



# Machine Learning

2024 Fall, Final Review

**Xiangchen Tian**

Jan 2, 2025



清华大学  
Tsinghua University



# Table of Contents

## 1 SVM

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Support Vector Machine(a.k.a. SVM)

1 SVM

- 支持向量机是一种监督学习算法
- 通过构造极大边距超平面以更好泛化
- 两种常见形式：Hard-SVM 和 Soft-SVM



## Hard-SVM

### 1 SVM

- 设训练数据集  $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ，其中  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \{-1, 1\}$ 。
- 线性可分数据集：存在一个超平面  $(\mathbf{w}, b)$  使  $\forall i, y_i = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ 。
- Hard-SVM 只能处理线性可分数据集。

算法：

#### Hard-SVM

**input:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$

**solve:**

$$(\mathbf{w}_0, b_0) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \|\mathbf{w}\|^2 \text{ s.t. } \forall i, y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad (15.2)$$

$$\textbf{output: } \hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \quad \hat{b} = \frac{b_0}{\|\mathbf{w}_0\|}$$

分析：固定标度最大化边距，相当于固定边距最小化标度；最后归一化还原成固定标度最大化边距的结果。



# Soft-SVM

1 SVM

- Soft-SVM 可以处理非线性可分数据集。
- 思想：放宽限制为  $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$

算法：

Soft-SVM

**input:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$

**parameter:**  $\lambda > 0$

**solve:**

$$\min_{\mathbf{w}, b, \xi} \left( \lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^m \xi_i \right) \quad (15.4)$$

s.t.  $\forall i, y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$  and  $\xi_i \geq 0$

**output:**  $\mathbf{w}, b$



# hinge loss

1 SVM

- 定义 hinge loss  $\ell^{\text{hinge}}(x) = \max \{0, 1 - x\}$
- 定义  $L_S^{\text{hinge}}((\mathbf{w}, b)) = \frac{1}{m} \sum_{i=1}^m \ell^{\text{hinge}}(y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b))$
- 注意到 (15.4) 式等价于

$$\min_{\mathbf{w}, b} \left( \lambda \|\mathbf{w}\|^2 + L_S^{\text{hinge}}((\mathbf{w}, b)) \right)$$

这就是标准的 regularized loss minimization problem，其中  $\lambda \|\mathbf{w}\|^2$  是  $\ell_2$  正则化。

- 因此 Soft-SVM 等价于一个 hinge loss、 $\ell_2$  正则化的优化问题。



# duality

## 1 SVM

- 以 Hard-SVM 为例 (忽略  $b$ ), 定义

$$g(\mathbf{w}) = \max_{\alpha \in \mathbb{R}^m, \alpha \geq 0} \sum_{i=1}^m \alpha_i \cdot (1 - y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle)) = \begin{cases} 0 & \text{if } \forall i, y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \\ +\infty & \text{otherwise} \end{cases}$$

则 (15.2) 式等价于

$$\begin{aligned} & \min_{\mathbf{w}} \left( \|\mathbf{w}\|^2 + g(\mathbf{w}) \right) \\ &= \min_{\mathbf{w}} \max_{\alpha \in \mathbb{R}^m, \alpha \geq 0} \left( \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i \cdot (1 - y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle)) \right) = p^* \\ &\geq \max_{\alpha \in \mathbb{R}^m, \alpha \geq 0} \min_{\mathbf{w}} \left( \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i \cdot (1 - y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle)) \right) = d^* \end{aligned} \quad (1)$$

- 在 Hard-SVM 中, 由于特殊条件的满足,  $p^* = d^*$ , 所以可以通过求解对偶问题来求解原问题。



# duality

1 SVM

- 对内部  $\mathbf{w}$  取最小值，得到  $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$
- 带回对偶问题(1)式，得到

$$\max_{\alpha \in \mathbb{R}^m, \alpha \geq 0} \left( \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

- 关键：只与  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  有关，而不与  $\mathbf{x}_i$  有关，这是核技巧的基础。



# kernel method

## 1 SVM

- 思想：由于很多问题在原空间中线形不可分，希望先将原数据点映射到一个（更高维）空间中，然后在更高维空间中执行 SVM 算法。
- 算法：
  - 设原数据点定义域为  $X$ , 映射函数为  $\phi : X \rightarrow F$ , 其中  $F$  是特征空间 (feature space)。
  - 给定原带标签数据集  $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ , 构造像数据集  $\hat{S} = \{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \dots, (\phi(\mathbf{x}_m), y_m)\}$ 。
  - 在  $\hat{S}$  上执行 SVM 算法, 得到超平面  $(\mathbf{w}, b)$ , 对应一个线性分类器  $h : \psi(\mathbf{x}) \mapsto y$ 。
  - 预测原空间中 test set example  $\mathbf{x}$  为  $h(\psi(\mathbf{x}))$



# kernel

## 1 SVM

- kernel 是 feature space 中的 inner product。
- 给定 embedding  $\phi : X \mapsto F$ , 定义 kernel function

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

- $K$  表征了  $\mathbf{x}, \mathbf{x}'$  的相似度



# kernel trick

## 1 SVM

- 很多 SVM 问题都可以总结为以下 general 问题的实例：

$$\min_{\mathbf{w}} (f(\langle \mathbf{w}, \mathbf{x}_1 \rangle, \langle \mathbf{w}, \mathbf{x}_2 \rangle, \dots, \langle \mathbf{w}, \mathbf{x}_m \rangle) + R(\|\mathbf{w}\|_2))$$

- Example 1: Soft-SVM,  $R(a) = \lambda a^2$ ,  $f(a_1, a_2, \dots, a_m) = \frac{1}{m} \sum_{i=1}^m \max \{0, 1 - y_i a_i\}$
- Example 2: Hard-SVM,  $R(a) = a^2$ ,  
$$f(a_1, a_2, \dots, a_m) = \begin{cases} 0 & \text{if } \exists b \text{ s.t. } y_i(a_i + b) \geq 1 \\ +\infty & \text{otherwise} \end{cases}$$
- 而  $\mathbf{w} \in \text{span}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$  ([reference](#), Theorem 16.1) , 因而事实上不会涉及  $\mathbf{x}_i$  的具体值，只会涉及到  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ 。
- 因此，只需要知道 kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$ ，就能隐式在高维特征空间中执行 SVM 算法。



# characterizing kernel function

1 SVM

- 一个良定义的 kernel function 必须对应一个合理的 feature space embedding  $\phi$ 。  
自然的问题：什么样的 kernel function 是合理的？

## Mercer's Theorem

一个对称函数  $K : X \times X \rightarrow \mathbb{R}$  (对称:  $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}), \forall \mathbf{x}, \mathbf{x}' \in X$ ) 可实现为某特征空间中的内积，当且仅当  $K$  对应的 Gram matrix 是半正定的。

- Gram matrix  $G : G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ .
- 证明思路: ( $\Rightarrow$ )straight forward, ( $\Leftarrow$ ) 采用构造法，构造出一个特征空间和一个内积，使得这个内积对应的 kernel function 为  $K$ 。



# Table of Contents

## 2 Decision Trees

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Decision Trees

## 2 Decision Trees

### Decision Tree

一个决策树是一个 Boolean Function  $f : \mathbb{F}_2^n \rightarrow \mathbb{R}$  的表示方法。它是一个含根二叉树，其中内部节点由某个  $i \in [n]$  来标记，每个内部节点的出边被标记为 0 或 1，每个叶子节点都有一个实数值，且要求没有  $i \in [n]$  在一条从根到叶节点的路径上出现多于一次。

