Regression

Regression

1. Linear Regression

- Problem Setting
  - **Data**: observed pairs (x, y), where $x \in \mathbb{R}^{n+1}$ (**input**) & $y \in \mathbb{R}$ (**output**)
  - **Goal**: find a linear function of the unknown ws:

$$f: \mathbb{R}^n \rightarrow \mathbb{R} \quad \text{s.t.} \quad \forall (x, y): y \approx f(x, w)$$

- Model

$$\hat{y}_i = \sum_{j=0}^{n} w_j x_{ij}$$

$$\hat{y} = Xw$$

$$\begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix}$$

  - $\hat{y}_i$: the model prediction for the ith observation
  - x_{ij}: the jth feature in the ith observation
  - w_j: the param for the jth feature
  - m: #observations
  - n: #features

- Learning
  - **Aim**: find the optimal w that minimizes a loss function / cost function
  - **Assumption**: $m \gg n$
  - **Loss Function**: **OLS [Ordinary Least Squares]**

$$\mathcal{L}(w) = \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

  - **Minimization Method 1: Gradient Descent** (the practical solution)

$$w_j \coloneqq w_j - \alpha \frac{\partial \mathcal{L}(w)}{\partial w_j} \quad | \quad \alpha\text{: learning rate}$$

- **Stochastic GD**: use 1 observation for each GD step

$$w_j \coloneqq w_j - \alpha(\hat{y}_i - y_i)x_{ij}$$

- **Mini-batch GD**: use mini-batches of size m' for each GD step

$$w_j \coloneqq w_j - \alpha \sum_{i=1}^{m'} (\hat{y}_i - y_i)x_{ij}$$

- **Batch GD** (LMS): use the entire training set for each GD step

$$w_j \coloneqq w_j - \alpha \sum_{i=1}^{m} (\hat{y}_i - y_i)x_{ij}$$

- Extra: **Newton's Method**

  ⇒ Newton's formula

$$w \coloneqq w - \frac{f(w)}{f'(w)}$$

  ⇒ Newton's method in GD

$$w \coloneqq w - H^{-1}\nabla_w \mathcal{L}(w)$$

  where H is Hessian Matrix:

$$H_{ij} = \frac{\partial^2 \mathcal{L}(w)}{\partial w_i \partial w_j}$$

  ⇒ Newton vs GD

  - YES: faster convergence, fewer iterations
  - NO: expensive computing ← inverse of a matrix

- **Minimization Method 2: Normal Equation** (the exact solution)

$$w_{LS} = (X^T X)^{-1} X^T Y$$

$$w_{LS} = \left(\sum_{i=1}^{m} x_i x_i^T\right)^{-1} \left(\sum_{i=1}^{m} y_i x_i\right)$$

- Derivation (matrix)

$$\begin{aligned}
\nabla_w \mathcal{L}(w) &= \nabla_w (Xw - y)^T (Xw - y) \\
&= \nabla_w \operatorname{tr}(w^T X^T X w - w^T X^T y - y^T X w + y^T y) \\
&= \nabla_w \big(\operatorname{tr}(w^T X^T X w) - 2\operatorname{tr}(y^T X w)\big) \\
&= 2X^T X - 2X^T y \\
&\Rightarrow X^T X w - X^T y = 0
\end{aligned}$$

- Derivation (vector)

$$\begin{aligned}
\nabla w \mathcal{L}(w) &= \sum_{i}^{m} \nabla_w (w^T x_i x_i^T w - 2 w^T x_i y_i + y_i^2) \\
&= -\sum_{i=1}^{m} 2 y_i x_i + \left(\sum_{i=1}^{m} 2 x_i x_i^T\right) w \\
&\Rightarrow \left(\sum_{i=1}^{m} x_i x_i^T\right) w - \sum_{i=1}^{m} y_i x_i = 0
\end{aligned}$$

- GD vs Normal Equation

	GD	**Normal Equation**
Advantage	Faster computing Less computing power	The exact solution
Disadvantage	Hard to reach the exact solution	$(X^T X)^{-1}$ must be full-rank

- Full rank: when the $m \times n$ matrix X has $\geq n$ linearly independent rows (i.e. any point in $\mathbb{R}^n$ can be reached by a weighted combination of n rows of X)

- Probabilistic Interpretation
  - **Probabilistic Model: Gaussian**

$$p(y_i|x_i,w) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}$$

    - $y_i = w^T x_i + \epsilon_i$
    - $\epsilon_i \sim N(0,\sigma)$
  - **Likelihood Function**

$$L(w) = \prod_{i=1}^{m} p(y_i|x_i,w) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}$$

  - **Log Likelihood**

$$\begin{aligned} l(w) &= \ln L(w) \\ &= \ln \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}} \\ &= \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}} \\ &= m \ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \end{aligned}$$

    - Why log?

