Introduction to Machine Learning

Statistics (less focus on algorithm, more model design & hypothesis testing)

Machine Learning (less focus on hypothesis testing, more on algorithms)

machine learning

1. **supervised learning** :

   Data: $\{x_j, y_j\}_{j=1}^n$

   Model: $y_j = f^*(x_j) + \epsilon_j$ where y_j is label, ϵ_j is noise.

   Objective: approximate f^*

2. **unsupervised learning** :

   Data: $\{x_j\}_{j=1}^n$

   Model: $y_j = f^*(x_j) + \epsilon_j$ where y_i is label,

   Objective: 找出规律

3. **reinforcement learning**:

   policy function: from state space to action space 从状态空间到行为空间

   Objective: optimal decision(寻找最优决策)

Supervised Learning

1. Regression 回归

$f^*: \ D \subset R^d \to R$ f^*连续取值

2. Classification 分类

$f^* : D \to G(finite\ set), G = \{-1, 1\}$

3. Framework

(1). **Hypothesis Space** $\mathcal{H}_m$

e.g. $\mathcal{H}_m = \{w_0 + w^\top x, w_0 \in R, w \in R^d\}$

e.g. $\mathcal{H}_m = \{\sum_{j=1}^m \alpha_j \phi_j(x)\}$ where $\{\phi_j(x)\}_{j=1}^m$ is fixed set of function. For example, we can set $\phi_j(x) = cos(k_j x)$

e.g. $\mathcal{H}_m = \{\sum_{j=1}^m a_j \sigma(b_j^\top x + c_j)\}$ For example, we can set $\sigma(x) = \max\{0, x\}$(ReLU, an activation function for Neural Network)

In examples, m represents (scale of) degree of freedom*(You can search **VC dimension** if you want know more about it)*

(2). **Objective Function** (Loss Function): *loss function here is an example of square loss*

$$\hat{R}_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}(\hat{y}_j - f(x_j, \theta))^2 + \lambda||\theta||$$

Where θ is parameter(参数), $||\theta||$ is norm(范数), $\lambda||\theta||$ is regularization term*(惩罚项，用于控制模型复杂度)*.

(3). **Optimization Method** 优化算法

- 梯度法*(e.g. Gradient Decent(最简单的梯度下降，只用一阶导数))*
- 随机梯度法*(e.g. SGD(随机梯度下降， 只用一阶导数), Adam(自适应梯度下降， 只用一阶导数) ...)*
- BFGS等 *（一种二阶优化算法， 用二阶导数）。。。*

Examples for Supervised Learning

1. Linear Model

$$\mathcal{H}_m = \{w_0 + w^\top x, w_0 \in R, w \in R^d\}$$

$$\begin{gathered}
\text{write } w_0 + w^\top x \text{ as } \ w^\top x \\
\hat{R}_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}(w^\top x_j - y_j)^2 \\
\nabla_\theta \hat{R}_n(\theta) = \sum(w^\top x_j - y_j)x_j = 0 \\
X = (x_1, \dots, x_n) \\
\hat{w} = (XX^\top)^{-1}Xy
\end{gathered}$$

Regularization(**Ridge Regression**)

$$\begin{gathered}
\hat{R}_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}(w^\top x_j - y_j)^2 + \lambda||w||^2 \\
\hat{w} = (XX^\top + \lambda I)^{-1}Xy \\
\lim_{\lambda \to 0}(XX^\top + \lambda I)^{-1} = (XX^\top)^{-1}(Generalized\ inverse\ matrix)
\end{gathered}$$

Use another regularization term

$$\begin{gathered}
\hat{R}_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}(w^\top x_j - y_j)^2 + \lambda N(w)\ ;(N(w)\text{的作用是求最稀疏的解}) \\
N(w) = number\ of\ non - zero\ component\ of\ w
\end{gathered}$$

Using $||w||_1$, Model become **Lasso Regression**

$$\hat{R}_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}(w^\top x_j - y_j)^2 + \lambda||w||_1$$