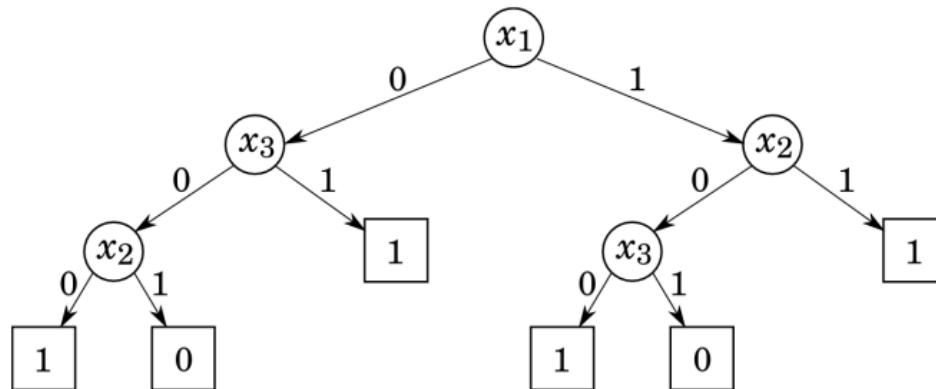
- 称决策树的叶节点总个数为决策树的大小 (size)  $s$ ，称决策树根到叶节点路径的最大长度为决策树的深度 (depth)  $d$ 。
- 根据定义，每个决策树都对应一个有  $n$  个变量的布尔函数。



# Decision Trees Example

## 2 Decision Trees

Example:



Note: 这棵树事实上对应函数  $\text{Sort}_3$ , 其中  $\text{Sort}_3(x_1, x_2, x_3) = 1$  当且仅当  $x_1 \geq x_2 \geq x_3$  或  $x_1 \leq x_2 \leq x_3$ 。



# Theoretical Guarantee

## 2 Decision Trees

### Theorem: Convert decision tree to low degree sparse function

对任意有  $s$  个叶节点的决策树  $T$ , 存在一个 degree 为  $\log(s/\epsilon)$ ,  $L_0$ -norm(sparsity) 为  $s^2/\epsilon$  的布尔函数  $h$  能够  $4\epsilon$ -approximate  $T$ 。

证明思路:

- 将决策树  $T$  截断到深度为  $\log(s/\epsilon)$ , 最大误差为  $\epsilon$ 。
- 截断后的树  $T'$  能用一个  $L_1(f) \leq s$ ,  $\deg(f) = \log(s/\epsilon)$  的布尔函数  $f$  严格表示。
- 上述满足  $L_1(f) \leq s$ ,  $\deg(f) = \log(s/\epsilon)$  的布尔函数  $f$  能用另一个满足  $L_0(h) \leq s^2/\epsilon$ ,  $\deg(h) = \log(s/\epsilon)$  的布尔函数  $h$  来  $\epsilon$ -approximate 表示。

$$\begin{aligned} \text{综上, } \|T - f\|^2 &\leq \epsilon, \|f - h\|^2 \leq \epsilon \\ \Rightarrow \|T - h\|^2 &\leq 2\|T - f\|^2 + 2\|f - h\|^2 \leq 4\epsilon \end{aligned}$$



# Practical Algorithms 1

## 2 Decision Trees

由于使用上述定理提供的方法求 low degree approximate function  $h$  要求已知决策树内部结构，但实际中决策树是黑箱，只能通过多次输入查看输出得到决策树的信息。实际中采用以下几种采样算法：

- LMN: 设  $T$  对应布尔函数  $f$ , 均匀采样  $m$  个  $f$  定义域内的点  $\{x_i, i \in [m]\}$ , 计算  $f(x_i)$ , 对每个  $|S| \leq \log(s/\epsilon)$  的  $S$  估计傅立叶系数  $\hat{f}(S) \approx \frac{1}{m} \sum_{i=1}^m f(x_i) \chi_S(x_i)$ 。最后返回

$$h = \sum_{S: |S| \leq \log(s/\epsilon)} \hat{f}(S) \chi_S$$

- Harmonica: 这个问题本质上是求一个在傅立叶基下稀疏的布尔函数，可以用 compress sensing 范式求解。

$$f = \begin{matrix} \text{entry } (i) \text{ corresponds to} \\ f(x_i) = \sum_S \alpha_S \psi_S(x_i), \\ \text{the } i\text{-th measurement} \end{matrix} \quad \begin{matrix} \text{columns } S \subseteq [n] \\ \text{for all } |S| \leq d \end{matrix} \quad \times \quad \begin{matrix} \text{entry } (i, S) \\ = \psi_S(x_i) \end{matrix} \quad \begin{matrix} \alpha \text{ has entry } \alpha_S \\ \text{for all } |S| \leq d \end{matrix}$$

rows corresponds to  $x_i \in \{-1, 1\}^n$



# Practical Algorithms 2

## 2 Decision Trees

现实问题中的决策树：给定若干样例，希望从样例中学习决策树。

Example:

Past trend	Open interest	Trading volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up



# Practical Algorithms 2

## 2 Decision Trees

与上面的 approach 不同，另一种方法是递归选取最重要的变量作为内部节点划分样例。

算法：

```
function LEARN-DECISION-TREE(examples, attributes, parent examples) returns 一棵树
    if examples不为空 then return PLURALITY-VALUE(parent examples)
    else if 所有examples有相同的分类 then return 分类
    else if attributes为空 then return PLURALITY-VALUE(examples)
    else
         $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$ 
        tree  $\leftarrow$  一个以测试A为根的新的决策树
        for each A中的值v do
            exs  $\leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$ 
            subtree  $\leftarrow$  LEARN-DECISION-TREE (exs, attributes}-A, examples)
            将一个带有标签 (A = v) 和子树 subtree的分支加入tree
        return tree
```

解释：每次根据某种准则选择最重要的变量 *i*（若某变量的取值几乎决定了最终的类别是什么，说明这个变量较重要）



# Gini Index

## 2 Decision Trees

- Gini Index 是一种判断某个变量是否重要的一种准则。
- 定义：对变量  $A$ ,  $\text{Gini}(A) = \sum_a p(A=a)\text{Gini}(a)$ , 其中  $\text{Gini}(a) = 1 - \sum_i p_i^2$  (详见课件例子)
- $\text{Gini}(a)$  越小表示区分度越好 (如果一个变量  $A$  取正时所有结果都是正, 一个变量取负时所有结果都是负, 则  $\text{Gini}(A) = 0$ ), 故每次划分变量时选择  $\text{Gini}(A)$  最小的变量  $A$ 。
- 其他准则：Information Gain



# Table of Contents

## 3 Boosting

- ▶ SVM
- ▶ Decision Trees
- ▶ **Boosting**
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Boosting

## 3 Boosting

- 思想：希望能有一种方法能够聚合多个弱学习器（每个弱学习器可以只比随机猜测表现好一点），使得整体学习器的性能更好。
- 一次性学习强学习器可能计算复杂度上承担不起，但每个弱学习器的计算复杂度较低，聚合多个弱学习器的计算复杂度可以接受。



# AdaBoost (a.k.a Adaptive Boosting)

## 3 Boosting

- AdaBoost 是最典型的 boosting 算法。
- AdaBoost 思想：在简单假设类上构建线性预测期作为更强大的假设类。

算法：

**AdaBoost**

**input:**

training set  $S = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$

weak learner WL

number of rounds  $T$

**initialize**  $\mathbf{D}^{(1)} = (\frac{1}{m}, \dots, \frac{1}{m})$ .

**for**  $t = 1, \dots, T$ :

    invoke weak learner  $h_t = \text{WL}(\mathbf{D}^{(t)}, S)$

    compute  $\epsilon_t = \sum_{i=1}^m D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$

    let  $w_t = \frac{1}{2} \log \left( \frac{1}{\epsilon_t} - 1 \right)$

    update  $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^m D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$  for all  $i = 1, \dots, m$

**output** the hypothesis  $h_s(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T w_t h_t(\mathbf{x}) \right)$ .



## AdaBoost Analysis

### 3 Boosting

#### Theorem: AdaBoost Training Error Upper Bound

$S$  是一个训练集，假设在 AdaBoost 的每一次迭代中，弱分类器返回的假设满足  $\epsilon_t \leq \frac{1}{2} - \gamma$ 。则 AdaBoost 输出的最终假设的训练误差满足以下不等式：

$$L_S(h_s) = \frac{1}{m} \sum_{i=1}^m \mathbb{1}[h_s(x_i) \neq y_i] \leq \exp(-2\gamma^2 T)$$

