + s_{\theta_0}(x) = -\frac{\theta'(1 - e^{-\theta^T x})}{1 + e^{-\theta^T x}} \quad s_{\theta_0}(x) - s_{\theta_0}(x) = -\theta' \frac{s_{\theta_0}(x)}{s_{\theta_1}(x)} = -e^{\theta^T x} \tag{41}\]

where \( \theta' \) equals \( \theta \) but with intercept removed after differentiation.

As a comparison to score function, the gradient of the densities are:

\[
\nabla_x p_{\theta_0}(x) = \frac{\nabla_x f_{\theta_1}(x) - \nabla_x f_{\theta_0}(x)}{[1 + e^{f_{\theta_0}(x)} - f_{\theta_1}(x)][1 + e^{f_{\theta_1}(x)} - f_{\theta_0}(x)]} \tag{42}\]
and \( \nabla_x p_\theta(x) = -\nabla_x p_\theta_0(x) \), as \( p(y = 1|x) = 1 - p(y = 0|x) \) is satisfied everywhere. Logistic density, for example, gives:

\[
\nabla_x p_\theta_0(x) = \frac{\theta}{(1 + e^{\theta^T x})(1 + e^{-\theta^T x})}
\]

The scores and gradients of the learned logistic densities are compared in the simulated 1D and 2D Gaussian data cases in Fig.8 and Fig.9, respectively.

**Appendix B  Gaussian scores and separality**

Extending the discussion in Section 4, here we take a closer look at the binary scenario where the two classes data are Gaussian distributed. We already know its score function \( s(x|\mu, \Sigma) = \Sigma^{-1}(\mu - x) \). Following the generative decision rule (Eq.1b), for a test point \( x = (x_1, x_2) \), a naive classifier would associate it with the class with higher density value, maybe with a soft margin \( \gamma = 0 \) introduced such that \( \hat{y} = 0 \) if \( p_\theta_1(x) - p_\theta_0(x) \geq \gamma \). Here for simplicity we just use \( \gamma = 0 \). The decision boundary is then determined by \( p_\theta_0(x^*) = p_\theta_1(x^*) \) where we denote the points on the decision boundary as \( x^* \).

Substituting the multivariate Gaussian density (Eq.8) into the equal density condition, we arrive at the boundary equation (BE):

\[
BE(x^*) = \log p_\theta_1(x^*) - \log p_\theta_0(x^*)
\]

\[
= -\frac{1}{2} x^T (\Sigma_1^{-1} - \Sigma_0^{-1}) x + (\mu_1^T \Sigma_1^{-1} - \mu_0^T \Sigma_0^{-1}) x
\]

\[
+ \frac{1}{2} \log(|\Sigma_0|/|\Sigma_1|) + \frac{1}{2} (\mu_0^T \Sigma_0^{-1} \mu_0 - \mu_1^T \Sigma_1^{-1} \mu_1) = 0
\]

(44)

where \( |\cdot| \) denotes matrix determinant. This quadratic boundary curve or surface is also used in QDA [28]. The first-order derivative of \( BE(x) \):

\[
\frac{\partial BE(x)}{\partial x} = (\Sigma_0^{-1} - \Sigma_1^{-1}) x + \Sigma_0^{-1} \mu_1 - \Sigma_1^{-1} \mu_0
\]

(45)

Again, Eq.45 can be used to locate the roots of Eq.44 in a Newton-Raphson scheme. The boundary between two one-dimensional Gaussian densities issued by Eq.44 for example, is:

\[
(1/\sigma_1^2 - 1/\sigma_0^2)x^2 - 2(\mu_1/\sigma_1^2 - \mu_0/\sigma_0^2)x - \log(\sigma_0^2/\sigma_1^2) - (\mu_0^2/\sigma_0^2 - \mu_1^2/\sigma_1^2) = 0
\]

(46)

with the discriminative points \( x = (\mu_1\sigma_0 - \mu_0\sigma_1)/(\sigma_0 - \sigma_1) \) and \( x = (\mu_1\sigma_0 + \mu_0\sigma_1)/(\sigma_0 + \sigma_1) \), assuming \( \mu_1 \geq \mu_0 \).

Further, if the two Gaussian classes share the same covariance, i.e. \( \Sigma_1 = \Sigma_2 = \Sigma \), Eq.44 reduces to:

\[
(\mu_1^T - \mu_0^T)\Sigma^{-1} x^* = \frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0)
\]

(47)
which is a linear boundary. This is the discriminator for LDA \[28\]. In one-dimensional case, two Gaussians with same standard deviation will meet at the equal probability point \(x^* = (\mu_1 + \mu_2)/2\).

