Linear Regression
线性回归和正则项

Jingbo Xia

Huazhong Agricultural University

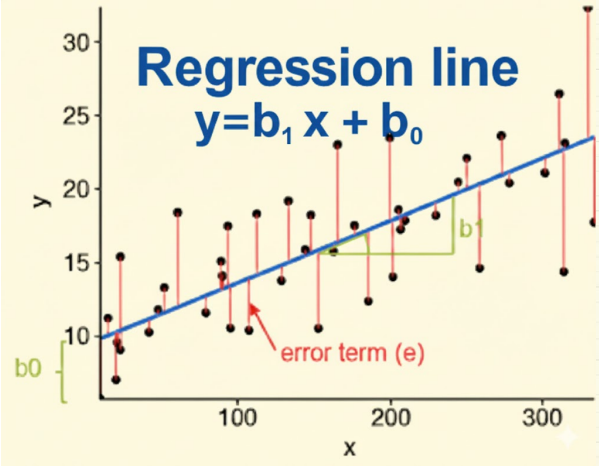
xiajingbo.math@gmail.com

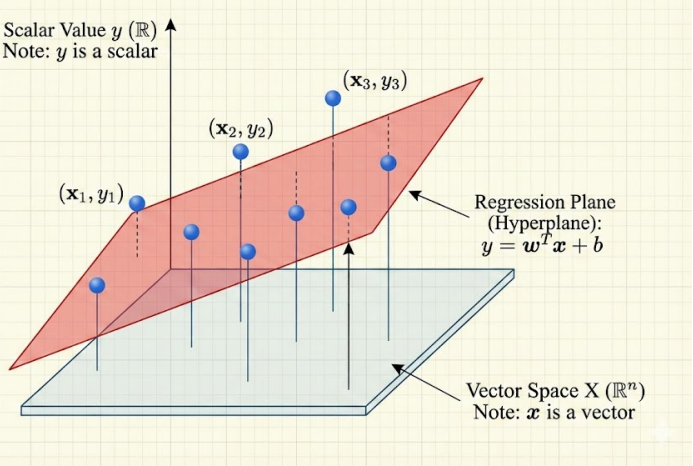
2025-11-26

Table of contents I

Linear Regression

线性回归是一个经典的回归学习算法，有利于理解 Loss 函数和正则项的作用，这其中需要适量的矩阵微分的基础。同时，也需要针对 $y = \vec{w}^T\vec{x} + b$获得一个高维空间的理解。



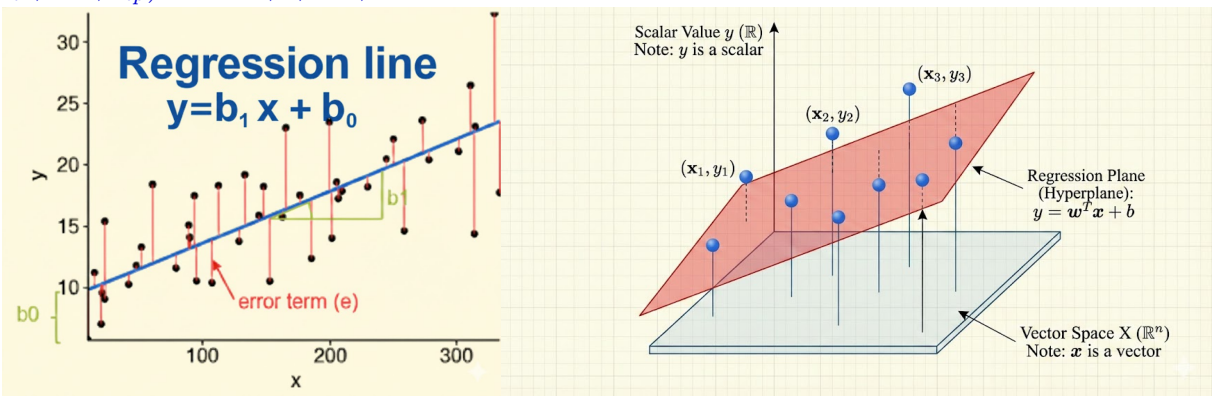


Outline

Linear Regression with Least Square I

最小二乘线性回归的建模和求解

观测值集合 Obs: $\{\vec{x}_i, y_i\}$.
There are n p-demensional sample data x_i, and their regression values are $y_i \in \mathbb{R}$, here, $\vec{x}_i = (x_{i1}, \cdots, x_{ip})^T$, $i = 1, 2, \cdots, n$.



Linear Regression with Least Square II

最小二乘线性回归的建模和求解

The goal of the linear regression model is to determine a regression function $f(\vec{x}, \vec{w}) = \vec{w}^T\vec{x}$ such that the predicted values $\tilde{y}_i = f(\vec{x}_i, \vec{w})$ approximate the observed values y_i:

$$y_i \approx \tilde{y}_i = f(\vec{x}_i, \vec{w}) = \sum_{j=1}^{p} w_j x_{ij} = \vec{w}^T\vec{x}_i$$

where:
$\vec{w} = (w_1, \ldots, w_p)^T \in \mathbb{R}^p$ is the weight vector (or parameter vector).
$\vec{x}_i = (x_{i1}, \ldots, x_{ip})^T \in \mathbb{R}^p$ is the feature vector for the i-th observation. [a]

[a] 该公式实际是一个简洁表示，完整表述应为：$f(\vec{x}_i, \vec{w}, b) = \sum_{j=1}^{p} w_j x_{ij} + b = \vec{w}^T\vec{x}_i + b$.
我们重写公式为：$f(\vec{x}_i, \vec{w}, b) := \hat{f}(\hat{\vec{x}}_i, \hat{\vec{w}})$.
此处我们使用 $\hat{\vec{x}}_i$ 和 $\hat{\vec{w}}$ 作为辅助记号：$\hat{\vec{x}}_i = (\vec{x}_i^T, 1)^T = (x_{i1}, \ldots, x_{ip}, 1)^T$, $\hat{\vec{w}} = (\vec{w}^T, b)^T = (w_1, \ldots, w_p, b)^T$.
不失一般性，我们在后面使用简洁的公式表示。

Linear Regression with Least Square III

最小二乘线性回归的建模和求解

The square sum of error terms will be the minimum goal.