证明：对于每一轮  $t$ ，记  $f_t = \sum_{p < t} w_p h_p$ ，因此 AdaBoost 的输出为  $f_T$ 。此外，记

$$Z_t = \sum_{i=1}^m \exp(-y_i f_t(x_i))$$



# AdaBoost Analysis

## 3 Boosting

注意到对于任何假设都有  $\mathbb{1}[h(x) \neq y] \leq \exp(-y h(x))$ 。因此，训练误差满足：

$$L_S(f_T) \leq Z_T$$

所以我们只需证明  $Z_T \leq \exp(-2\gamma^2 T)$ 。为了 bound 住  $Z_T$ ，我们将其重写为：

$$Z_T = \frac{Z_T}{Z_{T-1}} \cdot \frac{Z_{T-1}}{Z_{T-2}} \cdots \frac{Z_2}{Z_1} \cdot \frac{Z_1}{Z_0}$$

其中我们使用了  $Z_0 = 1$ ，因为  $f_0 \equiv 0$ 。现在我们只需证明对于每一轮  $t$ ：

$$\frac{Z_{t+1}}{Z_t} \leq \exp(-2\gamma^2)$$

为了证明上式，首先通过简单的归纳推理可知对于所有的  $t$  和  $i$  都有：

$$D_i^{(t+1)} = \frac{\exp(-y_i f_t(x_i))}{\sum_{j=1}^m \exp(-y_j f_t(x_j))}$$



# AdaBoost Analysis

## 3 Boosting

因此：

$$\begin{aligned}\frac{Z_{t+1}}{Z_t} &= \frac{\sum_{i=1}^m \exp(-y_i f_{t+1}(x_i))}{\sum_{j=1}^m \exp(-y_j f_t(x_j))} = \frac{\sum_{i=1}^m \exp(-y_i f_t(x_i)) \exp(-y_i w_{t+1} h_{t+1}(x_i))}{\sum_{j=1}^m \exp(-y_j f_t(x_j))} \\ \implies \frac{Z_{t+1}}{Z_t} &= \exp(-w_{t+1}) (1 - \epsilon_{t+1}) + \exp(w_{t+1}) \epsilon_{t+1} = 2 \sqrt{\epsilon_{t+1} (1 - \epsilon_{t+1})}\end{aligned}$$

根据我们的假设， $\epsilon_{t+1} \leq \frac{1}{2} - \gamma$ 。由于函数  $g(a) = a(1 - a)$  在区间  $[0, \frac{1}{2}]$  上是单调递增的，我们得到：

$$2 \sqrt{\epsilon_{t+1} (1 - \epsilon_{t+1})} \leq 2 \sqrt{\left(\frac{1}{2} - \gamma\right) \left(\frac{1}{2} + \gamma\right)} \leq \sqrt{1 - 4\gamma^2}$$

因此， $\frac{Z_{t+1}}{Z_t} \leq \exp(-2\gamma^2)$ 。

□



# Table of Contents

## 4 PCA

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ **PCA**
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Principal Component Analysis (a.k.a. PCA)

4 PCA

- PCA 是一种降维技术，将高维空间中的数据映射到低维空间。
- 详细请参考计算机与人工智能应用数学
- power method：一种高效求解最大本征值和对应的本征向量的方法



# Table of Contents

## 5 Nearest Neighbor and LSH

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Nearest Neighbor

## 5 Nearest Neighbor and LSH

- Nearest Neighbor 是一种非参数化模型。(数据集就是参数)
- 直接用遍历所有点的方式找到邻居的方法非常耗时，希望有一种数据结构能够高效返回近似最近邻
- LSH algorithm



# LSH

## 5 Nearest Neighbor and LSH

Formally, 我们希望解决以下最近邻高效查找问题:

### $\epsilon$ -NNS Problem

给定在 normed space  $\ell_p^d$  (距离定义为  $\ell_p$ -norm 的  $d$  维空间) 中的点集  $P$ , 处理  $P$  使任给查询点  $q$ , 数据结构能高效返回一个点  $p \in P$  满足  $d(q, p) \leq (1 + \epsilon)d(q, P)$ , 其中  $d(q, P)$  是  $q$  到  $P$  中最近点的距离。

思想:

- 使用 hash 方法, 希望距离近的点更有可能被分在同一组中。(即, 希望同一个 hash table 组中包含许多相互靠近的点, 但也可能包含距离较远的点)
- 单独一个 hash table 随机性太大, 因此创建一个 family of hash functions。



# LSH Family

## 5 Nearest Neighbor and LSH

### LSH Family: $(R, cR, P_1, P_2)$ -sensitive

一个 family  $H$  被称作是  $(R, cR, P_1, P_2)$ -sensitive 的，若对任意两点  $p, q \in \mathbb{R}^d$ ，有：

- 若  $d(p, q) \leq R$ ，则  $\Pr_{h \in H}[h(p) = h(q)] \geq P_1$
- 若  $d(p, q) \geq cR$ ，则  $\Pr_{h \in H}[h(p) = h(q)] \leq P_2$

其中对  $h$  从  $H$  中均匀随机选择 (uniformly at random)。

- 若希望能有区分度，要求  $P_1 > P_2$ 。
- 思想：如果两个点很近 ( $d \leq R$ )，则  $H$  中有很多 hash 函数能够将这两个点映射到同一个值；如果两个点很远 ( $d \geq cR$ )，则  $H$  中很少的 hash 函数能够将这两个点映射到同一个值。



# Example 1: LSH for Hamming Distance in $\{0, 1\}^d$ space

## 5 Nearest Neighbor and LSH

定义 LSH family 为

$$H = \left\{ h_i : \{0, 1\}^d \rightarrow \{0, 1\} \mid h_i(x) = x_i, \forall i \in [d] \right\}$$

注意到

- 若  $d_H(x, y) \leq R$ , 则  $x, y$  最多有  $R$  位不同, 因此至少有  $d - R$  个 hash 函数能够将  $x, y$  映射到同一个值  $\Rightarrow \Pr_{h \in H}[h(p) = h(q)] \geq 1 - \frac{R}{d} = P_1$ 。
- 若  $d_H(x, y) \geq cR$ , 则  $x, y$  至少有  $cR$  位不同, 因此至多有  $d - cR$  个 hash 函数能够将  $x, y$  映射到同一个值  $\Rightarrow \Pr_{h \in H}[h(p) = h(q)] \leq 1 - \frac{cR}{d} = P_2$ 。

因此, family  $H$  是  $(R, cR, 1 - \frac{R}{d}, 1 - \frac{cR}{d})$ -sensitive 的。



## Example 2: LSH for $\ell_p$ Distance in $\mathbb{R}^d$ space ( $p \in (0, 2]$ )

### 5 Nearest Neighbor and LSH

定义 LSH family 为

$$H = \left\{ h_{r,b} : \mathbb{R}^d \rightarrow \mathbb{R} \mid h_{r,b}(x) = \left\lfloor \frac{r \cdot x + b}{w} \right\rfloor \right\} \quad (2)$$

其中  $r \sim p - \text{stable Distribution}$ ,  $b \sim \text{Uniform}[0, w]$ 。

- 对  $p = 1$ , stable distribution 是 Cauchy distribution:

$$f_1(x) = \frac{1}{\pi(1+x^2)}$$

- 对  $p = 2$ , stable distribution 是 Gaussian distribution:

$$f_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

- 可以证明, 按 (2) 式定义的 family 满足  $(R, cR, P_1, P_2)$ -sensitive。



## Example 2: LSH for $\ell_p$ Distance in $\mathbb{R}^d$ space ( $p \in (0, 2]$ )

### 5 Nearest Neighbor and LSH

对  $p$ -stable distribution 的更详细解释:

- $p$ -stable distribution 定义为

#### Definition of $p$ -Stable Distribution

A distribution  $D$  over  $\mathbb{R}$  is called  $p$ -stable, if there exists  $p \geq 0$  such that for any  $n$  real numbers  $v_1, \dots, v_n$  and i.i.d. variables  $X_1, \dots, X_n$  with distribution  $D$ , the random variable

$$\sum_i v_i X_i$$

has the same distribution as the variable

$$\left( \sum_i |v_i|^p \right)^{1/p} X,$$

where  $X$  is a random variable with distribution  $D$ .

