Linear Regression

Regression: Housing Price

Features of a House

Regressor

Predict

Price = $0.5M

Train

label

features

Price	Selling price of the property ($1,000)
Beds	Number of bedrooms in the house
Baths	Number of bathrooms in the house
Square Feet	Size of the house in square feet
Miles to Resort	Miles from the property to the downtown resort area
Miles to Base	Miles from the property to the base of the ski resort's mountain
Acres	Lot size in number of acres
Cars	Number of cars that will fit into the garage
Years Old	Age of the house, in years, at the time it was listed
DoM	Number of days the house was on the market before it sold

Linear Algebra Basics

Vector & Matrix

Vector

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^n$$

Transpose

$$\mathbf{a}^T = [a_1, a_2, \cdots, a_n]$$

Matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1d} & a_{2d} & \cdots & a_{nd} \end{bmatrix} \in \mathbb{R}^{d \times n}$$

Row form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1:} \\ \mathbf{A}_{2:} \\ \vdots \\ \mathbf{A}_{n:} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix}$$

Column form

$$\mathbf{A} = \left[\mathbf{A}_{:1}, \mathbf{A}_{:2}, \cdots, \mathbf{A}_{:d}\right] = \left[\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_d\right]$$

Vector/Matrix-Vector Product

Vector-vector inner (dot) product (Scalar)

$$\mathbf{a}^T\mathbf{b} \in \mathbb{R} = \langle \mathbf{a}, \mathbf{b} \rangle = [a_1, a_2, \cdots, a_n] \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix} = \sum_{i=1}^{n} a_i b_i$$

Vector-vector outer product (Matrix)

$$\mathbf{a}\mathbf{b}^T \in \mathbb{R}^{m\times n} = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \end{bmatrix} [b_1, b_2, \cdots, b_n] = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_n \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_m b_1 & a_m b_2 & \cdots & a_m b_n \end{bmatrix}$$

Matrix-vector product (Vector) $\mathbf{A} \in \mathbb{R}^{m\cdot n}$ $\mathbf{b} \in \mathbb{R}^n$

Row form

$$\mathbf{A}\mathbf{b} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix} \mathbf{b} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{b} \\ \mathbf{a}_2^T\mathbf{b} \\ \vdots \\ \mathbf{a}_m^T\mathbf{b} \end{bmatrix} \in \mathbb{R}^m$$

Column form

$$\mathbf{A}\mathbf{b} = \left[\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n\right] \mathbf{b} = b_1\mathbf{a}_1 + b_2\mathbf{a}_2 + \cdots + b_n\mathbf{a}_n$$

Matrix-Matrix Product (4 Ways)

$$\mathbf{A} \in \mathbb{R}^{m \times n} \quad \mathbf{B} \in \mathbb{R}^{n \times k} \quad \mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times k}$$

A. Set of vector-vector products

1. Represent A by Rows and B by Columns (Most natural)

$$\mathbf{C} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \cdots \\ \mathbf{a}_m^T \end{bmatrix} [\mathbf{b}_1, \mathbf{b}_2, \cdots \mathbf{b}_k] = \begin{bmatrix} \mathbf{a}_1^T\mathbf{b}_1 & \mathbf{a}_1^T\mathbf{b}_2 & \cdots & \mathbf{a}_1^T\mathbf{b}_k \\ \mathbf{a}_2^T\mathbf{b}_1 & \mathbf{a}_2^T\mathbf{b}_2 & \cdots & \mathbf{a}_2^T\mathbf{b}_k \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_m^T\mathbf{b}_1 & \mathbf{a}_m^T\mathbf{b}_2 & \cdots & \mathbf{a}_m^T\mathbf{b}_k \end{bmatrix}$$

(i,j)th entry of C is the inner product of ith row of A and jth column of B

2. Represent A by Columns and B by Rows

$$\mathbf{C} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n] \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \cdots \\ \mathbf{b}_n^T] \end{bmatrix} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$$