      ⇒ Log = monotonic & increasing on [0,1] →

$$\operatorname*{argmax}_w L(w) = \operatorname*{argmax}_w \ln L(w)$$

      ⇒ Log simplifies calculation (especially & obviously for $\prod$)
  - **MLE [Maximum Likelihood Estimation]**

$$\begin{aligned} \operatorname*{argmax}_w l(w) &= \operatorname*{argmax}_w \left(m \ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \right) \\ &= \operatorname*{argmax}_w \left(-\sum_{i=1}^{m} (y_i - w^T x_i)^2 \right) \end{aligned}$$

$$= \underset{w}{\operatorname{argmin}} \left(\sum_{i=1}^{m} (y_i - w^T x_i)^2 \right)$$

$$= \underset{w}{\operatorname{argmin}} \|y - Xw\|^2$$

$\Rightarrow$ OLS & MLS share the exact same solution.

- **Expected Value**

$$\begin{aligned} \mathbb{E}[w_{ML}] &= \mathbb{E}[(X^T X)^{-1} X^T y] \\ &= (X^T X)^{-1} X^T X w \\ &= w \end{aligned}$$

- **Variance**

$$\begin{aligned} \operatorname{Var}[w_{ML}] &= \mathbb{E}[(w_{ML} - \mathbb{E}[w_{ML}])(w_{ML} - \mathbb{E}[w_{ML}])^T] \\ &= \mathbb{E}[w_{ML} w_{ML}^T] - \mathbb{E}[w_{ML}]\mathbb{E}[w_{ML}]^T \\ &= (X^T X)^{-1} X^T \mathbb{E}[yy^T] X (X^T X)^{-1} - ww^T \\ &= (X^T X)^{-1} X^T (\sigma^2 I + Xww^T X^T) X (X^T X)^{-1} - ww^T \qquad (1) \\ &= \sigma^2 (X^T X)^{-1} \end{aligned}$$

(1):

$$\begin{aligned} \sigma^2 &= \operatorname{Var}[y] = \mathbb{E}[(y - \mu)(y - \mu)^T] \\ &= \mathbb{E}[yy^T] - 2\mu\mu^T + \mu\mu^T \\ &\Rightarrow \mathbb{E}[yy^T] = \sigma^2 + \mu\mu^T \end{aligned}$$

- Summary
  - Assumption: Gaussian: $y \sim N(Xw, \sigma^2 I)$
  - Expected Value: $\mathbb{E}[w_{ML}] = w$
  - Variance: $\operatorname{Var}[w_{ML}] = \sigma^2 (X^T X)^{-1}$
  - Problem: w_{ML} becomes huge when $\operatorname{Var}[w_{ML}]$ is too large

- **Regularization**
  - Intuition: To prevent the problem above, we want to constrain w

$$w_{OP} = \underset{w}{\operatorname{argmin}} \|y - Xw\|^2 + \lambda g(w)$$

- ♦ $\lambda > 0$: regularization param
- ♦ $g(w) > 0$: penalty function

- o Examples: Ridge Regression, LASSO Regression, …

2. Polynomial Regression

- Polynomial Reg ∈ LinReg (f = a linear func of unknown params w)
- Different Preprocessing:

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} & x_{11}^2 & \cdots & x_{1n}^p \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} & x_{m1}^2 & \cdots & x_{mn}^p \end{bmatrix}$$

with the width of $p \times n + 1$.

- Everything else is exactly the same as LinReg
- Example models:
  - o 3rd order with 1 feature: $y_i = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3$
  - o 2nd order with 2 features: $y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i1}^2 + w_4 x_{i2}^2$
- Further extensions

We can generalize LinReg model as:

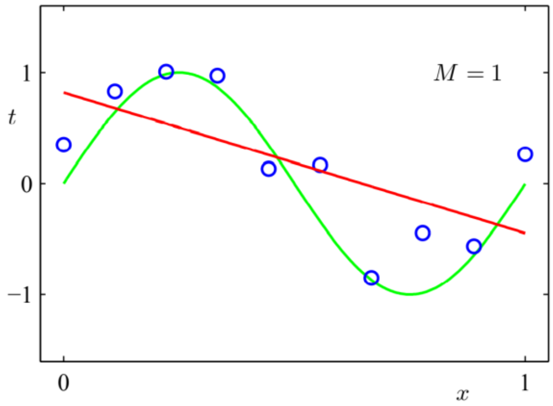
$$\hat{y}_i \approx f(x_i, w) = \sum_{s=1}^{S} g_s(x_i) w_s$$

where $g_s(x_i)$ can be any func of x_1 (e.g. $e^{x_{ij}}, \ln x_{ij}$, …)

While this seems useful, not really. Most of the patterns in the world can be represented into the first 3 orders of polynomials. Thx God for simplifying things for us.