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 \tag{1}$$

很多时候，上述损失函数被称为“均方误差”(Mean Squared Error, **MSE**)。

Linear Regression with Least Square IV

最小二乘线性回归的建模和求解

均方误差（MSE）损失函数的特性分析

均方误差（Mean Squared Error, MSE）是回归任务中最常用的损失函数之一，其标准定义为：

$$\mathcal{J}_{\text{MSE}}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2$$

其中，n 为样本数量，y_i 为第 i 个样本的真实值，$f(\vec{x}_i, \vec{w})$ 为模型在参数 $\vec{w}$ 下对输入 $\vec{x}_i$ 的预测值。

Linear Regression with Least Square V

最小二乘线性回归的建模和求解

MSE 的优点

MSE 具有以下显著优点，使其在众多场景中成为首选：

1. **处处可导性**：MSE 函数在其定义域内处处连续可微，这一光滑特性使其能够与基于梯度的优化算法（如随机梯度下降）完美协同，便于求解最优模型参数。
2. **对大误差的强惩罚**：其损失计算中的平方项 $(y_i - \hat{y}_i)^2$ 使得预测值与真实值之间的较大偏差会被指数级放大。这一特性驱使模型在训练过程中优先修正严重的预测错误，有利于快速降低整体误差水平。
3. **凸函数性**：当模型 $f(\vec{x}, \vec{w})$ 为参数的线性函数时，MSE 损失函数是关于参数 $\vec{w}$ 的凸函数。凸性保证了优化过程能够收敛至全局最优解，避免了陷入局部极值点的问题，为参数估计提供了理论上的可靠性。

Linear Regression with Least Square VI

最小二乘线性回归的建模和求解

MSE 的局限与缺点

尽管优势突出，MSE 也存在固有的局限性，在使用时需审慎考虑：

1. **对异常值高度敏感**：平方项的放大效应是一把“双刃剑”。当数据中存在异常值（Outliers）时，其所产生的巨大误差会被平方操作进一步放大，导致损失函数值被少数异常样本所主导。这可能会使模型参数为拟合这些异常点而发生严重偏离，从而损害模型在正常数据上的泛化性能。
2. **损失量纲问题**：由于对误差进行了平方运算，MSE 损失值的量纲变为原目标变量量纲的平方。这为损失值的直观理解和不同量纲数据集间的横向比较带来了不便。有时会采用其平方根形式（RMSE）以恢复量纲的一致性。

Linear Regression with Least Square VII

最小二乘线性回归的建模和求解

与其他回归损失函数的对比

Linear Regression with Least Square VIII

最小二乘线性回归的建模和求解

表 1: 常见回归损失函数对比

损失函数名称	数学表达式（简写）	核心特性与适用场景
均方误差（MSE）	$\frac{1}{n}\sum(y-\hat{y})^2$	对大误差惩罚严厉，收敛快，但对异常值敏感。适用于噪声较小、异常值少的回归问题。
平均绝对误差（MAE）	$\frac{1}{n}\sum\lvert y-\hat{y}\rvert$	对误差进行线性惩罚，对异常值的鲁棒性更强，但在零点不可导，优化相对平缓。
Huber Loss	$\begin{cases}\frac{1}{2}(y-\hat{y})^2, & \text{若}\lvert y-\hat{y}\rvert\le\delta\\ \delta\lvert y-\hat{y}\rvert-\frac{1}{2}\delta^2, & \text{其他}\end{cases}$	MSE 与 MAE 的折衷，在误差较小时为二次，较大时为线性，兼具光滑性与鲁棒性。

Linear Regression with Least Square IX

最小二乘线性回归的建模和求解

综上所述，MSE 因其良好的数学性质和优化特性成为回归分析的基石。然而，在实际应用中，需根据数据的具体情况（如是否存在显著异常值）审慎选择损失函数，或考虑采用如 Huber 损失等更鲁棒的替代方案，以在优化效率与模型稳健性之间取得平衡。

Linear Regression with Least Square X

最小二乘线性回归的建模和求解

Mean Squared Error (**MSE**) is the minimum goal.

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 \tag{2}$$

那么线性回归的优化目标就是寻找最佳的参数 $\vec{w}^*$, 使得

$$\vec{w}^* = \arg\min_{\vec{w}\in\mathbb{R}^p} \mathcal{J}(\vec{w})$$

Linear Regression with Least Square XI

最小二乘线性回归的建模和求解

Mean Squared Error (**MSE**) is the minimum goal.

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 \tag{3}$$

一个命题:

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 = \frac{1}{n}(\vec{y} - X\vec{w})^T(\vec{y} - X\vec{w})$$

Where $X = (\vec{x}_1, \ldots, \vec{x}_n)^T \in \mathbb{R}^{n\times p}$ is the design matrix for the sample data.
$\vec{y} = (y_1, \ldots, y_n)^T \in \mathbb{R}^n$ is the observed regression value vector.
$\vec{w} \in \mathbb{R}^p$ is the parameter vector we aim to optimize.

Linear Regression with Least Square XII

最小二乘线性回归的建模和求解

实际上，MSE 可以另写成：

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 = \frac{1}{n}(\vec{y} - X\vec{w})^T(\vec{y} - X\vec{w}) := \frac{1}{n}\|\vec{y} - X\vec{w}\|_2^2$$

here $||\cdot||_2$ is a $l2$ norm. [a]

[a] 此处请留意向量范数 l_2被用以表示目标优化函数。当然，这仅仅只是一个记号上的使用而已。

Linear Regression with Least Square XIII

最小二乘线性回归的建模和求解

Please notice that there is an explicit solution to this problem if X^TX is an invertible matrix[1], say:

$$\vec{w}^* = (X^TX)^{-1}X^T\vec{y}. \tag{4}$$

[Assignment] Prove formula (4).

