Regression

Linear Regression

Error: $\hat{R}(w) = \sum_{i=1}^{n}(y_i - w^T x_i)^2 = \|Xw - y\|_2^2$

Closed form: $w^* = (X^T X)^{-1} X^T y$

Gradient: $\nabla_w \hat{R}(w) = -2\sum_{i=1}^{n}(y_i - w^T x_i) \cdot x_i = 2X^T(Xw - y)$

Convex

g(x) is convex

$\Leftrightarrow x_1, x_2 \in \mathbb{R}, \lambda \in [0,1]:$

$g(\lambda x_1) + (1 - \lambda x_2) \leq \lambda g(x_1) + (1 - \lambda) g(x_2) \Leftrightarrow g''(x) > 0$

Jensen's inequality

X is a random variable & φ a convex function then the following holds $\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$

Gradient Descent

1. Start arbitrary $w_o \in \mathbb{R}$
2. For i do $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$

Expected Error

For generalization, minimize the expected error $R(w) = \int P(x,y)(y - w^T x)^2 \partial x \partial y$

$= \mathbb{E}_{x,y}[(y - w^T x)^2]$

Gaussian distribution

Standard deviation σ

Mean μ

$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$

Multivariate Gaussian

Covariance matrix Σ

Mean μ

$f(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$

Ridge regression

Error: $\hat{R}(w) = \sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|_2^2$

Closed form: $w^* = (X^T X + \lambda I)^{-1} X^T y$

Gradient: $\nabla_w \hat{R}(w) = -2\sum_{i=1}^{n}(y_i - w^T x_i) \cdot x_i + 2\lambda w$

Regularization

The error term L and the regularization C with regularization parameter λ: $\min_w L(w) + \lambda C(w)$

L1-regularization for number of features

L2-regularization for the length of w

Classification

0/1 loss

0/1 loss is not convex and not differentiable.

$l_{0/1}(w, y_i, x_i) = \begin{cases} 0 \text{ , if } y_i = sign(w^T x_i) \\ 1 \text{ , otherwise} \end{cases}$

Perceptron loss

Perceptron loss is convex and not differentiable, but gradient is informative.

$l_P(w, y_i, x_i) = max\{0, -y_i w^T x_i\}$

Stochastic Gradient Descent

1. Start arbitrary $w_0 \in \mathbb{R}^d$
2. For t do:

Pick data point $(x_1, y_1) \in_{u.a.r.} D$

$w_{t+1} = w_t - \eta_t \nabla l(w_t, x_1, y_1)$

Perceptron Algorithm

Stochastic Gradient + Perceptron loss

Theorem: If D is linearly seperable $\Rightarrow$ Perceptron will obtain a linear seperator.

Support Vector Machine

Try to maximize a "band" around the seperator.

Error: $\hat{R}(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \lambda\|w\|_2^2$

$= \max\{0, 1 - y^T X w\} + \lambda\|w\|_2^2$

Gradient: $\nabla_w \hat{R}(w) = \begin{cases} -X^T y + 2\lambda w \text{ , if } y_i w^T x_i < 1 \\ 2\lambda w, \text{ otherwise} \end{cases}$

Matrix-Vector Gradient

$\nabla_\beta(\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2) = 2X^T(y - X\beta) + 2\lambda\beta$

Hinge loss

loss for support vector machine.