- 按照如上定义, 在计算  $\Pr_{h \in H}[h(p) = h(q)]$  时遇到的  $r \cdot (p - q)$  的分布等于  $\|p - q\|_p \cdot r_1$ , 其中  $r_1$  服从一维  $p$ -stable distribution。这样  $\Pr_{h \in H}[h(p) = h(q)]$  容易计算。



# LSH Algorithm

## 5 Nearest Neighbor and LSH

算法：

### Preprocessing:

1. Choose  $L$  functions  $g_j, j = 1, \dots, L$ , by setting  $g_j = (h_{1,j}, h_{2,j}, \dots, h_{k,j})$ , where  $h_{1,j}, \dots, h_{k,j}$  are chosen at random from the LSH family  $\mathcal{H}$ .
2. Construct  $L$  hash tables, where, for each  $j = 1, \dots, L$ , the  $j^{\text{th}}$  hash table contains the dataset points hashed using the function  $g_j$ .

### Query algorithm for a query point $q$ :

1. For each  $j = 1, 2, \dots, L$ 
  - i) Retrieve the points from the bucket  $g_j(q)$  in the  $j^{\text{th}}$  hash table.
  - ii) For each of the retrieved point, compute the distance from  $q$  to it, and report the point if it is a correct answer ( $cR$ -near neighbor for Strategy 1, and  $R$ -near neighbor for Strategy 2).
  - iii) (optional) Stop as soon as the number of reported points is more than  $L'$ .

注意： $g_j(p) = (h_{j_1}(p), h_{j_2}(p), \dots, h_{j_k}(p))$  是新的 hash function，通过组合 LSH family 中若干随机性比较大的 hash function 组成。在 Yang Yuan 的版本中， $L' = 2L + 1$ ，Yang Yuan 证明了只用查找最多  $2L + 1$  个点就一定能保证 report 一个  $cR$ -near 的点。



# LSH Algorithm Analysis

## 5 Nearest Neighbor and LSH

### Theorem

使用以上算法查找  $q$  的近邻点，若存在  $p^* \in P$  使  $p^* \in B(q, R)$ ，则以上算法返回一个与  $q$   $cR$ -near 的点的概率至少为  $\frac{1}{2} - \frac{1}{e}$ 。

证明思路：

- 所有与  $q$  至少有一个  $g_j$  相等、且不与  $q$   $cR$ -near 的点  $p$ （换句话说，就是至少有一个 hash 表， $q$  和  $p$  在同一个桶中，因为只有这样算法才可能 report 这个点  $p$ ）的个数  $\leq 2L$  的概率  $\geq \frac{1}{2}$
- 另一方面，若  $p^* \in B(q, R)$ ，则  $p^*$  与  $q$  至少有一个  $g_j$  相等的概率  $\geq 1 - \frac{1}{e}$ 。
- 若两者同时成立（概率  $\geq \frac{1}{2} - \frac{1}{e}$ ），则只查询  $L' = 2L + 1$  个点时一定会返回一个  $cR$ -near 的点。（详见原始论文）



# Table of Contents

## 6 Metric Learning

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Metric Learning

## 6 Metric Learning

- 核心思路：用 neural network 训练出一个 good feature space, 在此空间中有更好的 nearest neighbor structure.
- Example: 图片分类问题中旋转物体视角、改变颜色、图片伸缩裁剪可能不会改变类别，但在像素空间中这些点相距非常远。
- 两种算法：NCA, LMNN



- 希望学习一个函数  $f$  从原空间映射到有良好最近邻结构的 feature space.
- 定义原空间中两点  $x_i, x_j$  的相似度为：

$$p_{ij} = \frac{\exp(-\|f(x_i) - f(x_j)\|^2)}{\sum_{k \neq i} \exp(-\|f(x_i) - f(x_k)\|^2)}, i \neq j$$

$$p_{ii} = 0$$

- 记  $C_i = \{j | c_i = c_j\}$  为与  $i$  有相同 label 的下标集合， $P_i = \sum_{j \in C_i} p_{ij}$  为  $x_i$  与其他与  $x_i$  有相同 label 的元素的总相似度。
- Loss function:  $L(A) = \sum_i P_i$ , 用于训练  $f$  模型中的参数。



LMNN

6 Metric Learning

- TODO



# Table of Contents

## 7 Clustering

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Clustering

## 7 Clustering

Clustering 任务：输入元素集合  $X$  及一个距离函数  $d : X \times X \rightarrow \mathbb{R}_+$ （满足  $d(x, y) = d(y, x)$  和  $d(x, x) = 0, \forall x, y \in X$ ），Cluster 算法应该输出一个划分 (partition)  $C = \{C_1, C_2, \dots, C_k\}$ ，其中  $C_i \subseteq X$ ，且  $\bigcup_{i=1}^k C_i = X$ 。典型算法：

- Linkage-based clustering
- K-means
- Spectral clustering



# K-means: Lloyd's method

## 7 Clustering

算法：

### $k$ -Means

**input:**  $\mathcal{X} \subset \mathbb{R}^n$  ; Number of clusters  $k$

**initialize:** Randomly choose initial centroids  $\mu_1, \dots, \mu_k$

**repeat until convergence**

$\forall i \in [k]$  set  $C_i = \{\mathbf{x} \in \mathcal{X} : i = \operatorname{argmin}_j \|\mathbf{x} - \mu_j\|\}$

(break ties in some arbitrary manner)

$\forall i \in [k]$  update  $\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$

Example: 参考上课 slides。



# K-means: Lloyd's method

## 7 Clustering

- 重要性质：每一次迭代过程中，objective function

$$G_{\text{K-means}}((X, d), (C_1, C_2, \dots, C_k)) = \min_{\mu_1, \mu_2, \dots, \mu_k} \sum_{i=1}^k \sum_{x \in C_i} d(x, \mu_i)^2$$

不会增加。

- Note: Lloyd's method 是最常用的一种 K-means 算法，但 K-means 算法还有很多其他变种。



# Spectral Clustering

## 7 Clustering

- 思想：用 similarity graph  $G$  表示集合  $X = \{x_1, x_2, \dots, x_m\}$  各点之间的关系。  
 $G = (V, E)$ , 其中  $V = \{x_1, x_2, \dots, x_m\}$ , 每两个顶点之间连接一条权重为  
 $W_{ij} = s(x_i, x_j)$  的边,  $s$  是相似度度量, 例如  $s(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{\sigma^2}\right)$
- Clustering 问题可表述为：希望找到一种 partition, 同一 part 各点之间边权重较大, 不同 part 各点之间边权重较小。
- 最小化  $Cut(C_1, C_2, \dots, C_k) = \sum_{i=1}^k \sum_{r \in C_i, s \notin C_i} W_{rs}$  经常会将单独一个点作为一个 part。
- 解决办法：最小化  $RatioCut(C_1, C_2, \dots, C_k) = \sum_{i=1}^k \frac{\sum_{r \in C_i, s \notin C_i} W_{rs}}{|C_i|}$ , 权衡 part 大小和 part 间边权重。



# Spectral Clustering Analysis

## 7 Clustering

### Definition: Unnormalized Graph Laplacian

一个有向图的 unnormalized graph Laplacian 定义为  $L = D - W$ , 其中  $D$  是 degree matrix,  $D_{ii} = \sum_{j=1}^m W_{ij}$ ,  $W$  是边权重矩阵, 对角元均为 0。

### Lemma

设  $C = \{C_1, C_2, \dots, C_k\}$  是一个 clustering, 定义  $H \in \mathbb{R}^{m \times k}$ :

$$H_{ij} = \frac{1}{\sqrt{|C_j|}} \mathbb{1}[i \in C_j]$$

则  $H^T H = I^{k \times k}$  且  $\text{RatioCut}(C_1, C_2, \dots, C_k) = \text{tr}(H^T L H)$ , 其中  $L$  是 similarity graph 的 unnormalized graph Laplacian.