[1] Gradient analysis would solve this problem directly, and I'd like to make it an assignment.

Linear Regression with Least Square I

最小二乘线性回归

[Answer sheet]:
To find the optimal parameters, we set the gradient of the cost function $\mathcal{J}(\vec{w})$ with respect to $\vec{w}$ to zero:

$$\begin{aligned} 0 &= \frac{\partial \mathcal{J}(\vec{w})}{\partial \vec{w}} \\ &= \frac{\partial \left(\frac{1}{n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) \right)}{\partial \vec{w}} \\ &= \frac{1}{n} \frac{\partial \left((\vec{y}^T - \vec{w}^T X^T)(\vec{y} - X\vec{w}) \right)}{\partial \vec{w}} \\ &= \frac{1}{n} \frac{\partial (\vec{y}^T\vec{y} - \vec{y}^T X\vec{w} - \vec{w}^T X^T \vec{y} + \vec{w}^T X^T X\vec{w})}{\partial \vec{w}} \\ &= ? \end{aligned} \tag{5}$$

Linear Regression with Least Square II

最小二乘线性回归

[Answer sheet]:
To find the optimal parameters, we set the gradient of the cost function $\mathcal{J}(\vec{w})$ with respect to $\vec{w}$ to zero:

$$\begin{aligned} 0 &= \frac{\partial \mathcal{J}(\vec{w})}{\partial \vec{w}} \\ &= \frac{\partial \left(\frac{1}{n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) \right)}{\partial \vec{w}} \\ &= \frac{1}{n} \frac{\partial \left((\vec{y}^T - \vec{w}^T X^T)(\vec{y} - X\vec{w}) \right)}{\partial \vec{w}} \\ &= \frac{1}{n} \frac{\partial (\vec{y}^T\vec{y} - \vec{y}^T X\vec{w} - \vec{w}^T X^T \vec{y} + \vec{w}^T X^T X\vec{w})}{\partial \vec{w}} \\ &= \frac{1}{n} (0 - X^T\vec{y} - X^T\vec{y} + 2X^T X\vec{w}) \\ &= \frac{2}{n} (-X^T\vec{y} + X^T X\vec{w}) \end{aligned} \tag{6}$$

Let the gradient equal to zero, we have

$$\hat{w} = (X^T X)^{-1} X^T \vec{y}. \tag{7}$$

Linear Regression with Least Square I

“房价预测”案例

“房价预测”案例。在这个案例中，$n = 4$（4 套房子的数据），$p = 2$（特征为：面积、到地铁站的距离）。

1. 原始数据 (Obs)

假设我们收集了 4 套公寓的数据：$\{ \vec{x}_i, y_i \}$, $i = 1, 2, 3, 4$

样本 1: 面积 $50m^2$, 距离 1km $\rightarrow$ 售价 200 万

样本 2: 面积 $60m^2$, 距离 2km $\rightarrow$ 售价 230 万

样本 3: 面积 $80m^2$, 距离 1km $\rightarrow$ 售价 310 万

样本 4: 面积 $90m^2$, 距离 5km $\rightarrow$ 售价 300 万

Linear Regression with Least Square II

“房价预测”案例

“房价预测”案例。在这个案例中，$n = 4$（4 套房子的数据），$p = 2$（特征为：面积、到地铁站的距离）。

1. 原始数据 (Obs)

假设我们收集了 4 套公寓的数据：$\{ \vec{x}_i, y_i \}$, $i = 1, 2, 3, 4$

样本 1: 面积 $50m^2$, 距离 1km $\rightarrow$ 售价 200 万

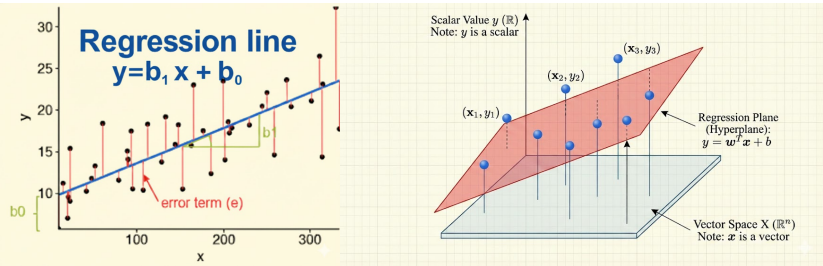
样本 2: 面积 $60m^2$, 距离 2km $\rightarrow$ 售价 230 万

样本 3: 面积 $80m^2$, 距离 1km $\rightarrow$ 售价 310 万

样本 4: 面积 $90m^2$, 距离 5km $\rightarrow$ 售价 300 万

Linear Regression with Least Square III

“房价预测”案例



2. 构建矩阵与向量

我们将数据转化为矩阵 X 和向量 $\vec{y}$ 形式：

$$X = \begin{pmatrix} 50 & 1 \\ 60 & 2 \\ 80 & 1 \\ 90 & 5 \end{pmatrix}, \quad \vec{y} = \begin{pmatrix} 200 \\ 230 \\ 310 \\ 300 \end{pmatrix}$$

其中 $\vec{w} = (w_{area}, w_{dist})^T$ 是我们要学习的回归函数 $f(\vec{x}, \vec{w}) = \vec{w}^T \vec{x}$ 的权重。

Linear Regression with Least Square IV

“房价预测”案例

3. 计算步骤
第一步：计算 X^TX (信息聚合)
这是一个 2×2 的矩阵，代表了特征之间的协方差关系：

$$X^TX = \begin{pmatrix} 50 & 60 & 80 & 90 \\ 1 & 2 & 1 & 5 \end{pmatrix} \begin{pmatrix} 50 & 1 \\ 60 & 2 \\ 80 & 1 \\ 90 & 5 \end{pmatrix} = \begin{pmatrix} 20600 & 700 \\ 700 & 31 \end{pmatrix}$$