3. Locally Weighted Linear Regression

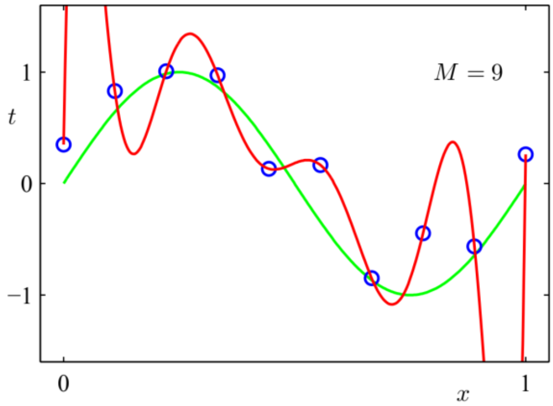
- <u>Problem Setting</u>
  - **Underfitting**: the model barely fits the data points



One single line is usually not enough to capture the pattern of x&y.

In order to get a better fit, we add more polynomial features x^j.

  - **Overfitting**: the model fits the data points too well that it can't be used on other data.



When we add too much (e.g. $\hat{y} = \sum_{j=0}^{9} w_j x^j$), the model captures the pattern of the given data too well and thus become useless on other data.

- <u>Intuition</u>: when we would like to estimate y at a certain x, instead of applying the original LinReg, we take a subset of data points (x_i, y_i) around the x and then do LinReg on that subset only, so that we can get a more accurate estimation.

- <u>Model</u>: **Locally Weighted Regression**
  - Original LinReg

$$w \leftarrow \underset{w}{\operatorname{argmin}} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$

We find the w that minimizes the cost function / maximizes the likelihood function → optimize our model to fit the data.

  - LWR

$$w \leftarrow \underset{w}{\operatorname{argmin}} \sum_{i=1}^{m} e^{-\frac{(x_i - x)^2}{2\tau^2}} (y_i - w^T x_i)^2$$

We add the weight function $\mathscr{w}_i = e^{-\frac{(x_i - x)^2}{2\tau^2}}$ to OLS.

    - **Numerator**:

$$\begin{gathered}\text{If } |x_i - x| \text{ is small} \Longrightarrow \mathscr{w}_i \approx 1 \\ \text{If } |x_i - x| \text{ is large} \Longrightarrow \mathscr{w}_i \approx 0\end{gathered}$$

    - **Bandwidth Param**: τ (how fast the weight of x_i falls off the query point x)

$$\begin{gathered}\text{When } \tau \gg 1 \Longrightarrow \text{LWR} \approx \text{LinReg} \\ \text{When } \tau \ll 1 \Longrightarrow \text{LWR} \rightarrow \text{overfitting}\end{gathered}$$

    - **Exact Solution**:

$$\begin{aligned}\nabla_w \mathcal{L}(w) &= \nabla_w \mathscr{w} (Xw - y)^T (Xw - y) \\ &\Rightarrow X^T \mathscr{w} X w - X^T \mathscr{w} y = 0 \\ &\Rightarrow w = (X^T \mathscr{w} X)^{-1} X^T \mathscr{w} y\end{aligned}$$

4. Ridge Regression

- Problem Setting: Regularization
- Model:

$$w_{RR} = \underset{w}{\operatorname{argmin}} \|y - Xw\|^2 + \lambda \|w\|_2^2$$

  - λ: regularization param

$$\begin{aligned} &\text{If } \lambda \to 0 \implies w_{RR} \to w_{LS} \\ &\text{If } \lambda \to \infty \implies w_{RR} \to 0 \end{aligned}$$

  - $g(w) = \|w\|_2^2 = w^T w$: L2 penalty function

- Solution:

$$\mathcal{L} = (y - Xw)^T (y - Xw) + \lambda w^T w$$

$$\nabla_w \mathcal{L} = -2X^T y + 2X^T X w + 2\lambda w = 0$$

$$\Rightarrow w_{RR} = (X^T X + \lambda I)^{-1} X^T y$$

- Data Preprocessing: **Standardization**
  - y:

$$y \leftarrow y - \bar{y}$$

  - x:

$$x_{ij} \leftarrow \frac{x_{ij} - \bar{x}_j}{\sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(x_{ij} - \bar{x}_j\right)^2}}$$