$l_{SVM}(w, x_i, y_i) = \max\{0, 1 - y_i w^T x_i\} + \lambda\|w\|_2^2$

derivation:

$\frac{\partial}{\partial w_k} l_{SVM}(w, y_i, x_i) = \begin{cases} 0 \text{ ,} 1 - y_i w^T x_i < 0 \\ -y_i x_{i,k} \text{ , otherwise} \end{cases} + 2\lambda w_k$

Kernels

Reformulating the perceptron

Ansatz: $w = \sum_{j=1}^{n} \alpha_j y_j x_j$

$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \max[0, -y_i w^T x_i]$

$= \min_{\alpha_{1:n}} \sum_{i=1}^{n} \max[0, -y_i (\sum_{j=1}^{n} \alpha_j y_j x_j)^T x_i]$

$= \min_{\alpha_{1:n}} \sum_{i=1}^{n} \max[0, -\sum_{j=1}^{n} \alpha_j y_i y_j x_i^T x_j]$

Polynomial kernel

$k_1 = (x^T y)^m$ represents monomial of deg m

$k_2 = (1 + x^T y)^m$ represents monomials up to deg m

Kernelized Perceptron

1. Initialize $\alpha_1 = ... = \alpha_n = 0$
2. For t do

Pick data $(x_i, y_i) \in_{u.a.r} D$

Predict $\hat{y} = sign(\sum_{j=1}^{n} \alpha_j y_j k(x_j, x_i))$

If $\hat{y} \neq y_i$ set $\alpha_i = \alpha_i + \eta_t$

Properties of kernel

- k must be symmetric
- the kernel matrix must be SPD

Kernel matrix

The kernel matrix K is SPD

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix}$$

$\left(XX^T\right)$ for inner product as kernel.

semi-positive-definite matrices

$M \in \mathbb{R}^{n \times n}$ is SPD $\Leftrightarrow$

$\forall x \in \mathbb{R}^n : x^T M x \geq 0 \Leftrightarrow$

all eigenvalues of M are positive ≥ 0

Kernel engineering

$k_1(x,y) + k_2(x,y)$

$k_1(x,y) \cdot k_2(x,y)$

$c \cdot k_1(x,y)$ for $c > 0$

$f(k_1(x,y))$, where f is exponential/polynomial with positive coefficents

Parametric vs. Nonparametric

Parametric: have finite set of parameters

E.g. linear regression, perceptron,...

$f(x) = w^T x, w \in \mathbb{R}^d$ (d is independent of data)

Nonparametric: grows in complexity with the size of the data

E.g. kernelized Perceptron, k-NN,...

$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x_n)$ (depends on data)

Parametric to nonparametric linear regression

Ansatz: $w = \sum_i \alpha_i x$

Parametric: $w^* = \underset{w}{\text{argmin}} \sum_i (w^T x_i - y_i)^2 + \lambda\|w\|_2^2$

$= \underset{\alpha_{1:n}}{\text{argmin}} \sum_{i=1}^{n}(\sum_{j=1}^{n} \alpha_j x_j^T x_i - y_i)^2 + \lambda \sum_i \sum_j \alpha_i \alpha_j (x_i^T x_j)$

$= \underset{\alpha_{1:n}}{\text{argmin}} \sum_{i=1}^{n}(\alpha^T K_i - y_i)^2 + \lambda \alpha^T K \alpha$

$= \underset{\alpha}{\text{argmin}} \|\alpha^T K - y\|_2^2 + \lambda \alpha^T K \alpha$

Closed form: $\alpha^* = (K + \lambda I)^{-1} y$

Prediction: $y^* = w^{*T} x = \sum_{i=1}^{n} \alpha_i^* k(x_i, x)$

Feature Selection

Lasso regression

$w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|_1$

The alternative penalty encourages coefficients to be exactly 0 (automatic feature selection).

Sparse L1-SVM

$\underset{w}{\text{argmin}} \sum_{i=1}^{n} \max(0, 1 - y_i w^T x_i) + \lambda\|w\|_1$

Neural Networks

Learning features

Parameterize the feature maps and optimize over the parameters:

$w^* = \underset{w,\Theta}{\text{argmin}} \sum_{i=1}^{n} l(y_i, \sum_{j=1}^{m} w_j \Phi(x_i, \Theta_j))$

Backpropagation

For each unit j on the output layer:

- Compute error signal: $\delta_j = \ell_j'(f_j)$
- For each unit i on layer L: $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$
- For each unit j on hidden layer $l = \{L-1, .., 1\}$:
- Error signal: $\delta_j = \phi'(z_j) \sum_{i \in Layer_{l+1}} w_{i,j} \delta_i$
- For each unit i on layer $l-1$: $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$

Probability Modeling

Goal

Given data $D = \{(x_1, y_1), ..., (x_n, y_n)\} \subseteq X \times Y$

Want to find the hypothesis with the minimum prediction error (risk). $R(h) = \int P(x,y) l(y; h(x)) dx dy = \mathbb{E}_{x,y}[l(y; h(x))]$

Fundamental assumption: $(x_i, y_i) \in_{iid} X \times Y$

Hypothesis: is a conditional mean $H^*(x) = \mathbb{E}[y|X = x]$

Maximum Likelihood Estimation (MLE)

Choose a particular parametric form $P(Y|X, \theta)$, then optimize the parameters using Maximum Likelihood Estimation.

$\theta^* = \underset{\theta}{\text{argmax}} P(y|x, \theta)$

$= \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} P(y_i|x_i, \theta)$ (iid)

$= \underset{\theta}{\text{argmin}} - \sum_{i=1}^{n} log P(y_i|x_i, \theta)$

Example: MLE for lin. Gaussian

Assume: $P(Y = y|X = x, \theta) = \mathcal{N}(y; h(x), \sigma^2)$

and $h(x) = w^T x$ is linear

Equivalent: $Y = w^T X + \epsilon, \epsilon \in \mathcal{N}(0, \sigma^2)$

$y_i \in \mathcal{N}(w^T x_i, \sigma^2)$ Maximizing the log likelihood: $\underset{w}{\text{argmax}} P(y|x, \theta) =$

$\underset{w}{\text{argmax}} \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(y_i - h(x_i))^2}{\sigma^2}}$

log is monotonic and cancel all constants

$= \underset{w}{\text{argmin}} \sum_i (y_i - w^T x_i)^2$

Bias/Variance/Noise

Prediction error = $Bias^2 + Variance + Noise$

Maximum a posteriori estimate (MAP)

Introduce bias by expressing assumption through a Bayesian prior $w_i \in \mathcal{N}(0, \beta^2)$

Bayes rule: $P(w|x, y) = \frac{P(w|x)P(y|x,w)}{P(y|x)}$

$= \frac{P(w)P(y|x,w)}{P(y|x)}$, we assume w is independent of x.

Example MAP for lin. Gaussian

$\underset{w}{\text{argmax}} P(w|x, y) =$

$\underset{w}{\text{argmin}} - log P(w) - log P(y|x, w) + const.$

$= \underset{w}{\text{argmin}} \frac{1}{2\beta^2}\|w\|_2^2 + \frac{1}{2\sigma^2} \sum_{i=1}^{n}(y_i - w^T x_i)^2$

$= \underset{w}{\text{argmin}} \lambda\|w\|_2^2 + \sum_{i=1}^{n}(y_i - w^T x_i)^2$, w/ $\lambda = \frac{\sigma^2}{\beta^2}$

Logistic regression

Link function: $\sigma(w^T x) = \frac{1}{1+exp(-w^T x)}$

Derivate: $\frac{\partial}{\partial z}\sigma(z) = \sigma(z)(1-\sigma(z))$

Logistic regression replaces the assumption of Gaussian noise by iid Bernoulli noise.

$P(y|x,w) = Ber(y;\sigma(w^T x))$

$= \begin{cases} 1/(1+\exp(-w^T x)) = \sigma(w^T x) \\ 1 - 1/(1+\exp(-w^T x)) = \sigma(-w^T x) \end{cases}$

Learning: $w = \underset{w}{\operatorname{argmax}} P(w|x,y)$

Classification: Use $P(y|x,w) = \frac{1}{1+exp(-yw^T x)}$ and predict most likely class label.