第二步：计算 $X^T\vec{y}$ (目标投影)

$$X^T\vec{y} = \begin{pmatrix} 50 & 60 & 80 & 90 \\ 1 & 2 & 1 & 5 \end{pmatrix} \begin{pmatrix} 200 \\ 230 \\ 310 \\ 300 \end{pmatrix} = \begin{pmatrix} 75600 \\ 2470 \end{pmatrix}$$

Linear Regression with Least Square V

“房价预测”案例

第三步：求解 $\vec{w}^* = (X^TX)^{-1}X^T\vec{y}$
通过计算逆矩阵（或使用高斯消元法），我们可以得到：

$$\begin{pmatrix} w^*_{area} \\ w^*_{dist} \end{pmatrix} \approx \begin{pmatrix} 4.12 \\ -13.35 \end{pmatrix}$$

所以，最终的回归函数为

$$f(\vec{x}, \vec{w}^*) = \vec{w}^{*T}\vec{x} = w^*_{area}x_1 + w^*_{dist}x_2.$$

Linear Regression with Least Square VI

“房价预测”案例

4. 结果的物理意义 (Physical Interpretation)
通过这个具体的计算，有如下观察结果：
$w^*_{area} \approx 4.12$: 意味着面积每增加 $1m^2$，房价平均上涨约 4.12 万。
$w^*_{dist} \approx -13.35$: 意味着距离地铁每增加 1km，房价平均下降约 13.35 万（负相关）。

Linear Regression with Least Square VII

“房价预测”案例

另外，残差向量：$\vec{e} = \vec{y} - X\vec{w}^*$。
检查结果，发现 $\vec{e}$ 并不为零，这说明模型无法完美拟合所有点，但这是在 L_2 意义下的最优折中方案。

Linear Regression with Least Square I

算法总结和思考

关于这个结果的几个总结和思考：

- 首先，$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$是一个显示解，该模型的求解过程中适度使用了一些矩阵微分的技巧。

Linear Regression with Least Square II

算法总结和思考

关于这个结果的几个注解：

- 首先，$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$是一个显示解，该模型的求解过程中适度使用了一些矩阵微分的技巧。
- 其次，这个模型及其求解所对应的“机器学习”思想在哪里？

Linear Regression with Least Square III

算法总结和思考

关于这个结果的几个注解：

- 首先，$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$是一个显示解，该模型的求解过程中适度使用了一些矩阵微分的技巧。
- 其次，这个模型及其求解所对应的“机器学习”思想在哪里？
- 但是，这个解并不完美。为什么？tips: if $p > n$...

Linear Regression with Least Square IV

算法总结和思考

关于这个结果的几个注解：

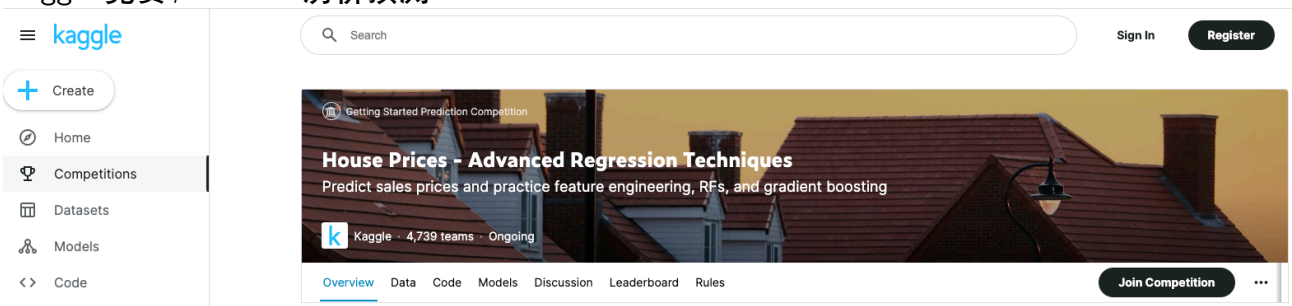
- 首先，$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$是一个显示解，该模型的求解过程中适度使用了一些矩阵微分的技巧。
- 其次，这个模型及其求解所对应的“机器学习”思想在哪里？
- 但是，这个解并不完美。为什么？tips: if $p > n$...
- 一个细节：针对 $\mathcal{J}(w)$的梯度下降是如何进行的，公式是？

Linear Regression with Least Square V

算法总结和思考

算法竞赛和更多的思考

Kaggle 竞赛，Boston 房价预测 [a] [b].



[a] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[b] 百度搜索。

Linear Regression with Least Square VI

算法总结和思考

算法比拼，考虑：

- 运算结果在测试集上的准确性以及可迁移性（鲁棒性），
- 时间复杂度和算法效率 (寻优效率)，
- 结果的深层分析（可解释性）.

Linear Regression with Least Square I

Gradient Descent！当上述方法失效的时候

Gradient descent (aka., steepest descent) is for minimizing multidimensional smooth convex objective functions of the form $\mathcal{J}: \mathbb{R}^p \to \mathbb{R}$.