- **SVD [Singular Value Decomposition]**
  - Definition: We can write any $m \times n$ ($m > n$) matrix X as $X = USV^T$
    - ♦ U: left singular vectors ($m \times r$)

      $\Rightarrow$ orthonormal in cols (i.e. $U^T U = I$)
    - ♦ S: singular values ($r \times r$)

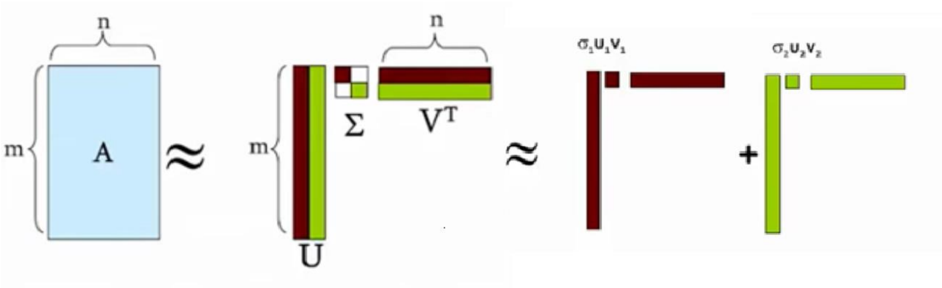
      $\Rightarrow$ non-negative diagonal (i.e. $S_{ii} \geq 0, S_{ij} = 0 \ \forall i \neq j$)

- $\Rightarrow$ sorted in decreasing order (i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$)
- ♦ V: right singular vectors ($n \times r$)
  - $\Rightarrow$ orthonormal (i.e. $V^T V = VV^T = I$)
- ♦ m: #samples
- ♦ n: #features
- ♦ r: #concepts ($r = \text{rank}(X)$)
- ♦ σ_i: the strength of the ith concept

- o Properties:
  - ♦ $X^T X = VS^2V^T$
  - ♦ $XX^T = US^2U^T$
  - ♦ If $\forall i: S_{ii} \neq 0 \implies (X^T X)^{-1} = VS^{-2}V^T$

- o Intuition:

$$X = USV^T = \sum \sigma_i \boldsymbol{u_i} \times \boldsymbol{v_i^T}$$

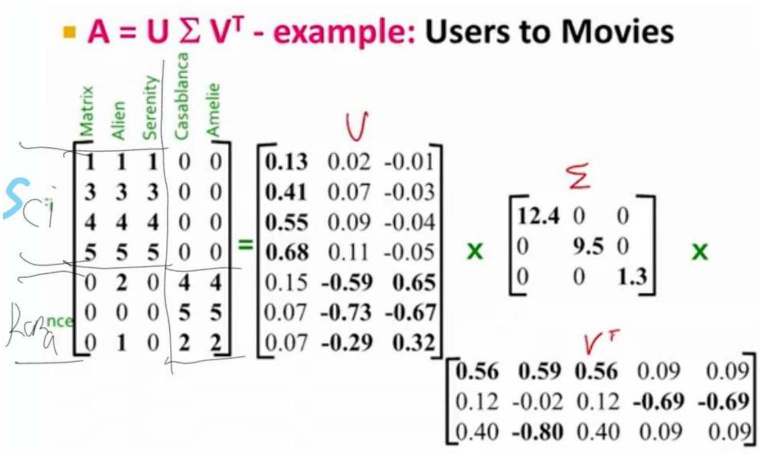


Why do we need this?

As an example, suppose we would like to analyze a dataset of the relationship between Users & Movies, in which:

- ♦ Each row = a user
- ♦ Each col = a movie
- ♦ Each entry X_{ij} = the rating of movie j from user i (0=unwatched, 1=hate, 5=love)

And here is the situation:



Cited from Stanford's Mining Massive Datasets

- U = "User-to-Concept" similarity matrix
  - $\Rightarrow$ $U[:,1]$ = Sci-fi concept of users
  - $\Rightarrow$ $U[:,2]$ = Romance concept of users

- S = "Strength of Concept" matrix
  - $\Rightarrow$ $S[1,1]$ = Strength of Sci-fi concept
  - $\Rightarrow$ $S[2,2]$ = Strength of Romance concept
  - $\Rightarrow$ $\because S[3,3]$ is very small $\therefore$ we ignore this concept together with $U[:,3]$ & $V^T[3,:]$