证明思路: 用  $H$  和  $L$  的定义直接得到。



# Spectral Clustering Analysis

## 7 Clustering

根据上述 Lemma，我们可以将 RatioCut 问题转化为如下问题：

问题 1 (original): 找一个矩阵  $H \in \mathbb{R}^{m \times k}$ ，满足

- $H^T H = I$
- $H_{ij} = 0$  或  $1/\sqrt{|C_j|}$
- 最小化  $\text{tr}(H^T L H)$

以上问题是一个 integer programming problem，无法高效求解。所以对问题进行 relax：

问题 2 (relax): 找一个矩阵  $H \in \mathbb{R}^{m \times k}$ ，满足

- $H^T H = I$
- 最小化  $\text{tr}(H^T L H)$

这时问题变为标准的 PCA 问题，其解为  $L$  的对应前  $k$  个最小本征值的特征向量。



# Spectral Clustering Algorithm

## 7 Clustering

算法：

### Unnormalized Spectral Clustering

**Input:**  $W \in \mathbb{R}^{m,m}$  ; Number of clusters  $k$

**Initialize:** Compute the unnormalized graph Laplacian  $L$

Let  $U \in \mathbb{R}^{m,k}$  be the matrix whose columns are the eigenvectors of  $L$   
corresponding to the  $k$  smallest eigenvalues

Let  $\mathbf{v}_1, \dots, \mathbf{v}_m$  be the rows of  $U$

**Cluster** the points  $\mathbf{v}_1, \dots, \mathbf{v}_m$  using  $k$ -means

**Output:** Clusters  $C_1, \dots, C_K$  of the  $k$ -means algorithm



# Birkhoff's Theorem

7 Clustering

TODO



# Table of Contents

## 8 SimCLR

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



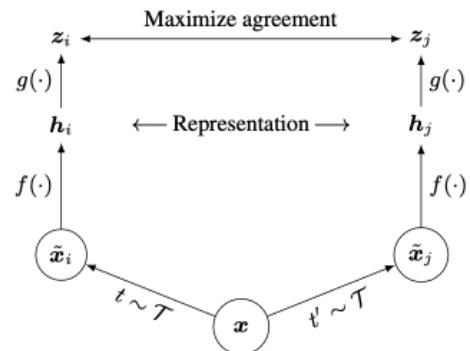
# SimCLR

## 8 SimCLR

- Goal: 用 contrastive learning 方法学习图片的 visual representations。
- 训练集是若干无标签图像，希望对每个图像输出一个 feature vector。

右图中， $f$  是一个神经网络，输入扰动后的图像，输出一个 feature vector。 $g$  是一个 projection head，将 feature vector 投影到一个用于衡量相似度的空间。

$T$  是对图像施加各种可能扰动的集合，例如随机裁剪、旋转、噪声、模糊等。





# SimCLR Algorithm

## 8 SimCLR

算法：

---

**Algorithm 1** SimCLR's main learning algorithm.

---

```
input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .  
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do  
    for all  $k \in \{1, \dots, N\}$  do  
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$   
        # the first augmentation  
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$   
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation  
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection  
        # the second augmentation  
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$   
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation  
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection  
    end for  
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do  
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity  
    end for  
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$   
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$   
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$   
end for  
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 
```

---

- Note: 其中  $\ell(i, j)$  被称为 InfoNCE loss。



# Contrastive Learning is spectral clustering on similarity graph

8 SimCLR

Yang Yuan 组的一篇论文指出：

- 定义 Cross entropy loss  $H_{\pi}^k(Z) = \mathbb{E}_{W_X \sim P(\cdot; \pi)} [\log P(W_Z = W_X; K_Z)]$ 。
- 可以证明：Cross entropy loss 等价于 InfoNCE loss；
- Cross entropy loss 隐式在 similarity graph 上执行了 spectral clustering
- 所以 SimCLR 可以看作是 spectral clustering on similarity graph。

(把他课件上的内容差不多记住，作业那道题会做应该就行了)



# CLIP

## 8 SimCLR

- CLIP 是一种多模态学习方法，可以同时处理图片和文本。
- 细节：给定一个 batch of image-text pairs ( $image, text$ ) 数据，CLIP 用一个 image encoder 和一个 text encoder 分别将 image 和 text 转换为 feature vectors 对，然后用 InfoNCE 作为 loss function 训练 encoder。
- CLIP 也隐式执行了 spectral clustering。



# Table of Contents

## 9 t-SNE

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# t-distributed Stochastic Neighbor Embedding (a.k.a. t-SNE)

9 t-SNE

t-SNE 是一种非线性降维算法，用于将高维数据 ( $\{x_1, x_2, \dots, x_N\}$ ) 映射到低维空间 ( $\{y_1, y_2, \dots, y_N\}$ ，通常  $y_i \in \mathbb{R}^2$  或  $\mathbb{R}^3$ ,  $i \in [N]$ )。

思想：t-SNE 分为两个阶段。

- 首先对每对高维对象分配一个概率，要求相似的对象有更高的概率，不相似的对象分配较低的概率。
- 然后，t-SNE 对低维空间定义一个类似的概率分布，并最小化与之前概率分布之间的 KL 散度。



## SNE

### 9 t-SNE

- SNE 是 t-SNE 的前身。
- 给定  $N$  个高维空间对象的集合  $\{x_1, x_2, \dots, x_N\}$ , 定义

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}, \forall i \neq j$$

$$p_{i|i} = 0$$

显见  $\sum_j p_{j|i} = 1$ 。

- 解释：按照高斯分布对所有  $x_i$  周围的点进行加权得到  $x_j$  对  $x_i$  的重要性。
- 定义  $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$ 。
- $\sigma_i$  的选取：用 bisection method 求解满足  $\text{perp}(P_i) = 2^{H(P_i)}$  = 预先人为设定值的  $\sigma_i$ ，其中  $H(P_i) = -\sum_j p_{j|i} \log_2(p_{j|i})$ 。
- perplexity 可解释为一种平滑的邻居个数度量。 $\text{perp}(P_i)$  越大， $H(P_i)$  越大， $p_{j|i}$  分布越平均， $\sigma_i$  越大，因此邻居越多。



## SNE

### 9 t-SNE

- 原始的 SNE 在低维空间也选取高斯分布进行距离度量。

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}, \forall i \neq j$$

其中  $y_i$  是  $x_i$  在低维空间中的对应点， $q_{ij}$  是低维空间中相似度度量。

- 这存在 crowding problem。解释：“There are several reasons why the pairwise distances in a two-dimensional map cannot faithfully model distances between points on the ten-dimensional manifold.”
  - 在 10 维空间中，可以有 11 个点彼此等距，但二维空间中无法忠实反映这种关系。
  - 以数据点  $x_i$  为中心，半径为  $r$  的球体在 10 维空间中体积按照  $r^{10}$  缩放，因此，如果数据点在 10 维空间中  $x_i$  周围的区域中大致均匀分布，如果我们试图用相同的高斯分布对二维地图中  $x_i$  到其他数据点的距离进行建模，数据点会非常拥挤：二维空间中可用于容纳中等距离数据点的区域与可用于容纳附近数据点的区域相比远远不够大。



## t-SNE

### 9 t-SNE

- 一种解决方案：“Mismatched Tails can Compensate for Mismatched Dimensionalities”
- 这就是 t-SNE 的思想：在高维空间中，使用 Gaussian Distribution 求数据点之间相似性；而在低维空间中，使用 Student t-distribution 求数据点之间相似性。

$$q_{j|i} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|y_i - y_k\|^2)^{-1}}, \forall i \neq j$$

- 与高维空间一样，定义  $q_{ij} = \frac{q_{j|i} + q_{i|j}}{2N}$
- Student t-distribution 的尾部（正比于  $\frac{1}{x^2}$ ）比高斯分布（正比于  $\exp(-x^2)$ ）更“重”，因此等效于通过加重尾部概率分布缓解了 crowding problem。
- 最终训练的 loss function:

$$C = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- 优化方法：求  $\frac{\partial C}{\partial y_i}$  更新  $y_i$  在低维空间中的位置。



# Table of Contents

## 10 Robust Machine Learning

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Robust Machine Learning

## 10 Robust Machine Learning

思想：很多机器学习算法只要对输入数据进行微小的扰动，输出结果就完全错误了。希望能有一种方法能够让机器学习算法对输入数据的扰动具有一定的鲁棒性。

几种方法：

- FGSM: 当学习率趋于无穷时，先更新再投影到正方体内结果大概率是顶点上。
- PGD: run gradient descent, and then project it back.