Example: MLE for logistic regression

$\underset{w}{\operatorname{argmax}} P(y|x,w) = \underset{w}{\operatorname{argmin}} -\sum_{i=1}^n log P(y|x_i,w)$

$\underset{w}{\operatorname{argmin}} \sum_{i=1}^n log(1+exp(-yw^T x_i))$

Gradient for logistic regression

Loss function $l(w) = log(1+exp(-yw^T x))$

$\nabla_w l(w) = \frac{1}{1+exp(-yw^T x)} exp(-yw^T x)(-yx)$

$= \frac{1}{1+exp(+yw^T x)}(-yx)$

$= P(-y|x,w)(-yx)$

SGD for logistic regression

1. Initialize w
2. For t=1,2,...

Pick data $(x,y) \in_{u.a.r} D$

Compute probability of misclassification $P(-y|w,x) = \frac{1}{1+exp(yw^T x)}$

Update step $w = w + \eta_t yx P(-y|w,x)$

Logistic regression and regularization

L2 (Gaussian prior):

$\min_w \sum_{i=1}^n log(1+exp(-y_i w^T x_i)) + \lambda \|w\|_2^2$

L1 (Laplace):

$\min_w \sum_{i=1}^n log(1+exp(-y_i w^T x_i)) + \lambda \|w\|_1$

SGD for L2 logistic regression

Update step $w = w(1-2\lambda\eta_t) + \eta_t yx P(-y|w,x)$

Bayesian decision theory

Given:

- Conditional distribution over labels $P(y|x)$
- Set of actions $\mathcal{A}$
- Cost function $C : Y \times \mathcal{A} \to \mathbb{R}$

Pick action that minimizes the expected cost: $a^* = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \mathbb{E}_y[C(y,a)|x] = \sum_y P(y|x) * C(y,a)$

Example: Asymmetric costs

Est. cond. dist: $P(y|x,w) = Ber(\sigma(w^T x))$
Action set: $\mathcal{A} = \{+1,-1\}$ Cost function:

$$C(y,a) = \begin{cases} c_{FP}\text{ , if } y=-1 \text{ and } y=+1 \\ c_{FN}\text{ , if } y=+1 \text{ and } y=-1 \\ 0\text{ , otherwise} \end{cases}$$

The action that minimizes the expected cost is:

$C_+ = \mathbb{E}_y[C(y,+1)|x] = P(y=+1|x)\cdot 0 + (P(y=-1)|x)\cdot c_{FP}$

$C_- = \mathbb{E}_y[C(y,-1)|x] = P(y=+1|x)\cdot c_{FN} + P(y=-1|x)\cdot 0$

Predict +1 if $C_+ \le C_- \Leftrightarrow P(y=+1|x) \ge \frac{c_{FP}}{c_{FP}+c_{FN}}$

Doubtful logistic regression

Est. cond. dist: $P(y|x,w) = Ber(\sigma(w^T x))$
Action set: $\mathcal{A} = \{+1,-1,D\}$ Cost function:

$$C(y,a) = \begin{cases} 1\text{ , if } y \ne a \\ c\text{ , if } a = D \\ 0\text{ , otherwise} \end{cases}$$

The action that minimizes the expected cost

$$a^* = \begin{cases} y\text{ , } P(y|x) \ge 1-c \\ D\text{ , otherwise} \end{cases}$$

Linear regression

Est. cond. dist: $P(y|x,w) = \mathcal{N}(y;w^T x,\sigma^2)$
Action set: $\mathcal{A} = \mathbb{R}$ Cost function: $C(y,a) = (y-a)^2$

The action that minimizes the expected cost

$a^* = \mathbb{E}_y[y|x] = \int P(y|x)dy = w^T x$

Asymmetric cost for regression

Est. cond. dist: $P(y|x) = \mathcal{N}(y;w^T x,\sigma^2)$
Action set: $\mathcal{A} = \mathbb{R}$ Cost function: $C(y,a) = c_1 \max(y-a,0) + c_2 \max(a-y,0)$