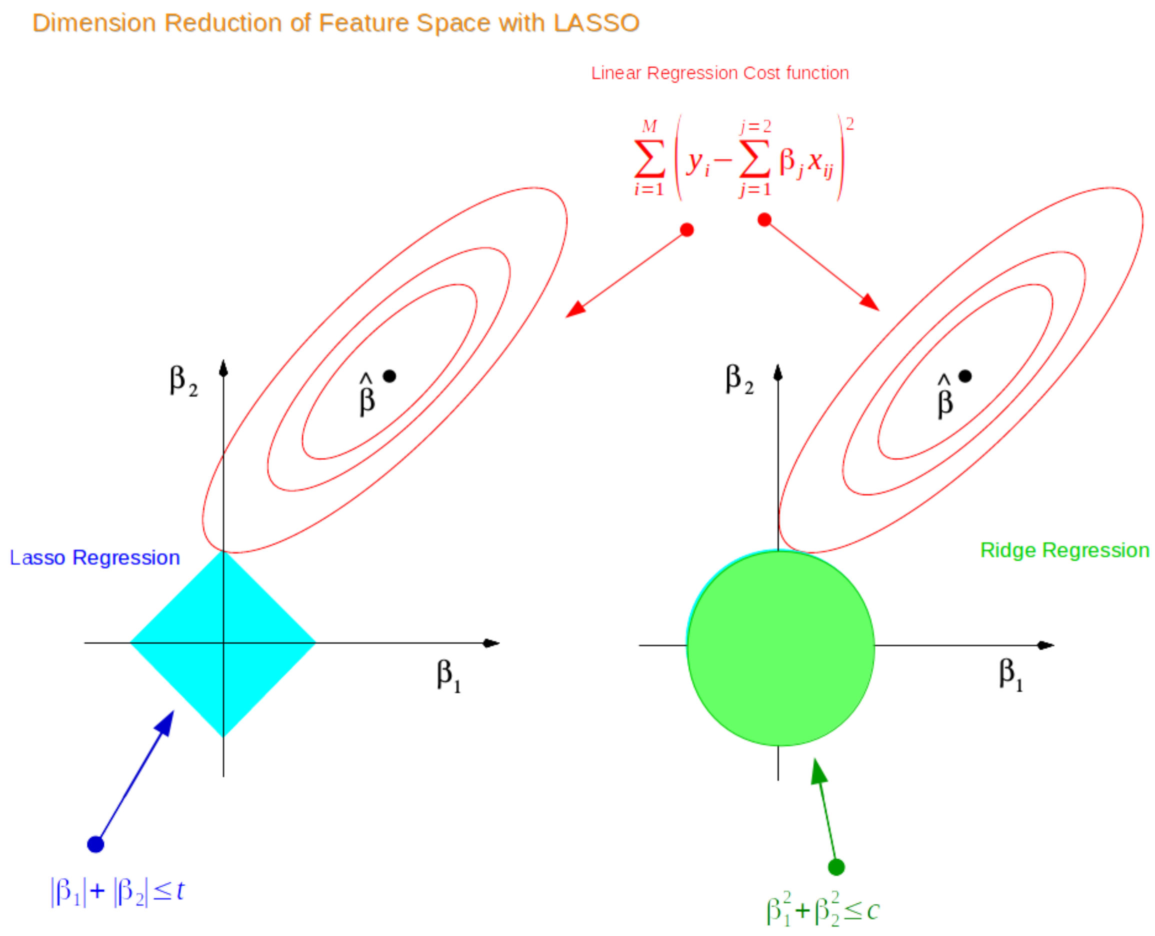

Figure source:

https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

2. Kernel Method(核方法)

kernel function $k(x,y), x,y \in R^d$

e.g.

- $k(x,y) = e^{-\frac{\|x-y\|^2}{2}}, k(x,y) = \phi(\|x-y\|^2) = \phi(r)$

Definition:

(1) k is symmetric

$$k(x,y) = k(y,x)$$

(2) $\forall \{x_j\}_{j=1}^n$

$$K = (k(x_i,x_j))_{n\times n} \geq 0$$

K is **SPD**(symmetric positive definite)

Define **Kernel Space**:

$$\mathcal{H}_m = \{\sum_{j=1}^{n} \alpha_j k(x_j, x)\}\ (m = n)$$

Feature-based method (feature，特征) $\{\phi_j(x)\}_{j=1}^{m}$ 是一组特征

$$\mathcal{H}_m = \{\sum_{i=1}^{m} \alpha_j \phi_j(x)\}$$

3. Neural Network(神经网络, NN)

$$\mathcal{H}_m = \{\sum_{j=1}^{m} a_j \sigma(b_j^\top x)\}$$

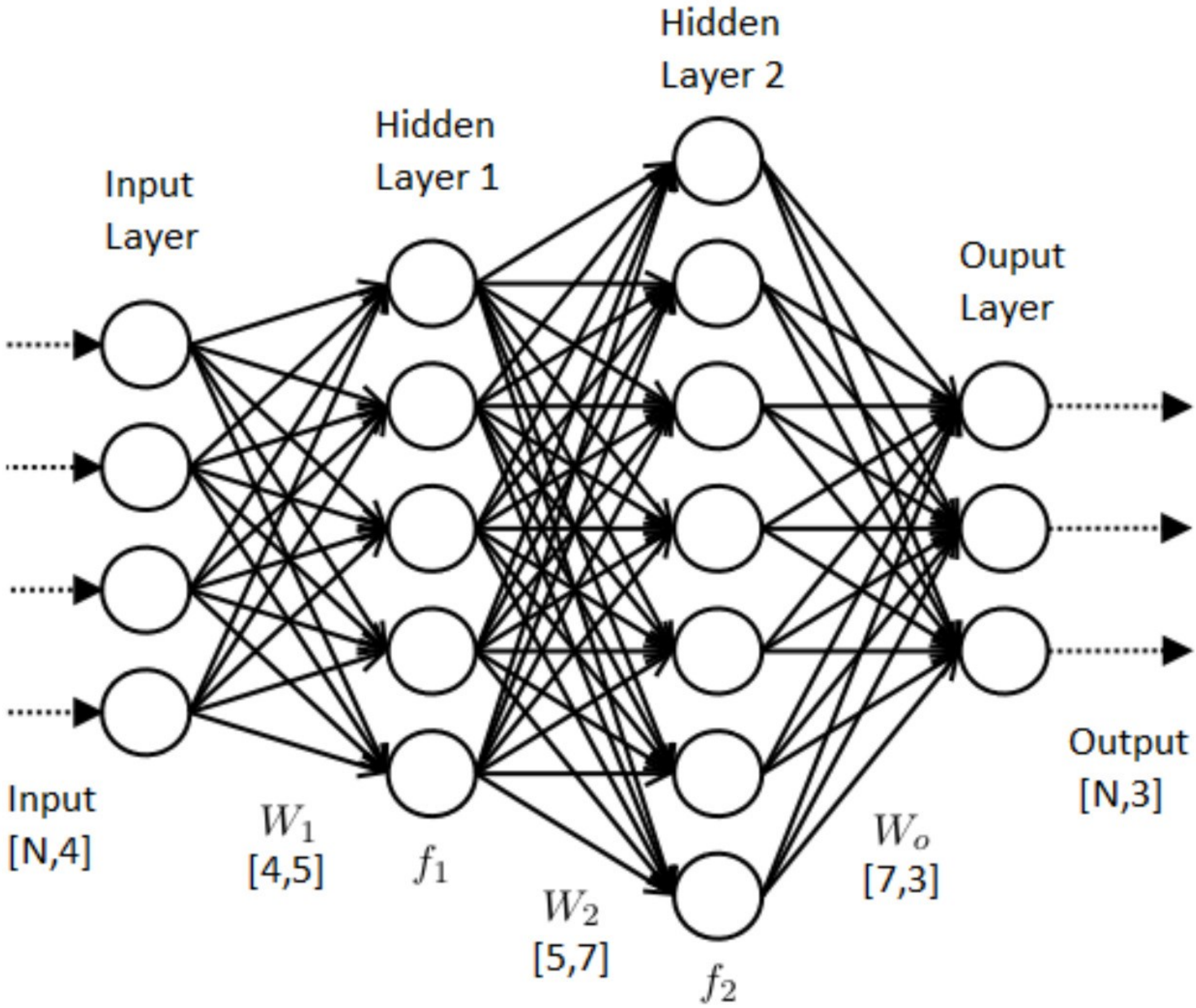

Figure source:

https://www.datasciencecentral.com/profiles/blogs/the-artificial-neural-networks-handbook-part-1?xg_source=activity

Why NN better? (No complete theory in mathematics)

Compared with generalized linear model(GLM)

$$\mathcal{H}_m = \{\sum_{i=1}^{m} \alpha_j \phi_j(x)\}$$

We have a "**Theorem**", For GLM

$$f : R^d \to R,\ d >> 1$$
$$\inf_{f_m \in \mathcal{H}_m} \|f_m - f\| \geq cm^{-\frac{1}{d}}$$

For NN

$$f : R^d \to R,\ d >> 1$$
$$\inf_{f_m \in \mathcal{H}_m} \|f_m - f\| \leq cm^{-\frac{1}{2}}$$

So let $error = 0.1$. For NN

$$m \sim 10^2$$

For GLM

$$m \sim 10^d \text{ (维数灾难)}$$

4. Optimization Algorithm (优化算法)

Gradient Descent:

$$\min_\theta F(\theta)$$
$$\theta_{k+1} = \theta_k - \eta_k \nabla F(\theta_k)$$

For example

$$\nabla F(\theta) = \frac{1}{2} \sum_{j=1}^{n} \nabla (f(\theta, x_j) - y_j)^2 = \sum (f(\theta, x_j) - y_j) \nabla_\theta f$$

Back Propagation(反向传播，BP) : Just use **Chain Rule**(链式法则)

In practice, we use Stochastic Gradient Descent (SGD) Methods(随机梯度方法) because of the large scale of data.

5. Classification(分类)

$$y = \{-1, 1\}$$

$$y = H(f(x))$$
$$H(z) = \begin{cases} 1, z > 0 \\ 0, z \le 0 \end{cases}$$

For continuity, we can let $H(z) = \frac{1}{1+e^{-z}}$(sigmoid)

Logistic Regression

set $f = w^\top x$ and $y = \frac{1}{1+e^{-w^\top x}}$

Above are binary classifications. For **multi-class classification** (K classes), we use softmax

$$q_j(x) = \frac{e^{f_j(x)}}{\sum_{k=1}^K e^{f_k(x)}}$$
$$\sum_{j=1}^K q_j(x) = 1$$

Where $q_j(x)$ can be regard as the probability of $x \in class_j$