定理 (Gradient Descent)

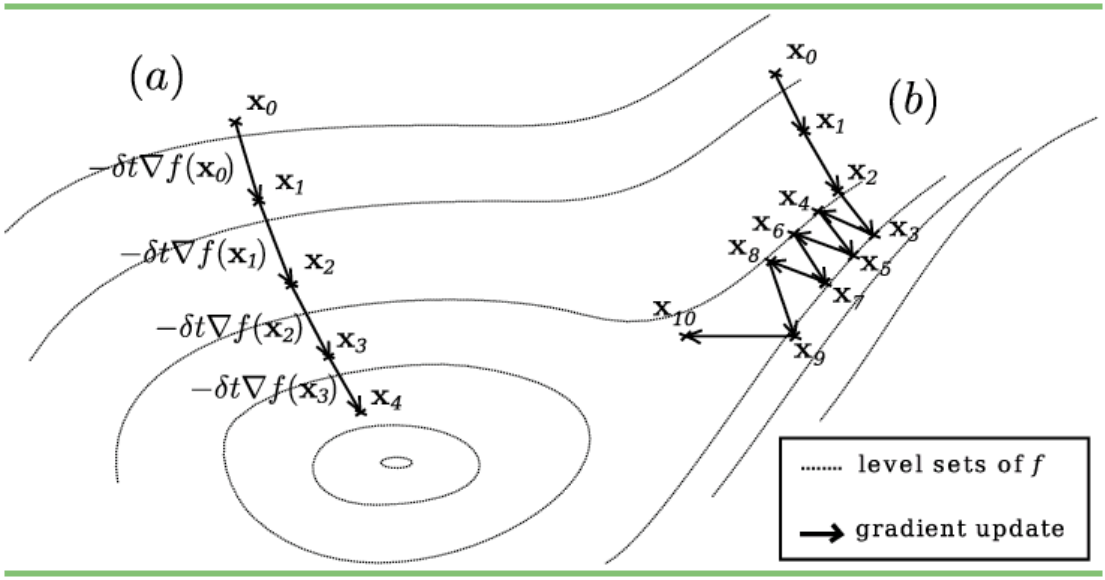
1: ***Input:*** *Initial point* $\vec{w}_0$*, gradient norm tolerance* ε
2: *Set* $t = 0$
3: ***while*** $\|\nabla \mathcal{J}(\vec{w}_t)\| \geq \varepsilon$ ***do***
4: $\quad \vec{w}_{t+1} = \vec{w}_t - \eta_t \nabla \mathcal{J}(\vec{w}_t)$
5: $\quad t = t + 1$
6: ***end while***
7: ***Return:*** $\vec{w}_t$

这里最关键的迭代步骤是:

$$\vec{w}_{t+1} = \vec{w}_t - \eta_t \nabla \mathcal{J}(\vec{w}_t) \tag{8}$$

Linear Regression with Least Square II

Gradient Descent！当上述方法失效的时候



Linear Regression with Least Square III

Gradient Descent！当上述方法失效的时候

问：当 $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$ 无法计算，具体该如何使用梯度下降？

Linear Regression with Least Square IV

Gradient Descent！当上述方法失效的时候

问：在使用 Pytorch 编写神经网络的时候，哪些部分是与梯度下降有关？

Linear Regression with Least Square V

Gradient Descent！当上述方法失效的时候

问：在使用 Pytorch 编写神经网络的时候，哪些部分是与梯度下降有关？

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import matplotlib.pyplot as plt
```


Linear Regression with Least Square VI

Gradient Descent！当上述方法失效的时候

问：在使用 Pytorch 编写神经网络的时候，哪些部分是与梯度下降有关？