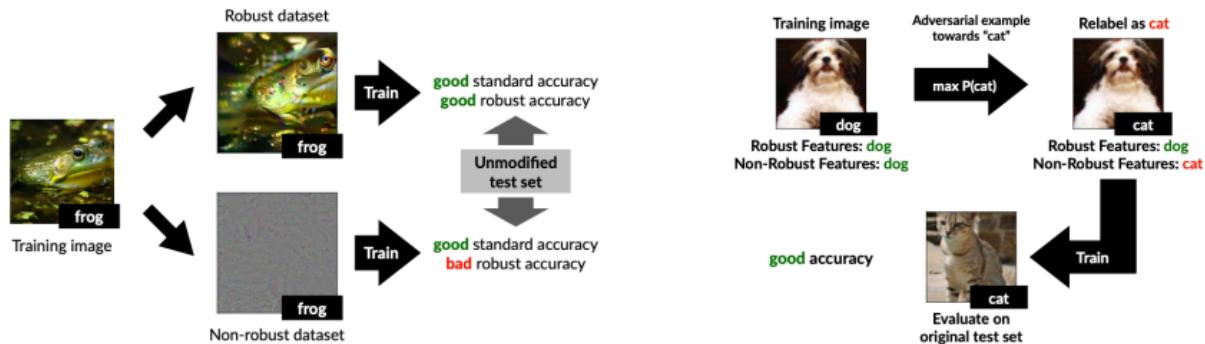
详见 Yang Yuan slides。



# Robust Features are Important

## 10 Robust Machine Learning

- 左图实验发现，从原始既包含 Robust Features 也包含 Non-Robust Features 的数据集上只提取 Robust Features 部分，并在获得的新数据集 Robust dataset 上进行训练，可以得到类似的 standard accuracy 和 robust accuracy（与在原始训练集上训练的结果相比）。
- 这说明模型可以只从 Robust Features 中学习，而不必须学习 Non-Robust Features。
- Note: Robust Features 通常指人能看到的特征，因为人识别就是 Robust 的。

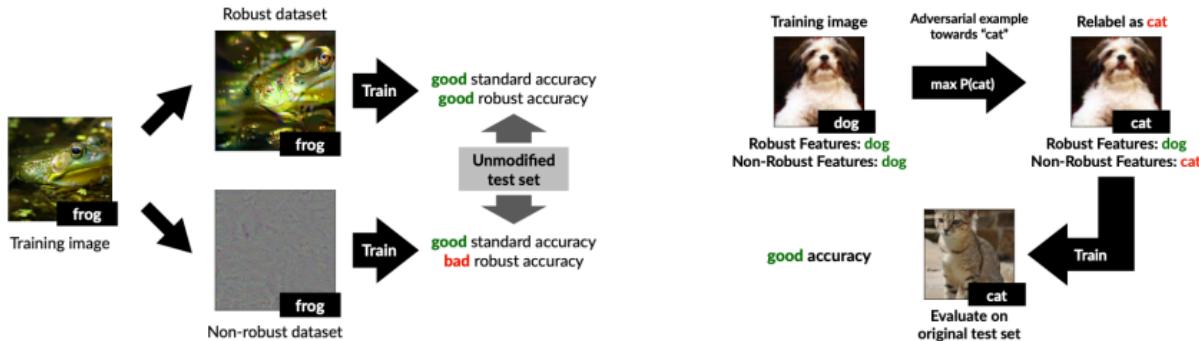




# Non-Robust Features are also Important

## 10 Robust Machine Learning

- 右图实验：通过调整 Non-Robust Features 让其向某错误的标签  $c_{\text{wrong}}$  偏移，然后构建一个新的数据集，这个数据集标签是按照  $c_{\text{wrong}}$  来标记的，图像是原图像加上 Non-Robust Features 的偏移之后的图像。
- 由于 Robust Feature 没有变化，人作为 Robust Learner，仍然能够正确识别这些图像。
- 但实验结果表明，模型用这个新数据集训练，在原始 test set 上测试 accuracy 差不多不变，这说明模型也可以只从 Non-Robust Features 中学习。





# Greedy Filling

## 10 Robust Machine Learning

- 问题：已知一个分类器  $f$ ，希望构建一个平滑分类器  $g$ ，使预测更加 robust。
- 自然想法：对单个输入  $x$ ，不只考虑  $f(x)$  的值，还考虑  $f(x')$  的值，其中  $x'$  与  $x$  在输入空间中非常接近。
- 具体的： $g(x)$  按照某概率分布函数  $p(x')$  (中心为  $x$ ) 加权，求不同标签  $c$  对应的  $x'$  占据的概率比例，概率比例最大的那个标签作为  $g(x)$  的预测结果。

$$g(x) = \arg \max_c \int_{x'} p(x') \mathbb{1}[f(x') = c] dx'$$

- Greedy Filling 算法用于计算对于给定输入点  $x$  和给定每个标签  $c$  对应的  $\int_{x'} p(x') \mathbb{1}[f(x') = c] dx'$  的值 (即给定每种标签的概率直方图)，在最坏情况下， $x$  偏移多少就有可能让  $g(x)$  的预测结果发生变化。
- 以下为简单，只考虑二分类问题 (只有两个 class  $c_1$  和  $c_2$ ) 的 Greedy Filling 算法。



# Greedy Filling Example 1: Unit Ball

## 10 Robust Machine Learning

- 将概率分布函数  $p$  取为单位球  $B = \{x' \in \mathbb{R}^d | \|x' - x\|_2 \leq R\}$  内的均匀分布

$$p(x') = \begin{cases} \frac{1}{\text{Vol}(B)} & x' \in B \\ 0 & \text{otherwise} \end{cases}$$

- 目标：求  $\delta$  使  $\|\delta\|_2$  最小，且  $g(x + \delta)$  能改变最大概率对应的 class (二分类问题中就是两个 class 概率各为 50%)。



# Greedy Filling Example 1: Unit Ball

## 10 Robust Machine Learning

- 目标：求  $\delta$  使  $\|\delta\|_2$  最小，且  $g(x + \delta)$  能改变最大概率对应的 class (二分类问题中就是两个 class 概率各为 50%)。
- 记  $S_1 = \{x' \in \mathbb{R}^d | \|x' - x\|_2 \leq R\}$ ,  $S_2 = \{x' \in \mathbb{R}^d | \|x' - x - \delta\|_2 \leq R\}$
- 不妨设  $B$  的体积为 1, 给定  $x$  点以及给定两个 class  $c_1$  和  $c_2$  在  $x$  点处概率占比为 80% 和 20%, 按照 Greedy 的核心思想, 为了让  $g(x + \delta)$  的预测结果发生尽量大的变化,
  - 若  $\text{vol}(S_1 - S_2) \geq 0.8$ , 应将  $S_2 - S_1$  内全部假设为  $c_2$  类, 由于有两个 class  $c_1$  和  $c_2$  在  $x$  点处概率占比为 80% 和 20% 的约束, 所以将  $S_1 - S_2$  内 0.8 体积假设为  $c_1$  类, 其他  $S_1$  中体积假设为  $c_2$  类。
  - 若  $\text{vol}(S_1 - S_2) < 0.8$ , 应将  $S_1 - S_2$  内全部假设为  $c_1$  类,  $S_2 - S_1$  内全部为  $c_2$  类,  $S_1 \hat{\cap} S_2$  内体积为  $0.8 - \text{vol}(S_1 - S_2)$  填成  $c_1$ , 其他填成  $c_2$ 。