- V^T = "Movie-to-Concept" similarity matrix
  - $\Rightarrow$ $V^T[1,1:3]$ = Sci-fi concept of the Sci-fi movies
  - $\Rightarrow$ $V^T[2,4:5]$ = Romance concept of the Romance movies

- <u>Calculation of SVD</u>:
  - Step 1: $X^TX = VS^2V^T \Longrightarrow$ calculate V, S^2
    - $\Rightarrow$ $S^2 \ni$ eigenvalues
    - $\Rightarrow$ $V \ni$ eigenvectors
  - Step 2: $XV = US^2 \Longrightarrow$ calculate U

- **RR vs LS** – through SVD

$$w_{LS} = (X^T X)^{-1} X^T y \Leftrightarrow w_{RR} = (\lambda I + X^T X)^{-1} X^T y$$

  - Problems with LS:
    - $\mathrm{Var}[w_{ML}] = \sigma^2 (X^T X)^{-1} = \sigma^2 V S^{-2} V^T$
      When S_{ii} is very small for some values of i, $\mathrm{Var}[w_{ML}]$ is very large.
    - $y_{new} = x_{new}^T w_{LS} = x_{new}^T (X^T X)^{-1} X^T y = x_{new}^T V S^{-1} U^T y$
      When S^{-1} has very large values, our prediction will be very unstable.
  - **LS = a special case of RR**:

$$\begin{aligned}
w_{RR} &= (\lambda I + X^T X)^{-1} X^T y \\
&= (\lambda I + X^T X)^{-1} (X^T X)(X^T X)^{-1} X^T y \\
&= (\lambda I + X^T X)^{-1} (X^T X) w_{LS} \\
&= [(X^T X)(\lambda (X^T X)^{-1} + I)]^{-1} (X^T X) w_{LS} \\
&= (\lambda (X^T X)^{-1} + I)^{-1} w_{LS} \\
&= (\lambda V S^{-2} V^T + I)^{-1} w_{LS} \\
&= V (\lambda S^{-2} + I)^{-1} V^T w_{LS} \\
&\coloneqq V M V^T w_{LS}
\end{aligned}$$

    where $M = (\lambda S^{-2} + I)^{-1}$ is a diagonal matrix with $M_{ii} = \frac{S_{ii}^2}{\lambda + S_{ii}^2}$.

$$\begin{aligned}
w_{RR} &\coloneqq V M V^T w_{LS} \\
&= V (\lambda S^{-2} + I)^{-1} V^T (V S^{-1} U^T y) \\
&= V S_\lambda^{-1} U^T y
\end{aligned}$$

    where S_λ^{-1} is a diagonal matrix with ${S_\lambda^{-1}}_{ii} = \frac{S_{ii}}{\lambda + S_{ii}^2}$.

    Therefore, we get another clearer expression of the relationship between RR & LS:

$$w_{LS} = V S^{-1} U^T y \Leftrightarrow w_{RR} = V S_\lambda^{-1} U^T y$$

    And w_{LS} is simply a special case of w_{RR} where $\lambda = 0$.

- **RR = a special case of LS**:

  If we just do some simple preprocessing to our model $y \approx X'w$:

$$\begin{bmatrix} y \\ 0 \\ \vdots \\ 0 \end{bmatrix} \approx \begin{bmatrix} - & X & - \\ \sqrt{\lambda} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

  Now we have the exact same loss function:

$$(y - X'w)^T(y - X'w) = \|y - Xw\|^2 + \lambda\|w\|^2$$

- Probabilistic Interpretation

	LS	**RR**
Mean	$\mathbb{E}[w_{LS}] = w$	$\mathbb{E}[w_{RR}] = (\lambda I + X^TX)^{-1}X^TXw$
Variance	$\mathrm{Var}[w_{LS}] = \sigma^2(X^TX)^{-1}$	$\mathrm{Var}[w_{LS}] = \sigma^2 Z(X^TX)^{-1}Z^T$, where $Z = (I + \lambda(X^TX)^{-1})^{-1}$

5. Lasso Regression

- Everything is the same as Ridge Regression except the **model**:

$$w_{lasso} = \underset{w}{\operatorname{argmin}} \|y - Xw\|^2 + \lambda\|w\|_1$$

  - $g(w) = \|w\|_1 = |w|$: L1 penalty function

- **Solution**: we are yet able to find a solution to the multivariate Lasso because of the absolute value.