```
    optimizer = optim.SGD(
        model.parameters(),
        lr=0.01,
        momentum=0.9,
        weight_decay=1e-4
    )
```

Outline

1. Linear Regression with Least Square （最小二乘线性回归） 4
2. 增添了正则项后的 Linear Regression 变种— Ridge and LASSO 40
   - Ridge regression/岭回归——多出的一个正则项 41
   - 欧几里得范数 (L2 范数) 45
   - LASSO 回归——挑选向量范数 L1，更换正则项 49
   - L1 范数（曼哈顿距离） 52
3. LASSO Regression 和 Python 代码示例 56
4. 一般情形下的线性模型，从回归到分类 61

增添了正则项后的 Linear Regression 变种— Ridge and LASSO I

Ridge regression/岭回归——多出的一个正则项

回忆最小二乘线性回归的目标函数

The square sum of error terms is the minimum goal.

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 \tag{9}$$

增添了正则项后的 Linear Regression 变种— Ridge and LASSO II

Ridge regression/岭回归——多出的一个正则项

回忆最小二乘线性回归的目标函数

The square sum of error terms is the minimum goal.

$$\mathcal{J}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 \tag{10}$$

考虑最小二乘线性回归的变种，**ridge regression/岭回归**，它引入一个$L2$ regularizer (正则项):

$$\mathcal{J}_{Ridge}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 + \lambda||\vec{w}||_2^2. \tag{11}$$

增添了正则项后的 Linear Regression 变种— Ridge and LASSO III

Ridge regression/岭回归——多出的一个正则项

几个问题：

增添了正则项后，对比最小二乘下的线性回归 w估计结论，对这个结论的影响在哪里？

对 $\mathcal{J}(w)$的梯度下降的训练影响在哪里？
这是可以通过列式求解获知的。请尝试。

增添了正则项后的 Linear Regression 变种— Ridge and LASSO IV

Ridge regression/岭回归——多出的一个正则项

考虑最小二乘线性回归的变种，**ridge regression/岭回归**，它引入一个$L2$ regularizer (正则项)：

$$\mathcal{J}_{Ridge}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2 + \lambda||\vec{w}||_2^2. \text{ —Ridge regression.} \tag{12}$$

a b

[a]凸规划的结论告诉我们，增加了正则项后，以上 Loss 函数的最优化问题等同于

$$\min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\vec{x}_i, \vec{w}))^2, \qquad s.t., ||\vec{w}||_2 < C. \tag{13}$$

此处 C与 Lagrange 常数 λ有关.

[b]增加的正则项使得解搜索需要在 $||\vec{w}||_2 < C$取值范围的约束下进行. 这减小了可行解的搜索空间。

增添了正则项后的 Linear Regression 变种— Ridge and LASSO I

欧几里得范数 (L2 范数)

欧几里得范数 (L2 范数) 定义：

对于一个 p 维向量 $\vec{w} = (w_1, w_2, \ldots, w_p)^T \in \mathbb{R}^p$，其欧几里得范数定义为：

$$\|\vec{w}\|_2 = \sqrt{\sum_{j=1}^{p} w_j^2} = \sqrt{w_1^2 + w_2^2 + \cdots + w_p^2}$$

使用矩阵表示法，它可以简洁地写作：

$$\|\vec{w}\|_2 = \sqrt{\vec{w}^T \vec{w}}$$

增添了正则项后的 Linear Regression 变种— Ridge and LASSO II

欧几里得范数 (L2 范数)

核心特性与几何意义

几何长度:

在二维空间（$p = 2$）中，这正是勾股定理：如果向量是 $\vec{w} = (x, y)^T$，那么其长度就是 $\sqrt{x^2 + y^2}$。

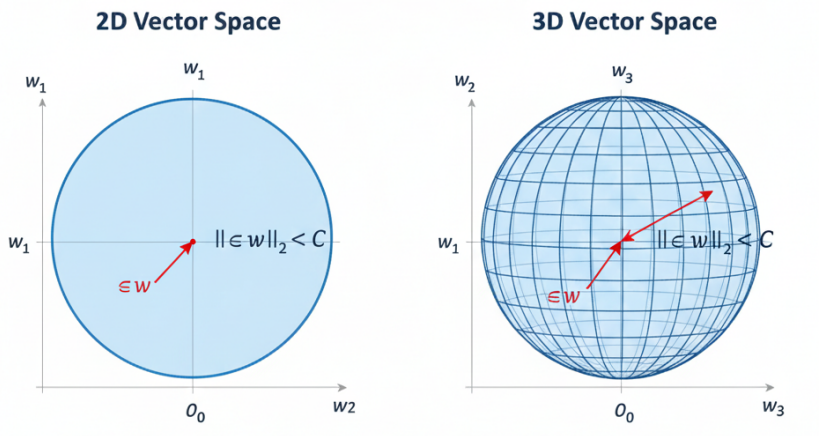
增添了正则项后的 Linear Regression 变种— Ridge and LASSO III

欧几里得范数 (L2 范数)

范数球 (Norm Ball): $\|\vec{w}\|_2 \leq C$:

在二维情况下，所有满足条件的点构成一个实心圆盘。

在三维情况下，所有满足条件的点构成一个实心球体。



增添了正则项后的 Linear Regression 变种— Ridge and LASSO IV

欧几里得范数 (L2 范数)

在机器学习中的应用（岭回归）:
在线性回归中，我们经常加入这个范数的平方作为惩罚项，称为正则项（Regularization）:

$$\mathcal{J}_{Ridge}(\vec{w}) = \mathcal{J}(\vec{w}) + \lambda \|\vec{w}\|_2^2$$

这被称为 Ridge Regression（岭回归）。它的作用是限制参数 $\vec{w}$ 的大小，防止模型为了拟合噪声而产生过大的权重，从而缓解过拟合。同时能够收缩寻优空间。

增添了正则项后的 Linear Regression 变种— Ridge and LASSO I

LASSO 回归——挑选向量范数 L1，更换正则项

Question

在很多问题求解的情况下，我们希望线性回归能获得一个包含很多 0 分量的 w，以达到“Sparsity”。为什么会这样？

怎么理解一个具有 Sparsity 特性的 w？例如在 Figure **??**中？

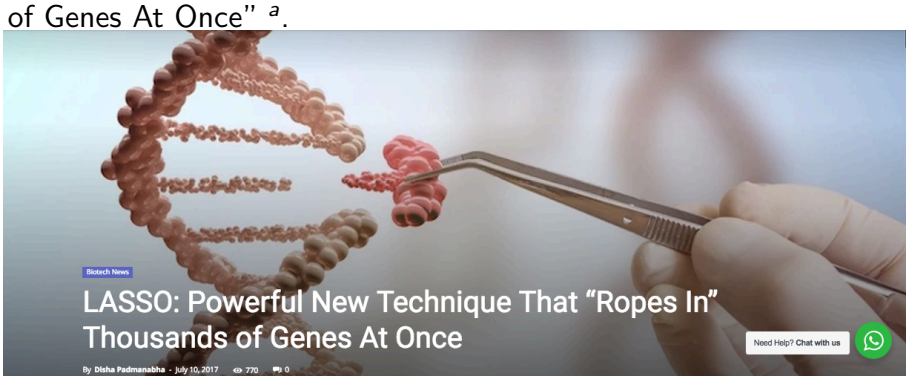
——服务于可解释性。

增添了正则项后的 Linear Regression 变种— Ridge and LASSO II

LASSO 回归——挑选向量范数 L1，更换正则项

Why do we like to have sparsity in $\vec{w}$? Here is an example:

Associate genotypes to a given phenotype. Ref "LASSO: Powerful New Technique That 'Ropes In' Thousands of Genes At Once" [a].



[a] https://www.biotecnika.org/2017/07/lasso-powerful-new-technique-that-ropes-in-thousands-of-genes-at-once/

增添了正则项后的 Linear Regression 变种— Ridge and LASSO III

LASSO 回归——挑选向量范数 L1，更换正则项

LASSO: Least absolute shrinkage and selection operator.
利用 L_1范数，我们获得 **Lasso Regression** 的目标函数。

$$\mathcal{J}_{Lasso}(\vec{w}) = \mathcal{J}(\vec{w}) + \lambda||\vec{w}||_1^2. \tag{14}$$

a b c

[a]Equivalently, from a view of convex optimization, the minimization of the above loss function suffices to:

$$\min_w \frac{1}{n}||y - Xw||_2^2, \qquad s.t., ||w||_1 < C. \tag{15}$$

Here, C is a constant, related to λ.

[b]请对照 Ridge 回归，LASSO 的唯一区别在于正则项中的范数选择。不同的范数选择，为什么会带来 w的 Sparsity 呢？

[c]实际上，$L0$范数会带来更加直接的特征约减。选用 L_1的原因是为了计算的可能性。

增添了正则项后的 Linear Regression 变种— Ridge and LASSO I

L1 范数（曼哈顿距离）

L_1 范数（L_1 Norm），也称为曼哈顿距离（Manhattan Distance）。它与之前讨论的 L_2 范数在几何形状和物理特性上有显著区别。

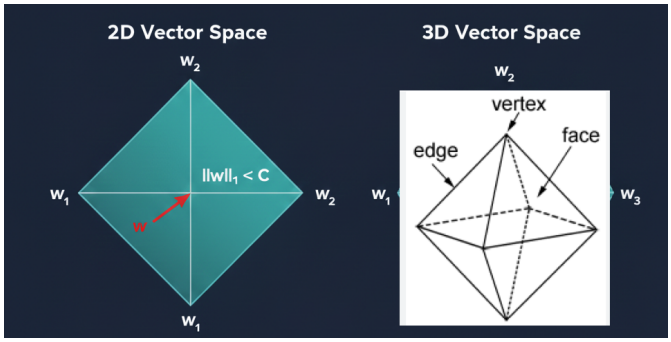
L_1 范数的数学定义：
对于一个 p 维向量 $\vec{w} = (w_1, w_2, \ldots, w_p)^T \in \mathbb{R}^p$，其 L_1 范数定义为向量各分量绝对值之和：

$$\|\vec{w}\|_1 = \sum_{j=1}^{p} |w_j| = |w_1| + |w_2| + \cdots + |w_p|$$

增添了正则项后的 Linear Regression 变种— Ridge and LASSO II

L1 范数（曼哈顿距离）

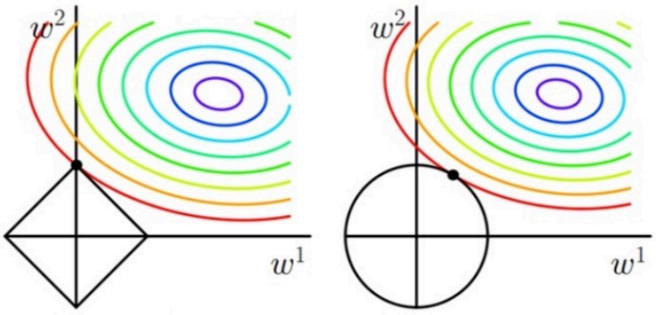
L_1 范数的关键特性：
1. 几何形状：菱形与正八面体
L_1 范数球 $\|\vec{w}\|_1 \leq C$ 的边界在不同维度下呈现出“尖锐”的特征：
二维 ($p = 2$)：是一个旋转了 45 度的正方形（菱形）。其顶点位于坐标轴上，例如 $(C, 0), (0, C), (-C, 0), (0, -C)$。
三维 ($p = 3$)：是一个正八面体（Regular Octahedron）。



增添了正则项后的 Linear Regression 变种— Ridge and LASSO III

L1 范数（曼哈顿距离）

2. 稀疏性 (Sparsity) ——这是 L_1 范数在机器学习中被广泛应用的核心原因。
直观：由于 L_1 范球的“角”正好位于坐标轴上，寻优过程中，等高线往往最先接触到这些“角”。
结果：最优解 $\vec{w}^*$ 中许多分量为 0。常被用作特征选择，从大量特征中筛选重要部分。



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes.

增添了正则项后的 Linear Regression 变种— Ridge and LASSO IV

L1 范数（曼哈顿距离）

3. 稳健性 (Robustness)
相比于 L_2 范数对误差进行平方（这会极大地放大离群点/Outliers 的影响），L_1 范数对异常值不那么敏感，因此在处理含有噪声或离群点的数据时更加稳健。

Outline

LASSO Regression 和 Python 代码示例

示例代码来自知乎[2].

例 (Command lines)