AB as the sum of outer product of the ith column of A and the jth row of B

B. Set of matrix-vector products

3. AB as multiplying rows of A and B

$$\mathbf{C} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \cdots \\ \mathbf{a}_m^T \end{bmatrix} \mathbf{B} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{B} \\ \mathbf{a}_2^T\mathbf{B} \\ \cdots \\ \mathbf{a}_m^T\mathbf{B} \end{bmatrix} \qquad \mathbf{c}_i^T = \mathbf{a}_i^T\mathbf{B}$$

4. AB as multiplying A and columns of B

$$\begin{aligned} \mathbf{C} &= \mathbf{A}[\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k] \\ &= [\mathbf{A}\mathbf{b}_1, \mathbf{A}\mathbf{b}_2, \cdots, \mathbf{A}\mathbf{b}_k] \end{aligned}$$

$$\mathbf{c}_j = \mathbf{A}\mathbf{b}_j$$

Gradient vs. Matrix/Vector

Real-valued function

$$f : \mathbb{R}^{m \times n} \to \mathbb{R}$$

Gradient of *f* (with respect to a matrix) is the matrix of partial derivatives

$$\mathbf{A} \in \mathbb{R}^{m \times n} \qquad \nabla_{\mathbf{A}} f(\mathbf{A}) = \frac{\partial f(\mathbf{A})}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial f(\mathbf{A})}{\partial A_{11}} & \frac{\partial f(\mathbf{A})}{\partial A_{12}} & \cdots & \frac{\partial f(\mathbf{A})}{\partial A_{1n}} \\ \frac{\partial f(\mathbf{A})}{\partial A_{21}} & \frac{\partial f(\mathbf{A})}{\partial A_{22}} & \cdots & \frac{\partial f(\mathbf{A})}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{A})}{\partial A_{m1}} & \frac{\partial f(\mathbf{A})}{\partial A_{m2}} & \vdots & \frac{\partial f(\mathbf{A})}{\partial A_{mn}} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\mathbf{x} \in \mathbb{R}^{n} \qquad \nabla_{\mathbf{x}} f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{n}$$

Operations and Properties

Square matrix: a matrix with the same number of rows and columns.

Symmetric: a square matrix $\mathbf{A}$ is symmetric if $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$

Full rank: a matrix is full rank if the rank equals to #rows or #columns.

Vector norm: informally a measure of the "length" of the vector

Vector Norm

ℓ_p norm: $\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$

ℓ_1 norm: $\|\mathbf{x}\|_1 = \sum_i |x_i|$

ℓ_2 norm: $\|\mathbf{x}\|_2 = \sqrt{\sum_i |x_i|^2}$ (Euclidean norm)

ℓ_∞ norm: $\|\mathbf{x}\|_\infty = \max_i |x_i|$ (max norm)

Gradient Descent

Gradient (Steepest) Descent

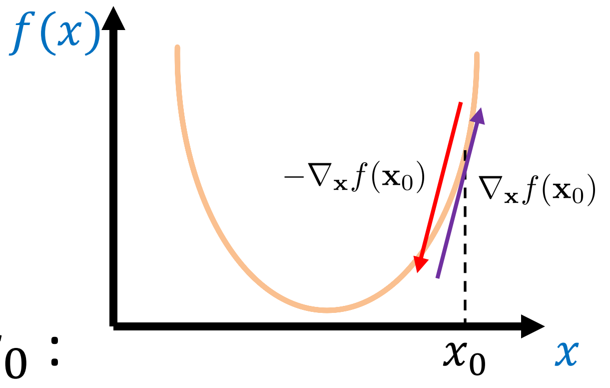
Main idea:

- Start a random point (e.g., x_o);
- Iteratively take the steepest descent direction Δx and apply $x_0 = x_0 + \alpha\, \Delta x$ (α>0)

Gradient: $\frac{\partial f}{\partial \mathbf{x}}, \nabla_{\mathbf{x}} f(\mathbf{x})$

- $\mathbf{x}$ is a d-dimensional vector.
- $f(x)$ is a scalar.
- $\frac{\partial f}{\partial \mathbf{x}} = [\frac{\partial f}{\partial x_1}; \frac{\partial f}{\partial x_2}; \cdots; \frac{\partial f}{\partial x_d}]$ is a d-dimensional vector.