As with this naive classifier, for each class, all points located beyond \(x^*\) are mis-labelled. The mis-classification rate is controlled by the probability that those points are from the beyond-boundary arena, which is:

\[
p(x > x^*) = 1 - \Phi[L^{-1}(x^* - \mu)]
\]

(48)

here \(x > x^*\) denotes that \(x\) is beyond the boundary (with any coordinate exceeding the boundary), \(\Phi(z) = \int_{-\infty}^{z} N(x|0, I)dx\) is the CDF of standard normal, and \(L\) is the lower triangular matrix in covariance cholesky decomposition \(\Sigma = LL^T\). The total probability of mis-classifying an unknown point \(x\) is \(p_{mis}(x) = p_0(x > x^*) + p_1(x > x^*)\), with only one term of the summation is non-zero at any time.

If we have learned the score functions \(s_{\theta_0}\) and \(s_{\theta_1}\), we can also construct BE by integration. Here are describe the details of finding the boundary points starting from scores, as used before in constructing the score-based decision boundary in Fig\[25\]. The first step is to recover the density ratio from score functions. As the mapping between score function and density is a differential equation, we can simply inverse it by integration. Then we apply the simple generative decision rule of equal density to find the boundary. We start by writing the (log) density ratio differential equation:

\[
s_{\theta_1}(x) - s_{\theta_0}(x) = \nabla_x \log p_{\theta_1}(x) - \nabla_x \log p_{\theta_0}(x) = \nabla_x \log \frac{p_{\theta_1}(x)}{p_{\theta_0}(x)}
\]

(49)

By integration we can solve for the pdf ratio function \(R(x)\):

\[
R(x) = \frac{p_{\theta_1}(x)}{p_{\theta_0}(x)} = Ce^{\int_{\Omega}[s_{\theta_1}(x) - s_{\theta_0}(x)]dx}
\]

(50)

where the integration over the intersection support domain \(\Omega\) can be finite or improper. Unfortunately, neither the differentiation (Eq\[49\]) nor integration (Eq\[50\]) form informs any corresponding equality between densities and scores: if \(s_{\theta_1}(x) = s_{\theta_0}(x)\), from Eq\[49\] we have \(p_{\theta_1}(x) = C p_{\theta_0}(x)\), i.e. the two densities can off-set by a multiplicative constant; if \(p_{\theta_1}(x) = p_{\theta_0}(x)\), we have the equation for a normal decision boundary \(\int_{\Omega}[s_{\theta_1}(x) - s_{\theta_0}(x)]dx + \log C = 0\) (obtained by setting \(R(x^*) = 1\), which doesn’t give any hints on the local (e.g. at particular points) equality between \(s_{\theta_0}(x)\) and \(s_{\theta_1}(x)\).

In practice, we are done because we have the learned \(s_{\theta_0}(x)\) and \(s_{\theta_1}(x)\) from data (optimally in parametric form), and \(C\) can be determined by empirically estimate an initial point density ratio \(\hat{p}_{\theta_1}(x_0)/\hat{p}_{\theta_0}(x_0)\). They together can be substituted into \(\int_{\Omega}[s_{\theta_1}(x) - s_{\theta_0}(x)]dx + \log C = 0\) to find \(x^*\), which was used in Fig\[25\]. However, given the analytical form of Gaussian densities, we shall be able to further derive an analytical formula of \(C\), which can provides convenient sample-based estimate of \(C\) without using an initial point to evaluate.
\[ \hat{p}_{\theta_1}(x_0)/\hat{p}_{\theta_0}(x_0). \] To this regard, we first substitute the analytical score function \( s(x) = \Sigma^{-1}(\mu - x) \) into \( R(x) \):

\[
R(x) = Ce^{-\frac{1}{2}x^T(\Sigma_1^{-1} - \Sigma_0^{-1})x + (\mu_1^T \Sigma_1^{-1} - \mu_0^T \Sigma_0^{-1})x}
\] (51)

Comparing the above formula with exact density ratio (referencing Eq.8) gives:

\[
\log C = \frac{1}{2} \log(|\Sigma_0|/|\Sigma_1|) + \frac{1}{2}(\mu_0^T \Sigma_0^{-1} \mu_0 - \mu_1^T \Sigma_1^{-1} \mu_1)
\] (52)

With Eq.52, under the assumption of Gaussianalities, we can conveniently estimate the constant \( C \) via Eq.52 using sample-based means and covariances.