据此可以求出  $\|\delta\|_2$  的最小值。



## Greedy Filling Example 2: Gaussian Distribution

### 10 Robust Machine Learning

- 将概率分布函数  $p$  取为 Gaussian Distribution

$$p(x') = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{1}{2\sigma^2} \|x' - x\|_2^2\right)$$

(作业中取  $\sigma = 1$ )

- 目标：求  $\delta$  使  $\|\delta\|_2$  最小，且  $g(x + \delta)$  能改变最大概率对应的 class。不妨假设  $g(x) = c_1$ 。
- 按照 Greedy 的核心思想，为了让  $g(x + \delta)$  的预测结果发生尽量大的变化，应该先从  $p(x' - x - \delta)/p(x' - x)$  最大的  $x'$  处开始设成  $c_2$ ，因为先从这里开始选取  $c_2$  可以让  $g(x + \delta)$  中  $c_2$  的成分相较  $g(x)$  中  $c_2$  的成分增加最多。
- 再由高斯分布的特性， $p(x' - x - \delta)/p(x' - x)$  的等值线是超平面，退化为一维问题， $\|\delta\|_2$  的最小值容易求解。（剩下的步骤参考习题课）



# Table of Contents

## 11 Hyperparameter Optimization

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Hyperparameter Optimization

## 11 Hyperparameter Optimization

思想：超参数的选择多种多样，如何高效地找到最优超参数？

几种算法

- Bayesian Optimization
- Gradient Optimization
- Random Search
- Multi-armed Bandit: Successive Halving (SH) Algorithm
- Neural Architecture Search (NAS): ProxylessNAS



# Bayesian Optimization & Gradient Optimization & Random Search

## 11 Hyperparameter Optimization

- 人工智能回答的非常好
- 看看课件基本上了解大致原理就行



# Multi-armed Bandit: Successive Halving (SH) Algorithm

## 11 Hyperparameter Optimization

算法：

### Algorithm 1 Successive Halving

**Input:** budget  $B$

1:  $S_0 \leftarrow [n]$

2: Per round budget  $B' \leftarrow \frac{B}{\log_2(n)}$

3: **for**  $r = 0$  to  $\log_2(n) - 1$  **do**

4:     Sample each arm  $i \in S_r$  for  $\frac{B'}{|S_r|}$  times

5:     Let  $S_{r+1}$  be the set of  $|S_r|/2$  arms in  $S_r$  with the largest empirical average

6: **end for**

**Output:**  $S_{\log_2(n)}$



### Theorem

不妨假设  $v_1 \geq v_2 \geq \dots \geq v_n$ , 记  $\Delta_i = v_1 - v_i$ 。

在取摇老虎机次数  $B = O\left(H_2 \log n \log \frac{\log n}{\delta}\right)$  时, 上述算法以至少  $1 - \delta$  的概率找到了最优的老虎机, 其中  $H_2 = \max_{i>1} \frac{i}{\Delta_i^2}$ 。

证明思路:

- 每一个 iteration 中最优的老虎机被淘汰的概率被 bound 住
- 通过 union bound 证明最后最优的老虎机被淘汰的概率也被 bound 住



# SH Algorithm in Hyperparameter Optimization

## 11 Hyperparameter Optimization

算法：

---

**Algorithm 2** Successive Halving

---

**Require:** budget  $B$

1:  $S_0 \leftarrow [n]$

2: Per round budget  $B' \leftarrow \frac{B}{\log_2(n)}$

3: **for**  $r = 0$  to  $\log_2(n) - 1$  **do**

4:     Pull each arm  $i \in S_r$  for  $\frac{B'}{|S_r|}$  times, get the current value  $\ell_{i,k_i}$ .

5:     Let  $S_{r+1}$  be the set of  $|S_r|/2$  arms in  $S_r$  with the smallest  $\ell_{i,k_i}$

6: **end for**

**Ensure:**  $S_{\log_2(n)}$

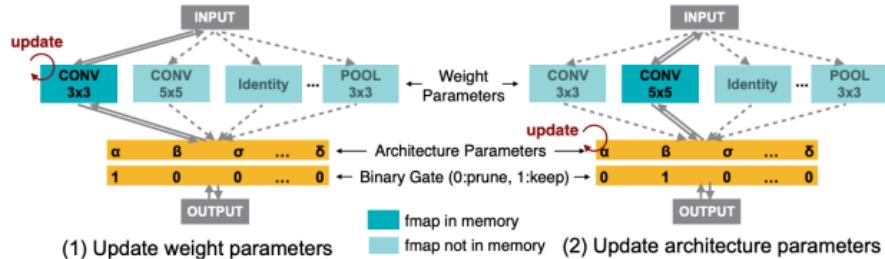
---

- 由于 Hyperparameter Optimization 问题与原始老虎机问题不同：老虎机中是每次玩老虎机，每次可以获得一个独立的输出；但 Hyperparameter Optimization 中每次训练模型，是时间累计的，故要进行一定的修改。
- 这也有一个理论保证，详见课件。



# Neural Architecture Search: ProxylessNAS

## 11 Hyperparameter Optimization



- 先构建由很多组成部分和连接方式的参数数量极大的模型，其中有一些特殊的层，对应的参数称为 **architecture parameter**，其他正常的模型参数称为 **weight parameter**。
- 训练方法：交替训练。先固定 **architecture parameter**，训练 **weight parameter**（训练方式与通常模型一样）；再固定 **weight parameter**，训练 **architecture parameter**，学习最好的 **architecture**。
- architecture parameter** 更新时由于有采样部分，无法梯度更新，只能近似：  
$$\frac{\partial L}{\partial \alpha_i} = \sum_{j=1}^N \frac{\partial L}{\partial p_j} \frac{\partial p_j}{\partial \alpha_i} \approx \sum_{j=1}^N \frac{\partial L}{\partial g_j} \frac{\partial g_j}{\partial \alpha_i}$$



# Table of Contents

## 12 Interpretability

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ **Interpretability**
- ▶ Rectified Flow
- ▶ RoPE



# Interpretability

## 12 Interpretability

思想：如果给模型一个输入，模型直接输出一个结果，模型对用户来说是一个黑箱，这对一些场景下是不合适的。希望模型能够输出一些直观解释，让用户能理解模型的决策过程。

几种方法：

- LIME Algorithm
- Integrated Method
- SHAP



一般框架：

- 设希望被解释的模型为  $f: \mathbb{R}^d \rightarrow \mathbb{R}$ , 输入为  $x$ 。目标：解释为什么  $f(x)$  这样输出。
- 设  $g \in G$  是一个解释模型，定义域为  $\{0, 1\}^{d'}$  代表  $x$  可解释的表示的二进制向量。
- $\Omega(g)$  代表解释的复杂度度量。
- $\Pi_x(z)$  表示 instance  $z$  与  $x$  的接近程度。
- 目标：最小化  $\xi(x) = \arg \min_{g \in G} (L(f, g, \Pi_x) + \Omega(g))$ ,  $\xi(x)$  就是最优解释。

我们只考虑  $G$  是线性模型族的情况，即  $g(z') = w_g \cdot z'$ 。

通常取  $L(f, g, \Pi_x) = \sum_{z, z' \in Z} \Pi_x(z) \cdot (f(z) - g(z'))^2$ , 其中  $\Pi_x(z) = \exp(-D(x, z)^2 / \sigma^2)$ ,  
 $\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > k]$