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$$



Tayler expansion around x_0 :

$$f(\mathbf{x}_0 + \Delta\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla_{\mathbf{x}} f(\mathbf{x}_0)^T \Delta\mathbf{x}$$

Steepest descent direction:

$$\Delta\mathbf{x}^\star = \arg\min_{\Delta\mathbf{x}} f(\mathbf{x}_0) + \nabla_{\mathbf{x}} f(\mathbf{x}_0)^T \Delta\mathbf{x}$$

$$= \arg\min_{\Delta\mathbf{x}} \nabla_{\mathbf{x}} f(\mathbf{x}_0)^T \Delta\mathbf{x} \quad = -\frac{\nabla_{\mathbf{x}} f(\mathbf{x}_0)}{||\nabla_{\mathbf{x}} f(\mathbf{x}_0)||_2}$$

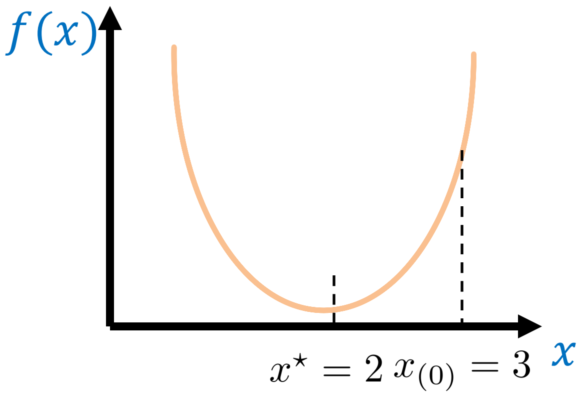
$$\text{Lemma: } \forall \mathbf{a}, \mathbf{b}, \arg\min_{\mathbf{b}:||\mathbf{b}||_2=1} \mathbf{a}^T\mathbf{b} = -\frac{\mathbf{a}}{||\mathbf{a}||_2}$$

Gradient descent

Negative direction of **a**

Gradient (Steepest) Descent

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$$



$f(x) = (x-2)^2 \quad \frac{\partial f}{\partial \mathbf{x}} = 2(x-2) \quad \alpha = 0.25$

$$\mathbf{g}_{(0)} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{(0)}=3} = 2(3-2) = 2$$
$$x_{(1)} = x_{(0)} - 0.25 g_{(0)} = 3 - 0.5 = 2.5$$

$$\mathbf{g}_{(1)} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{(1)}=2.5} = 2(2.5-2) = 1$$
$$x_{(2)} = x_{(1)} - 0.25 g_{(1)} = 2.5 - 0.25 = 2.25$$

$$\mathbf{g}_{(2)} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{(2)}=2.25} = 2(2.25-2) = 0.5$$
$$x_{(3)} = x_{(2)} - 0.25 g_{(2)} = 2.25 - 0.125 = 2.125$$

… $\Longrightarrow$ $x^\star = 2$

Gradient (Steepest) descent

- Randomly initialize $x_{(0)}$; α>0
- For t = 0 to T
  - Gradient at $x_{(t)}$: $\mathbf{g}_{(t)} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{(t)}}$
  - Update x: $\mathbf{x}_{(t+1)} = \mathbf{x}_{(t)} - \alpha \mathbf{g}_{(t)}$

