Overfitting, Regularization and Support Vector Machine

Prof. Mingkui Tan

SCUT Machine Intelligence Laboratory (SMIL)

Contents

Contents

Basic Concepts about Machine Learning

- **Machine Learning compose of three parts:**
  - **Data**
  - **Model(function)**
  - **Loss(prediction)**

Task of Machine Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $)=$ "How are you"

- Image Recognition

$f($  $)=$ "Cat"

- Playing Go

$f($  $)=$ "5-5" (next move)

- Dialogue System

$f($ "Hi" $)=$ "Hello"

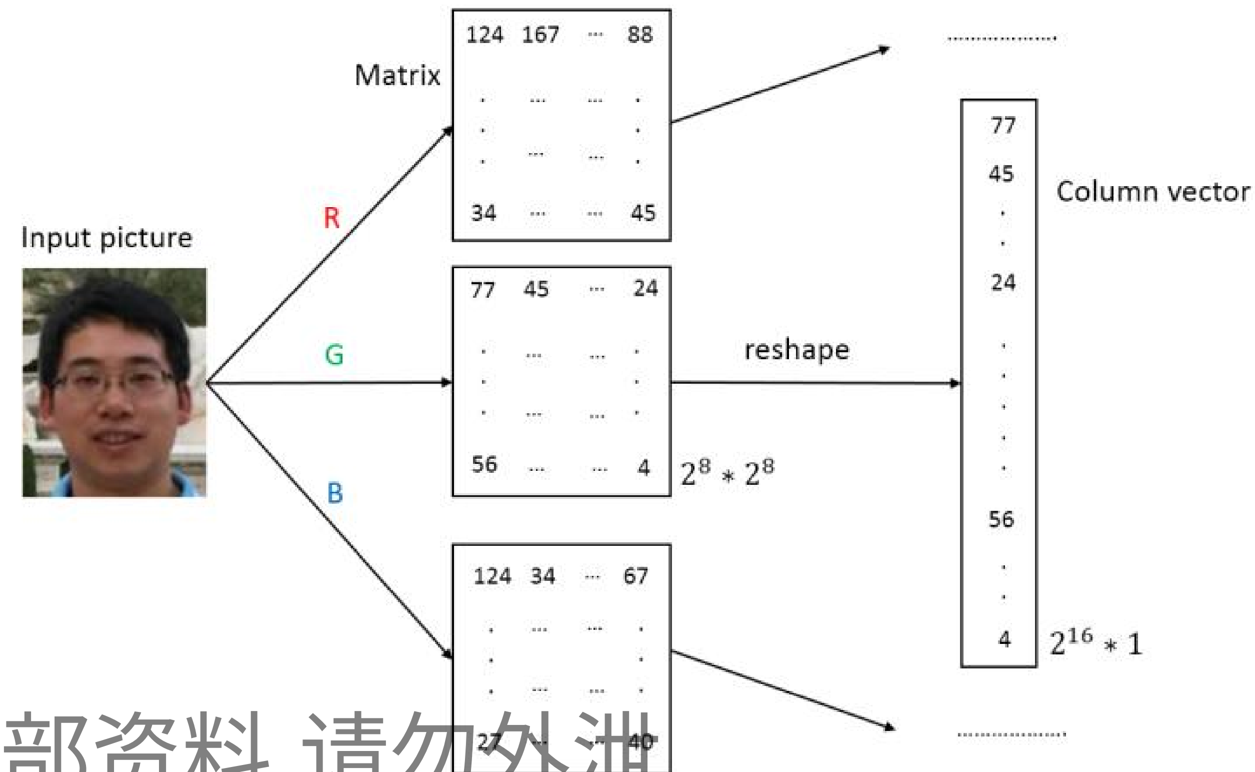
(what the user said) (system response)

Column Vector

- Data：

$$D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$$

- $\mathbf{x}$ is input, and we usually present it as column vector
- For example, x may be a picture stored as a matrix:
- Loss(prediction)



Regression

- We want to learn a function $f(x)$ to predict x by

$$\hat{y} = f(x)$$

- The prediction may be inconsistent with the ground truth.