We aim to minimize the expected risk by $\frac{\partial}{\partial a}$

$\frac{\partial}{\partial a}\mathbb{E}_y[C(y,a)|x] = \int_{-\infty}^{\infty} C(y,a)P(y|x)dy \overset{!}{=} 0$

$= -c_1 \int_a^{\infty} P(y|x)dy + c_2 \int_{-\infty}^{a} P(y|x)dy$

$= -c_1[1-\phi(a;w^T x,\sigma^2)] + c_2\phi(a;w^T x,\sigma^2)$

$\phi(a;w^T x,\sigma^2) = \frac{c_1}{c_1+c_2}$

using $\phi(u;v,w) = \phi((u-v)/\sqrt{w};0,1)$ and applying inverse CDF of std. ND ϕ^{-1} we get

$a^* = w^T x + \sigma\phi^{-1}(\frac{c_1}{c_1+c_2})$

Discriminative vs. Generative Modeling

Discriminative models: aim to estimate $P(y|x)$
generative models: aim to estimate joint distribution $P(y,x)$

Typical approach:

- Estimate prior on labels $P(y)$
- Estimate conditional distribution for each class y $P(x|y)$
- Obtain predictive distribution using Bayes rule $P(y|x) = \frac{1}{Z}P(y)P(x|y)$

Example MLE for P(y)

Want: $p = P(Y=1), P(y=-1) = 1-p$

Given: $D = \{(y_1,x_1),...,(y_n,x_n)\}, D_y = \{y_1,...,y_n\}$

$P(D_y|p) = \prod_{i=1}^n p^{[y_i=+1]}(1-p)^{[y_i=-1]}$

$= p^{n_+}(1-p)^{n_-}$ where $n_+ =$ of $y=+1$

$\frac{\partial}{\partial p} log P(D_y|p) = n_+\frac{1}{p} - n_-\frac{1}{1-p} \overset{!}{=} 0 \Rightarrow p = \frac{n_+}{n_+ + n_-}$

Example MLE for P=(x|y)

Assume: $P(X=x_i|y) = \mathcal{N}(x_i;\mu_{i,y},\sigma_{i,y}^2)$

Given: $D, D_{x_i|y} = \{x, \text{s.t. } x_{j,i} = x, y_i = y\}$

Thus MLE yields:

$\mu_{i,y} = \frac{1}{n_y}\sum_{x \in D_{x_i|y}} x$, where $n_y = |D_{x_i|y}|$

$\sigma_{i,y}^2 = \frac{1}{n_y}\sum_{x \in D_{x_i|y}} (x-\mu_{i,y})^2$

Deriving decision rule

In order to predict label y for new point x, use

$P(y|x) = \frac{1}{Z}P(y)P(x|y)$

$y = \underset{y}{\operatorname{argmax}} P(y|x)$

$= \underset{y}{\operatorname{argmax}} P(y)\prod_{i=1}^d P(x_i|y)$

$= \underset{y}{\operatorname{argmax}} log P(y) + \sum_{i=1}^d log P(x_i|y)$

Example: Gaussian Naive Bayes classifier

MLE for class prior: $P(Y=y) = p_y = \frac{\text{Count(Y=y)}}{n}$

MLE for feature distribution: $P(X_1,...,X_d|Y) = \prod_{i=1}^d P(X_i|Y) = \mathcal{N}(x_i|\mu_{y,i},\sigma_{y,i}^2)$

$\mu_{y,i} = \frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y} x_j, i$

$\sigma_{y,i}^2 = \frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y}(x_{j,i}-\mu_{y,i})^2$

Prediction given new point

$y = \underset{y}{\operatorname{argmax}} P(y|x) = \underset{y}{\operatorname{argmax}} P(y)\prod_{i=1}^d P(x|y)$