```
>git clone https://github.com/PytLab/MLBox.git
```

The raw data are:

例 (Raw data for regression)

```
1       0.455   0.365   0.095   0.514   0.2245  0.101   0.15    15
1       0.35    0.265   0.09    0.2255  0.0995  0.0485  0.07    7
-1      0.53    0.42    0.135   0.677   0.2565  0.1415  0.21    9
1       0.44    0.365   0.125   0.516   0.2155  0.114   0.155   10
0       0.33    0.255   0.08    0.205   0.0895  0.0395  0.055   7
0       0.425   0.3     0.095   0.3515  0.141   0.0775  0.12    8
-1      0.53    0.415   0.15    0.7775  0.237   0.1415  0.33    20
...
```

[2] https://zhuanlan.zhihu.com/p/30535220]

LASSO Regression 和 Python 代码示例

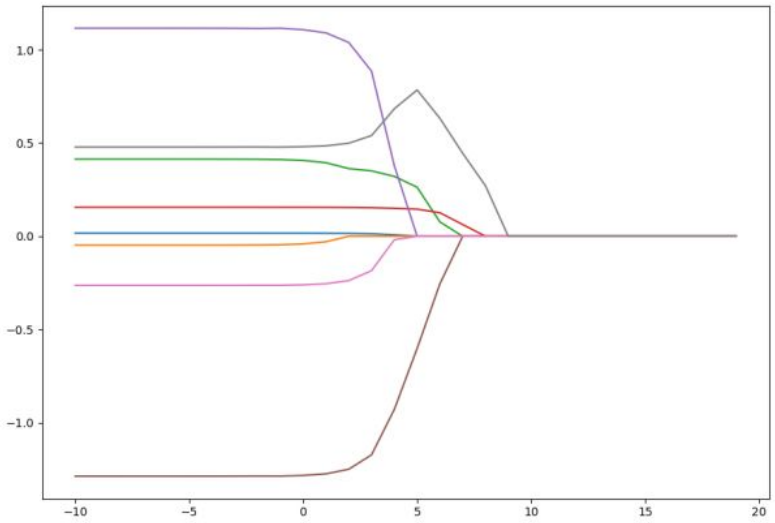
w approximate 0, while λ increases.

例 (Command lines)

