Linear prediction : features, overfitting, and losses

Feature Engineering :

$\mathbb{R}^d \to \mathbb{R}^p$ replace with x with $\phi(x)$

Eg

① non-linear transformation $x \in \mathbb{R}$

semi-often $\phi(x) = \ln(1+x)$

② logical formula, for $x = (x_1 \dots x_d) \in \{0,1\}^d$

never seen it $\phi(x) = (x_1 \wedge x_3 \wedge \neg x_{10}) \vee (\neg x_2 \wedge x_7)$

③ Trigonometric expansion $x \in \mathbb{R}$

rare $\phi(x) = (1, \sin(x), \cos(x), \sin(2x), \cos(2x) \dots)$

④ polynomial expansion : for $x = (x_1 \dots x_d) \in \mathbb{R}^d$

frequent $\phi(x) = (1, x_1 \dots x_d, x_1^2 \dots x_d^2, x_1 x_2 \dots x_1 x_d \dots x_{d-1} x_d)$

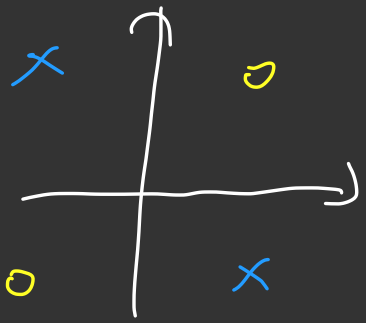
Monomial features

$\phi(x) = (1, x, x^2)$ $\quad w = (a, b, c)$

$x \to w^T \phi(x) = a + bx + cx^2$

$$w^T \phi(x) = \begin{bmatrix} a \\ b \\ c \end{bmatrix}^T \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix} = a + bx + cx^2$$

Eg. XOR : $x \in (\pm 1, \pm 1)$ $y = x_1 x_2$



feature : $\phi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

$$w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Feature is problem-specific

Feature expansion:

Reduce training error

$$\phi(x) = \begin{cases} e_i & x_i = x \\ 0 & \text{o.w.} \end{cases}$$

$y_i \in \{-1, +1\}$ $\quad w = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \begin{bmatrix} | \\ \phi(x) \\ | \end{bmatrix}^T = \begin{cases} y_i & x = x_i \\ 0 \end{cases}$

But feature expansion can do bad on future data.

Eg. (x, y) $x \sim U(0,1)$ $y \sim N(0,1)$

$x \sim w^T(1, x, x^2, \cdots, x^r)$

Method: OLS solution $\hat{w}_{OLS}$

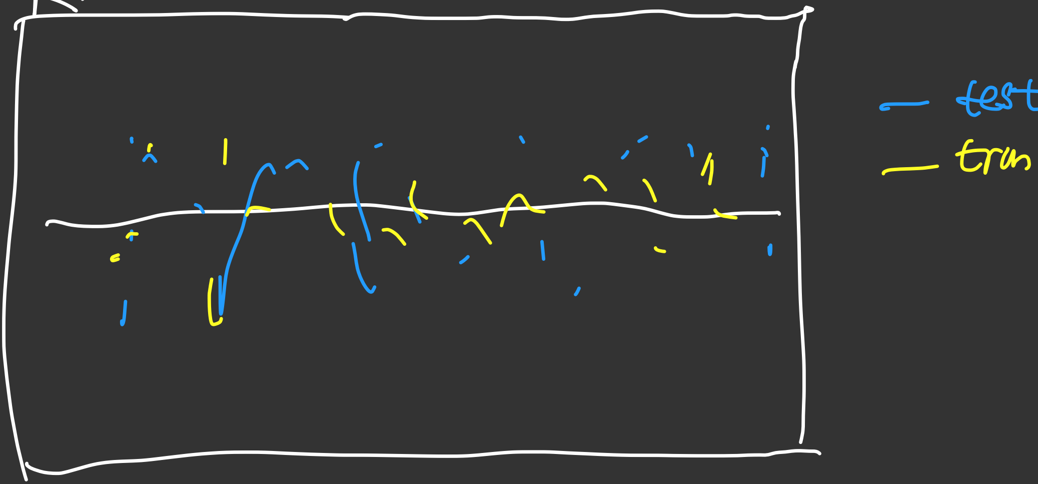
$$\left\| \begin{bmatrix} - \phi(x_1)^T - \\ \vdots \\ - \phi(x_n)^T - \end{bmatrix} \begin{bmatrix} | \\ w \\ | \end{bmatrix} - \begin{bmatrix} | \\ y \\ | \end{bmatrix} \right\|^2 = 0$$