Example: Gaussian Bayes Classifier

MLE for class prior: $P(Y=y) = p_y = \frac{\text{Count(Y=y)}}{n}$

MLE for feature distribution: $P(x|Y) = \mathcal{N}(x;\mu_y,\Sigma_y)$ $\hat{\mu}_y = \frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y} x_i \in \mathbb{R}^d$

$\hat{\Sigma}_y = \frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y}(x_i-\hat{\mu}_y)(x_i-\hat{\mu}_y)^T \in \mathbb{R}^{d\times d}$

Categorical Naive Bayes Classifier

MLE class prior: $P(Y=y) = \frac{Count(Y=y)}{n}$

MLE for feature dist.:

$P(X_i=c|y) = \frac{Count(X_i=c,Y=y)}{Count(Y=y)}$

Outlier Detection

$P(x) = \sum_y P(x,y) = \sum_y P(y)P(x|y) \le \tau$

Latent: Missing Data

Mixture modeling

Model each cluster as probability distribution $P(x|\theta_j)$

Assuming data iid, likelihood is $P(D|\theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i|\theta_j)$

Choose parameters to minimize negative log likelihood

$L(D;\theta) = -\sum_i log \sum_j w_j P(x_i\theta_j)$

Soft-EM algorithm

While not converged

E-step: For each i and j calculate $\gamma_j^{(t)}(x_i)$

M-step: Fit clusters to weighted data points:

$w_j^{(}t) = \frac{1}{n}\sum_{i=1}^n \gamma_j^{(t)}(x_i)$

$\mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$

$\Sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i-\mu_j^{(t)})(x_i-\mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$

Log-likelihood

$l(\theta) = log P(\mathcal{D})$

$= \sum_{\substack{i=1 \\ y_i=\times}}^n log P(x_i;\theta) + \sum_{\substack{i=1 \\ y_i\ne\times}}^n log P(x_i,y_i;\theta)$

$= \sum_{\substack{i=1 \\ y_i=\times}}^n log \sum_{i=1}^m P(x_i,Y=j;\theta) +$

$\sum_{\substack{i=1 \\ y_i\ne\times}}^n log P(x_i,y_i;\theta)$

$= \sum_{\substack{i=1 \\ y_i=\times}}^n log \sum_{i=1}^m P(x_i|Y=j;\theta)P(Y=j|\theta) +$

$\sum_{\substack{i=1 \\ y_i\ne\times}}^n log P(x_i,y_i;\theta)$

Latent variable

We denote the latent variable indicating the component the point is sampled from by Z, which takes on values in $\{1,...,c\}$.

E-step: Posterior probabilities

$\gamma_j(x_i) = P(Y=j|x_i,\theta) = \frac{P(x_i|Y=j,\theta)P(Y=j|\theta)}{Z}$

M-step: maximizing expected log likelihood

$\underset{\theta}{\operatorname{argmax}} \mathbb{E}_\gamma[\log P(\mathcal{D})] =$

$\mathbb{E}_\gamma[\log \Pi_{i=1}^n P(x_i,y_i)] =$

$\sum_{i=1}^n \mathbb{E}_\gamma[\log P(x_i,y_i)] =$

$\sum_{i=1}^n \sum_{j=1}^k \gamma_j(x_i)\log(P(x_i|y_i)P(y_i))$

Timeseries

Markov Chain

Markov assumption: $P(Y_t|Y_{1:t-1}) = P(Y_t|Y_{t-1})$

Stationarity assumption:

$P(Y_{t+1}=y_1|Y_t=y_2) = P(Y_t=y_1|Y_{t-1}=y_2)$

Product rule:

$P(Y_t,...,Y_1) = P(Y_t|Y_{t-1},...,Y_1)\cdot...\cdot P(Y_1)$

Sum rule:

$P(Y_{t+2}|Y_{1:t}) = \sum_{Y_{t+1}^i} P(Y_{t+2}Y_{t+1}^1|Y_{1:t})$