```
lambda = e^(0), w = [[ 0.0164  -0.0412  0.4066   0.1553  1.1076 -1.2825  -0.2605   0.
lambda = e^(1), w = [[ 0.0161 -0.0295  0.3941  0.1550   1.0905 -1.2741  -0.2547  0.48
lambda = e^(2), w = [[ 0.0153   0.     0.3626  0.1542  1.0391 -1.2494  -0.2378  0.498
lambda = e^(3), w = [[ 0.01325  0.     0.3505  0.1528  0.8850 -1.1720  -0.1846  0.539
lambda = e^(4), w = [[ 0.0076  0.          0.3209  0.1497  0.3782 -0.9284  -0.0188  0
lambda = e^(5), w = [[ 0.   0.     0.2627  0.1453  0.         -0.6018   0.      0.784
lambda = e^(6), w = [[ 0.    0.     0.0766  0.1260  0.         -0.2552   0.    0.6322
lambda = e^(7), w = [[ 0.     0.    0.  0.0628  0.  0. 0.   0.4449]]
lambda = e^(8), w = [[ 0.   0.     0.   0.    0.  0. 0.   0.2707]]
lambda = e^(9), w = [[ 0.  0.  0.  0.  0.  0.  0.  0.]]
lambda = e^(10), w = [[ 0.  0.  0.  0.  0.  0.  0. 0.]]
lambda = e^(11), w = [[ 0.  0.  0.  0.  0.  0.  0.  0.]]
lambda = e^(12), w = [[ 0.  0.  0.  0.  0.  0.  0.  0.]]
lambda = e^(13), w = [[ 0.  0.  0.  0.  0.  0.  0.  0.]]
```

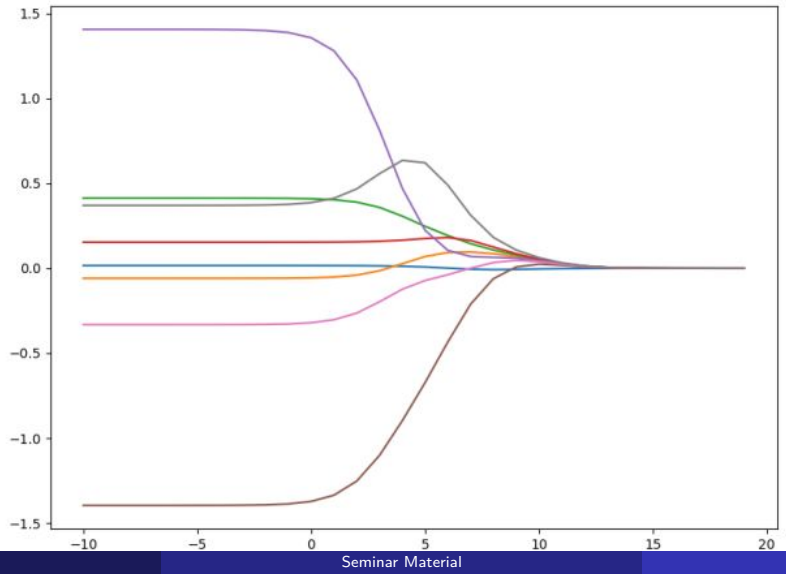

LASSO Regression 和 Python 代码示例

LASSO regression: w_i approximates zero when λ increases.



LASSO Regression 和 Python 代码示例

Ridge regression: w_i doesn't approximates zero very quickly when λ increases.



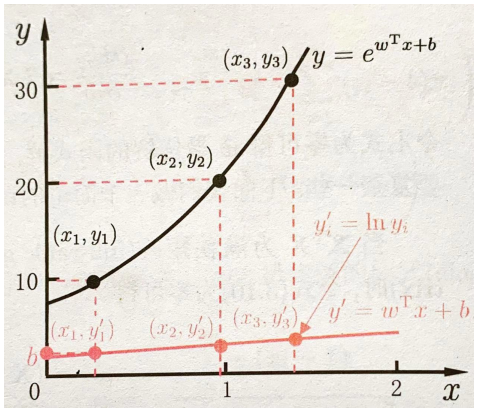
Outline

一般情形下的线性模型，从回归到分类
Log-linear regression

We've already known how to make regression by using a linear regression model. Sometimes, we would like to generalize the linear regression model and make it approximate a series of observations with non-linear values. For example,

$$\ln y = \vec{w}^T\vec{x} + b \tag{16}$$

is called "log-linear regression".



一般情形下的线性模型，从回归到分类
Generalized linear model

Generally, if we consider a monotonic differentiable function $g(\cdot)$,

$$y = g^{-1}(\vec{w}^T\vec{x} + b) \tag{17}$$

is called "generalized linear model". The funtion, $g(\cdot)$ is called "link function".

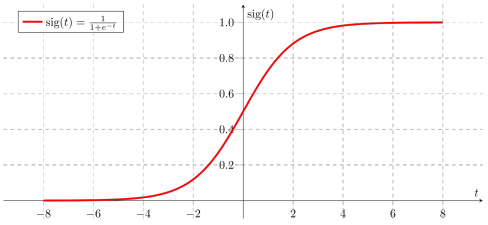
一般情形下的线性模型，从回归到分类

Unit-step function for classification

Consider a two-class classification, and the label is $y \in \{0, 1\}$, and the only attempt needed is to convert a real number $z = \vec{w}^T\vec{x} + b$ to a binary value y.
The ideal choice is "unit-step function":

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0. \end{cases} \tag{18}$$

However unit-step function is not continuous. So, "sigmoid" function replaces it. That's Logistic regression model for binary classification!



一般情形下的线性模型，从回归到分类

更多的讨论，引向 Logistic 回归。
让我们转到下一章。

Acknowledgement

课堂知识点回顾!