Linear Regression
& Least Square

Linear Regression

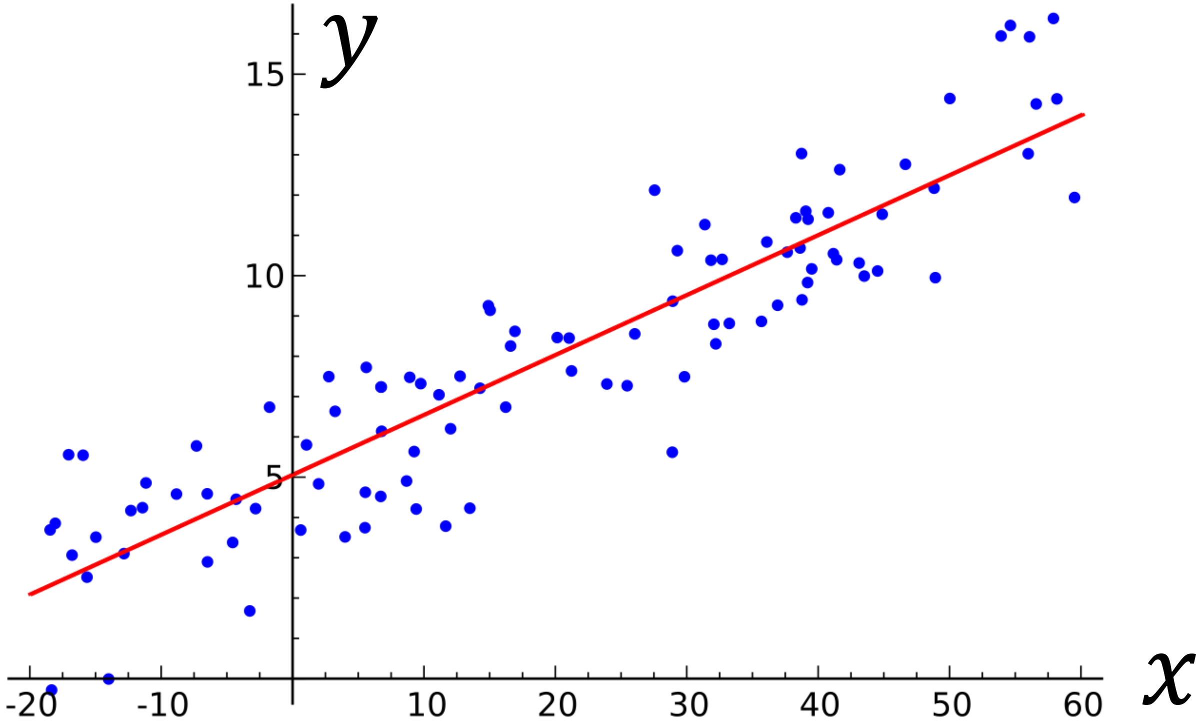
Input: data vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n, \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i, \forall i$

1-dim ($d = 1$) example:

Solution:

$$y_i \approx 0.15\, x_i + 5.0$$

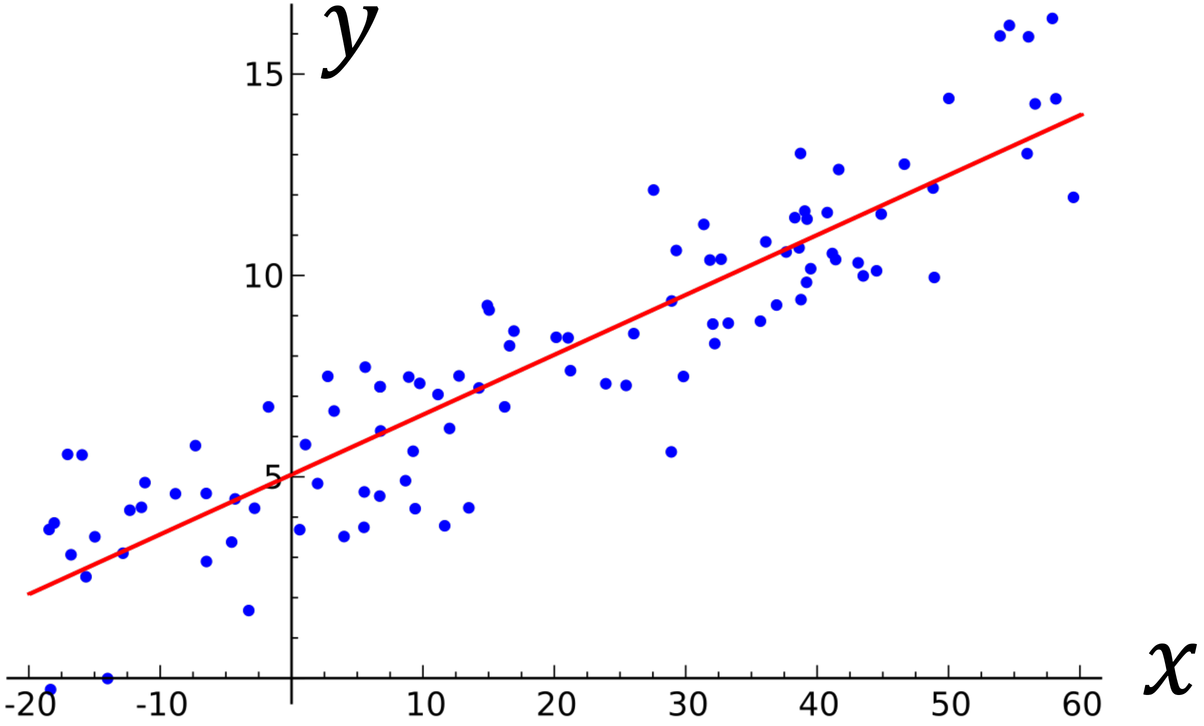


Linear Regression

Input: data vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i, \forall i$

Question: how to obtain $\mathbf{w}$ and b ?



Least Square

Input: data vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i, \forall i$

Least square loss:

$$\min_{\mathbf{w} \in \mathbb{R}^d, b} L(\mathbf{w}, b) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i \right)^2$$

Intercept (or bias)

$$\bar{\mathbf{x}}_i = [\mathbf{x}_i; 1] \quad \bar{\mathbf{w}} = [\mathbf{w}; b]$$

$$\mathbf{x}_i^T \mathbf{w} + b = \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}$$

$$\Longrightarrow \quad \min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} L(\bar{\mathbf{w}}) = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

Least Square in Matrix Form

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} L(\bar{\mathbf{w}}) = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2 \quad = \min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \|\bar{\mathbf{X}} \bar{\mathbf{w}} - \mathbf{y}\|_2^2$$

$$\bar{\mathbf{X}} = \left[\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_n \right] \in \mathbb{R}^{(d+1)\times n}$$

$$\bar{\mathbf{X}}^T = \begin{bmatrix} \bar{\mathbf{x}}_1^T \\ \bar{\mathbf{x}}_2^T \\ \cdots \\ \bar{\mathbf{x}}_n^T \end{bmatrix} \in \mathbb{R}^{n\times(d+1)}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

$$\bar{\mathbf{X}}^T \bar{\mathbf{w}} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} \\ \cdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} \end{bmatrix} \qquad \bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \cdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix}$$

$$\|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2 = \left\| \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \cdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix} \right\|_2^2 = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

Remember: $\|\mathbf{x}\|_2^2 = \sum_i |x_i|^2 = \sum_i x_i^2$

Analytical Solution: Normal Equations

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} L(\bar{\mathbf{w}}) = \|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2$$

$$\|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2 = \left(\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\right)^T \left(\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\right) = \bar{\mathbf{w}}^T \bar{\mathbf{X}} \bar{\mathbf{X}}^T \bar{\mathbf{w}} - 2\bar{\mathbf{w}}^T \bar{\mathbf{X}} \mathbf{y} + \mathbf{y}^T \mathbf{y}$$

First-order optimality: $\dfrac{\partial \|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2}{\partial \bar{\mathbf{w}}} = \mathbf{0} \Longrightarrow 2\bar{\mathbf{X}}\bar{\mathbf{X}}^T \bar{\mathbf{w}} - 2\bar{\mathbf{X}}\mathbf{y} = \mathbf{0}$

Normal equation: $\bar{\mathbf{X}}\bar{\mathbf{X}}^T \bar{\mathbf{w}} = \bar{\mathbf{X}}\mathbf{y} \Longrightarrow \bar{\mathbf{w}}^\star = \left(\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right)^{-1} \bar{\mathbf{X}}\mathbf{y}$