$r \to r+1 = n$ $\qquad n \le r+1$

eg # of points

$\Rightarrow$ Maybe we can make $r \to \infty$ so this model can fit to inf points?

$\hat{R}_1 = 0.0069204$ $\hat{R}_1 = 0.0089745$ $R_2 = 0.006937$ $\hat{R}_2 = 0.0057077$



Degree 1: training error 0.0069, test error 0.00897

Degree 16: training error 0.0031, test error 309126

Overfitting: when training error good, but bad on testing error.

① often it means a model is complicated in a way that is comparable with the data

② a bias / variance tradeoff
train error — random test error

③ not mean "complex models are bad", DNN can perform good

Resolution for overfitting

① Reduce model complexity via regularized ERM: Pick $\lambda > 0$ and

$$\min_{w \in \mathbb{R}^d} \hat{R}(w) + \frac{\lambda}{2}\|w\|_2^2$$

regularization (penalty) (weight decay)

(not very large optimization)

$$w = (X^T X + \lambda n I)^{-1} X^T y$$

λ: hyperparameter

$$\nabla_w \left(\hat{R}(w) + \frac{\lambda}{2}\|w\|^2\right) = \nabla \hat{R}(w) + \lambda w$$

Loss functions and multi-class prediction

Standard Classification losses

① hinge : $\ell_{\text{hinge}}(z) = \max\{0, 1-z\}$

② squared: $\ell_{(y)}(z) = (1-z)^2$

③ logistic: $\ell_{\text{logistic}}(z)/\ln(2) = (\ln(1+e^{-z}))/\ln(2)$

④ exponential: $\ell_{\exp}(z) = e^{-z}$

Design losses with MLE

$$\arg\max_w \prod_{i=1}^n P_w(y_i|x_i) = \arg\min_w -\ln \prod_{i=1}^n P_w(y_i|x_i) = \arg\min_w \ln \sum_{i=1}^n \frac{1}{P_w(y_i|x_i)}$$

Squared loss $P_w(y|x)$ = standard Gaussian with mean $x^T w$

$$\ln P_w(y|x) = \ln \frac{\exp\left(-\frac{1}{2\sigma^2}(y - w^T x)^2\right)}{\sigma\sqrt{2\pi}}$$

$$= \frac{-1}{2\sigma^2}(y - w^T x)^2 - \frac{n}{2}\ln(2\pi\sigma^2)$$

result same

$$\Rightarrow \arg\max_w P_w(y|x) = \arg\min_w \pm(y - w^T x)$$

Logistic loss

$$P_w(y=1|x) = \frac{1}{1+\exp(-x^T w)}$$ ← sigmoid distribution

$$P_w(y=-1|x) = 1 - P_w(y=1|x) = \frac{\exp(-x^T w)}{1+\exp(-x^T w)} = \frac{1}{1+\exp(x^T w)}$$

Then

$$\ln \frac{1}{P_w(y|x)} = \ln \frac{1}{P_w(y=1|x)^{\frac{1+y}{2}} (1-P_w(y=1|x))^{\frac{1-y}{2}}}$$

$$= \frac{1+y}{2} \ln(1+\exp(-x^T w)) + \frac{1-y}{2} \ln(1+\exp(x^T w))$$

$$= \ln(1+\exp(-yx^T w))$$

$$\operatorname{argmin}_w \ln \prod_{i=1}^{n} \frac{1}{P_w(y_i|x_i)} = \operatorname{argmin}_w \frac{1}{n} \sum_{i=1}^{n} \text{logistic}(y_i \cdot x_i^T w)$$