- We measure the differences by some losses $l(f(x), \mathrm{y}_i) \geq 0$:
  - Absolute value loss:

$$l(f(x), \mathrm{y}_i) = |f(x) - \mathrm{y}_i|$$

  - Least squares loss:

$$l(f(x), \mathrm{y}_i) = \frac{1}{2}(f(x) - \mathrm{y}_i)^2$$

Regression

- The loss function $\mathcal{L}_D$ plays a major role in machine learning
- The smaller value of $\mathcal{L}_D$, the better

Find the best � by solving the following optimization problem:

$$f^* = \min_f \sum_{n=1}^{n} l(f(x), y_i)$$

Linear Regression

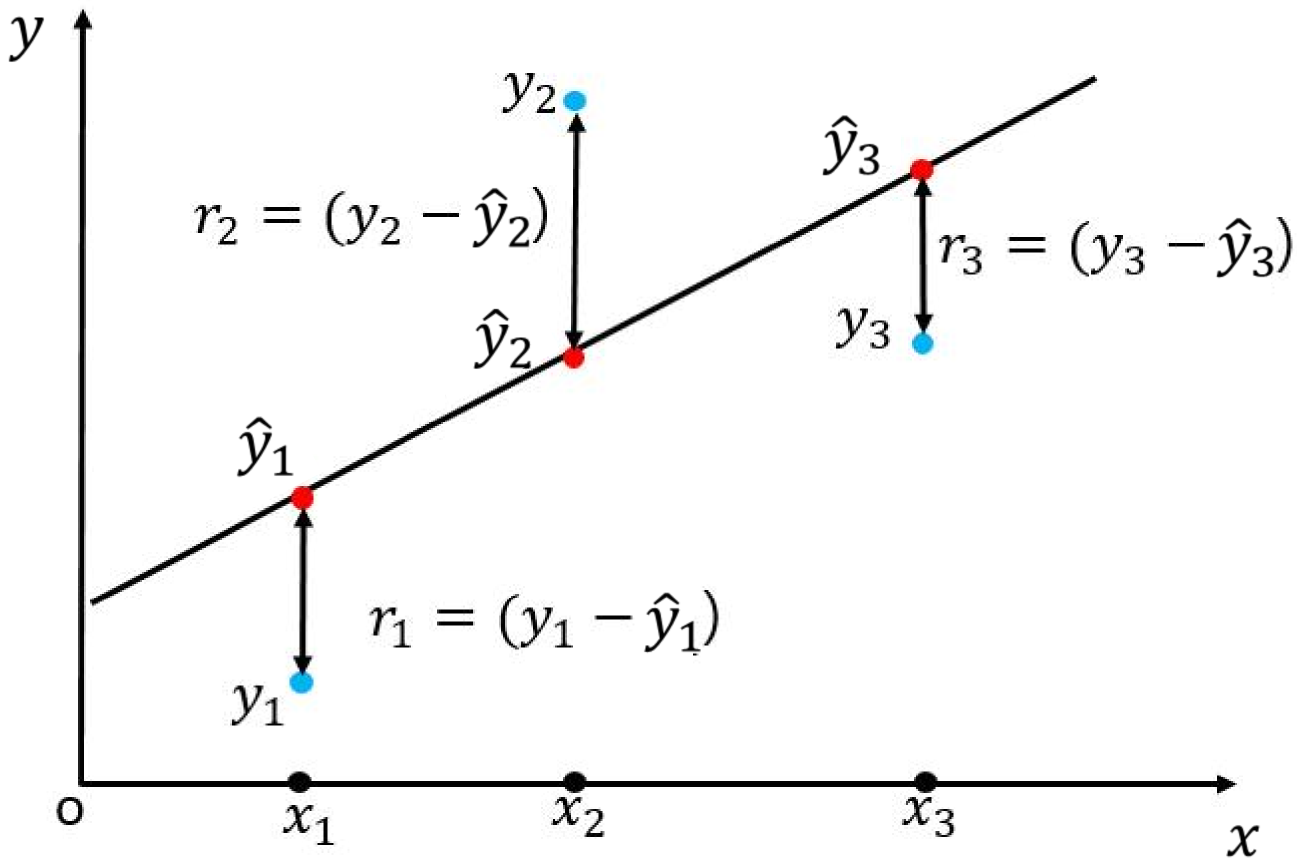
Learn $f(\mathbf{x};\mathbf{w},b)$ with

- Parameters: $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$
- Input: $\mathbf{x}$ where $x_i \in \mathbb{R}$, features for $i \in \{1, \cdots, m\}$
- Model Function:

$$
\begin{aligned}
f(\mathbf{x};\mathbf{w},b) &= w_1 x_1 + \cdots + w_m x_m + b \\
&= \sum_{i=1}^{m} w_i x_i + b \\
&= \mathbf{w}^{\mathrm{T}}\mathbf{x} + b
\end{aligned}
$$

Linear Regression

■ **What makes a good model?**



Least Square Regression

- Least squared loss

$$\mathcal{L}_D(\mathbf{w}) =+ \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i; w)\right)^2$$
$$= \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{w}||_2^2$$

Training: find minimizer of least squared loss

$$\mathbf{w}^* = arg\min_{w} \mathcal{L}_D(\mathbf{w})\mathrm{c}$$

Closed-form Solution

- First-order condition of the optimal solution:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}=0$$

- For the Least Regression problem, we have

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}=-\mathbf{X}^{\mathrm{T}}\mathbf{y}+\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}=0$$
$$\Rightarrow \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}=\mathbf{X}^{\mathrm{T}}\mathbf{y}$$
$$\Rightarrow \mathbf{w}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

- We obtain the optimal $\mathbf{w}^*$ by

$$\mathbf{w}^*=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Issue of the Closed-form Solution

- Closed-form solution: $\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$
  - The matrix $(\mathrm{X}^T\mathrm{X})^{-1}$ may not be invertible, which means the matrix may have infinite number of solutions!

Regularized Least Square (RLS) Regression

- Impose regularization on **w**:

$$\mathcal{L}_D(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||_2^2 + \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i; w)\right)^2$$
$$= \frac{\lambda}{2}||\mathbf{w}||_2^2 + \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{w}||_2^2$$

- Here, $\frac{\lambda}{2}||\mathbf{w}||_2^2$ is called Regularizer, λ is called **trade-off parameter** or **regularization parameter**

Training: find minimizer of least squared loss

$$\mathbf{w}^* = \underset{w}{argmin}\, \mathcal{L}_D(\mathbf{w})$$

Closed-form Solution for Regularized Least Square(RLS)

- First-order condition of the optimal solution:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}=0$$

- For the Least Regression problem, we have

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}&=\lambda\mathbf{w}-\mathbf{X}^{\mathrm{T}}\mathbf{y}+\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}=0\\ &\Rightarrow(\lambda\mathbf{I}+\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{w}=\mathbf{X}^{\mathrm{T}}\mathbf{y}\\ &\Rightarrow\mathbf{w}=(\mathbf{X}^{\mathrm{T}}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\end{aligned}$$

- We obtain the optimal $\mathbf{w}^*$ by

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

Issue of the Closed-form Solution

- Closed-form solution: $\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$
  - The inverse of a large matrix needs huge memory
  - The inverse takes $O(m^3)$ complexity to compute

Contents

Machine Learning

- Training Procedure:

  - Identify a set of hypotheses $f(\boldsymbol{x}; \boldsymbol{w})$
  - Define a loss criterion $\mathcal{L}_D$
  - Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$

$$\mathbf{w}^* = arg\min_{w} \mathcal{L}_D(\mathbf{w})$$

- Learning is done through optimization

Main Tool: Gradient

- Typical case (with possibly parameterized g):