# LIME Algorithm

## 12 Interpretability

算法：

---

### Algorithm 1 LIME for Sparse Linear Explanations

---

**Require:** Classifier  $f$ , Number of samples  $N$

**Require:** Instance  $x$ , and its interpretable version  $x'$

**Require:** Similarity kernel  $\Pi_x$ , Length of explanation  $K$

$\mathcal{Z} \leftarrow \{\}$

**for**  $i \in \{1, 2, 3, \dots, N\}$  **do**

$z'_i \leftarrow sample\_around(x')$

$\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \Pi_x(z_i) \rangle$

**end for**

$w \leftarrow K\text{-Lasso}(\mathcal{Z}, K)$   $\triangleright$  with  $z'_i$  as features,  $f(z)$  as target **return**  $w$

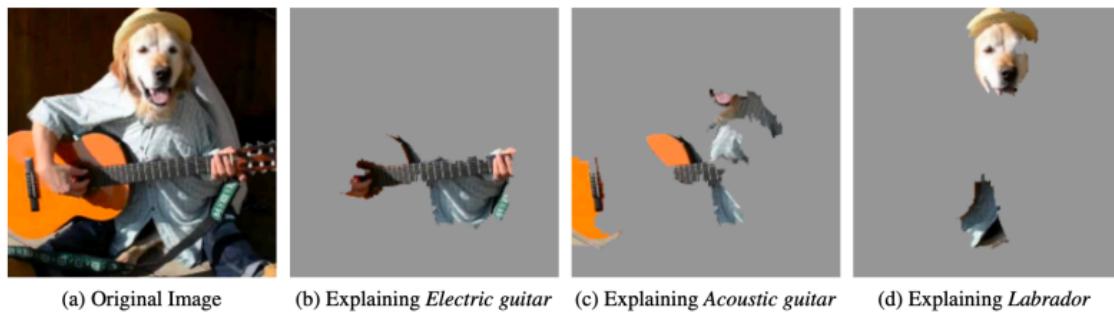
---



# LIME Algorithm

## 12 Interpretability

- 在图像分类模型解释问题中，每个可解释的  $z$  是对每个 super-pixel 分配 0 或 1。
- 为了解释为什么模型将一个弹着吉他的狗头人识别出来是 Labrador、Electric guitar 和 Acoustic guitar，按照上述算法中  $K - Lasso$  的原理，最终每种标签对应的返回权重  $w$  是比较稀疏的，将这些非 0 的  $w$  分量对应的 super-pixel 按原图输出出来，0 的  $w$  分量对应的 super-pixel 输出灰色图像，可得到下图。



**Figure 4: Explaining an image classification prediction made by Google's Inception network, highlighting positive pixels. The top 3 classes predicted are “Electric Guitar” ( $p = 0.32$ ), “Acoustic guitar” ( $p = 0.24$ ) and “Labrador” ( $p = 0.21$ )**



# Integrated Method

## 12 Interpretability

思想：Integrated Method 论文中指出了一些关于可解释性的公理，然后从这些公理出发推导可能的解释方法。Integrated Gradients 满足所有他们提出的公理。

- 设函数  $F: \mathbb{R}^n \rightarrow [0, 1]$  表示一个 deep network,  $x \in \mathbb{R}^n$  是输入,  $x' \in \mathbb{R}^n$  是一个 baseline input。
- 定义

$$\text{IntegratedGrads}_i(x) := (x_i - x'_i) \cdot \int_{\alpha=0}^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha$$

- Completeness:

$$\sum_{i=1}^n \text{IntegratedGrads}_i(x) = F(x) - F(x')$$

证明方法：简单的微积分。



# Shapley Value Algorithm (a.k.a. SHAP)

## 12 Interpretability

- 问题:  $x_1, x_2, \dots, x_n$  是输入变量,  $S \subseteq \{x_1, x_2, \dots, x_n\}$ ,  $f(S)$  表示  $S$  对最终结果的贡献。如何计算每个输入变量单独的作用?
- 生动的例子: 抢银行分钱, 总钱数为  $f([n])$ .
- $x_i$  的作用是 ( $x_i$  应该分得的钱)

$$\phi_i(f) = \sum_{S \subseteq [n] \setminus \{i\}} \frac{|S|! (n - |S| - 1)!}{n!} (f(S \cup \{i\}) - f(S))$$

- 性质:

$$\sum_{i=1}^n \phi_i(f) = f([n])$$



# Table of Contents

## 13 Rectified Flow

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ **Rectified Flow**
- ▶ RoPE



# Rectified Flow

## 13 Rectified Flow

算法：

---

**Algorithm 1** Rectified Flow: Main Algorithm

---

**Procedure:**  $Z = \text{RectFlow}((X_0, X_1))$ :

*Inputs:* Draws from a coupling  $(X_0, X_1)$  of  $\pi_0$  and  $\pi_1$ ; velocity model  $v_\theta: \mathbb{R}^d \rightarrow \mathbb{R}^d$  with parameter  $\theta$ .

*Training:*  $\hat{\theta} = \arg \min_{\theta} \mathbb{E} \left[ \|X_1 - X_0 - v(tX_1 + (1-t)X_0, t)\|^2 \right]$ , with  $t \sim \text{Uniform}([0, 1])$ .

*Sampling:* Draw  $(Z_0, Z_1)$  following  $dZ_t = v_{\hat{\theta}}(Z_t, t)dt$  starting from  $Z_0 \sim \pi_0$  (or backwardly  $Z_1 \sim \pi_1$ ).

*Return:*  $Z = \{Z_t: t \in [0, 1]\}$ .

**Reflow** (optional):  $Z^{k+1} = \text{RectFlow}((Z_0^k, Z_1^k))$ , starting from  $(Z_0^0, Z_1^0) = (X_0, X_1)$ .

**Distill** (optional): Learn a neural network  $\hat{T}$  to distill the  $k$ -rectified flow, such that  $Z_1^k \approx \hat{T}(Z_0^k)$ .

---



# Table of Contents

14 RoPE

- ▶ SVM
- ▶ Decision Trees
- ▶ Boosting
- ▶ PCA
- ▶ Nearest Neighbor and LSH
- ▶ Metric Learning
- ▶ Clustering
- ▶ SimCLR
- ▶ t-SNE
- ▶ Robust Machine Learning
- ▶ Hyperparameter Optimization
- ▶ Interpretability
- ▶ Rectified Flow
- ▶ RoPE



# Transformer

## 14 RoPE

- 记  $S_N = \{w_i\}_{i=1}^N$  是  $N$  个输入词汇序列
- $E_N = \{x_i\}_{i=1}^N$  是对应的 word embedding,  $x_i$  中没有位置信息
- 融入位置信息:

$$\begin{cases} q_m &= f_q(x_m, m) \\ k_n &= f_k(x_n, n) \\ v_n &= f_v(x_n, n) \end{cases}$$

- attention weight:  $a_{mn} = \frac{\exp(q_m^T k_n / \sqrt{d})}{\sum_{j=1}^N \exp(q_m^T k_j / \sqrt{d})}$
- output embedding:  $o_m = \sum_{n=1}^N a_{mn} v_n$



# Rotary Position Embedding (a.k.a. RoPE)

14 RoPE

- 在 Transformer 模型中，RoPE 是一种将 token 位置融入 attention 模块的方法。
- 核心思想：希望计算 attention weight 的时候  $q_m^T k_n$  点积结果只与  $x_m, x_n$  和  $m - n$  相关。
- 2D 情况下，一个解为复数解，可以等价地写成矩阵形式方便计算。

$$f_{\{q,k\}}(\mathbf{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_m^{(1)} \\ \mathbf{x}_m^{(2)} \end{pmatrix}$$

- 更高维情况下可以写成

$$f_{\{q,k\}}(\mathbf{x}_m, m) = \mathbf{R}_{\Theta, m}^d \mathbf{W}_{\{q,k\}} \mathbf{x}_m$$

$$\mathbf{R}_{\Theta, m}^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

其中  $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, \dots, d/2]\}$



# Machine Learning

*Thank you for listening!  
Any questions?*