$$\mathcal{L}_D(\mathbf{w}) : \mathbb{R}^n \mapsto \mathbb{R}$$

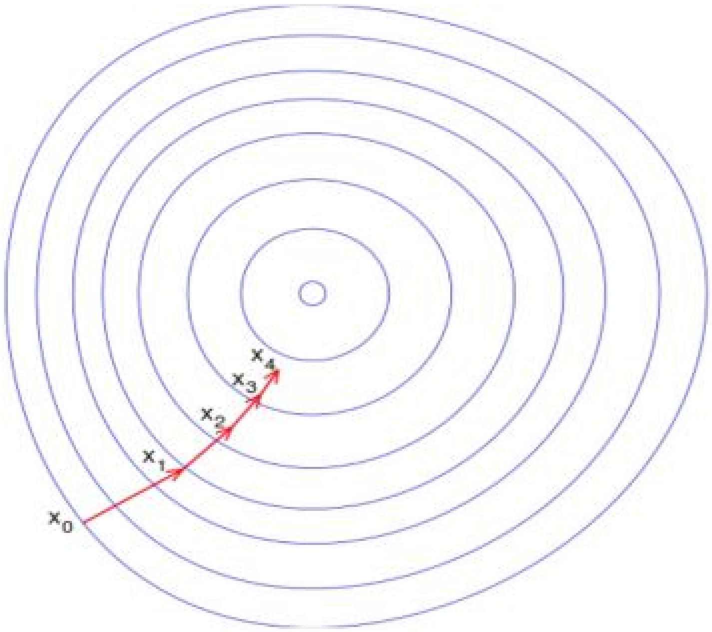
- Gradient (vector of partial derivatives)

$$\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}_D(w_1)}{\partial w_1} \\ \frac{\partial \mathcal{L}_D(w_2)}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}_D(w_n)}{\partial w_n} \end{bmatrix}$$

(We will always write as column vectors)

Decent Direction

- We use $\mathbf{d} = -\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$ as the direction of optimization



- Why $\mathcal{L}_{\mathcal{D}}(\mathbf{w}') = \mathcal{L}_{\mathcal{D}}(\mathbf{w} + \eta \mathbf{d}) \leq \mathcal{L}_{\mathcal{D}}(\mathbf{w})$, $\eta \to 0^+$?

Descent Direction

Proof:

By Taylor expansion, when $\eta \to 0^+$:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w} + \eta\mathbf{d}) = \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \eta\mathbf{d} + o(\eta\mathbf{d})$$

$$= \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \eta' \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \mathbf{d}$$

Note that $\eta' > 0$ and

$$\eta' \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \mathbf{d} = -\eta' \mathbf{d}^{\mathrm{T}}\mathbf{d} \leq 0$$

We have:

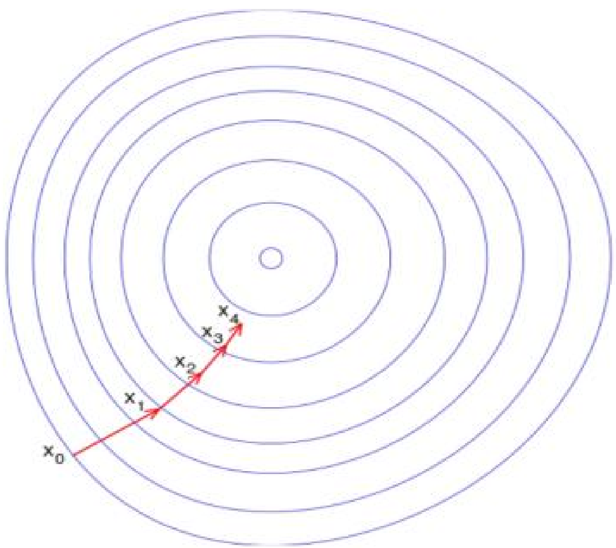
$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}') = \mathcal{L}_{\mathcal{D}}(\mathbf{w} + \eta\mathbf{d}) \leq \mathcal{L}_{\mathcal{D}}(\mathbf{w})$$

Gradient Descent

Minimize loss by repeated gradient steps (when no closed form):

- Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$
- Update parameters with learning rate η

$$\mathbf{w}' = \mathbf{w} - \eta \frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$$



- Figure: Gradient steps on a simple m=2 loss function.

Appropriate Value of Learning Rate

Learning rate η has a large impact on convergence

- Too large $\eta \Rightarrow$ oscillate and may even diverge
- Too small $\eta \Rightarrow$ too slow to converge

Adaptive learning rate (For example) :

- Set larger learning rate at the beginning
- Use relatively smaller learning rate in the later epochs
- Decrease the learning rate:

$$\eta_{k+1} = \frac{\eta_k}{k+1}$$

Thank You