(1) $\bar{\mathbf{X}}\bar{\mathbf{X}}^T$ full rank $\quad \bar{\mathbf{X}} \in \mathbb{R}^{(d+1)\times n} \quad \bar{\mathbf{X}}\bar{\mathbf{X}}^T \in \mathbb{R}^{(d+1)\times(d+1)} \Longrightarrow n > d$

(2) $\left(\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right)^{-1}$ $\quad$ Complexity: $O(d^3)$ $\quad$ Computationally intensive

Approximate Solution: Gradient Descent

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} L(\bar{\mathbf{w}}) = \|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2$$

Gradient on all data:

$$\frac{\partial \|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2}{\partial \bar{\mathbf{w}}} = 2\bar{\mathbf{X}}\left(\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\right)$$

Gradient descent repeats:

1. Compute gradient: $\mathbf{g}_t = 2\bar{\mathbf{X}}\left(\bar{\mathbf{X}}^T \bar{\mathbf{w}}_t - \mathbf{y}\right)$
2. Update parameters: $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \alpha \mathbf{g}_t$

Convergence: after K iterations if satisfies

$$L(\bar{\mathbf{w}}_K) - L(\bar{\mathbf{w}}^*) \leq \frac{1}{2\alpha K} \|\bar{\mathbf{w}}_0 - \bar{\mathbf{w}}^*\|_2^2$$

Complexity per iteration: $O(nd)$

$\bar{\mathbf{X}}^T \in \mathbb{R}^{n \times (d+1)}$

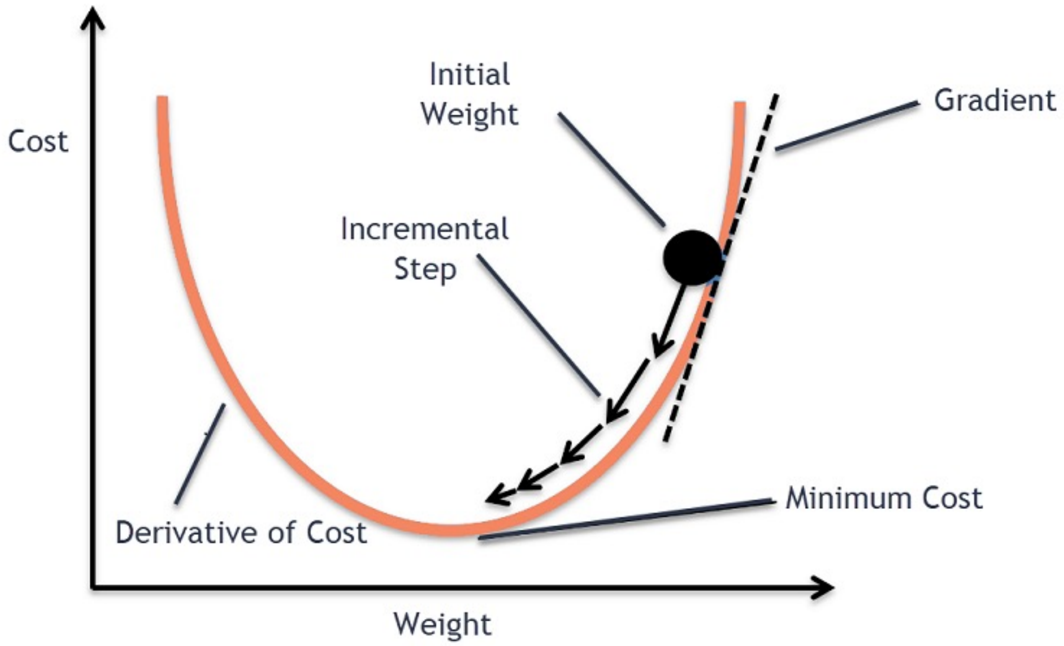
$\mathbf{a} = \bar{\mathbf{X}}^T \bar{\mathbf{w}}_t : n(d+1)$

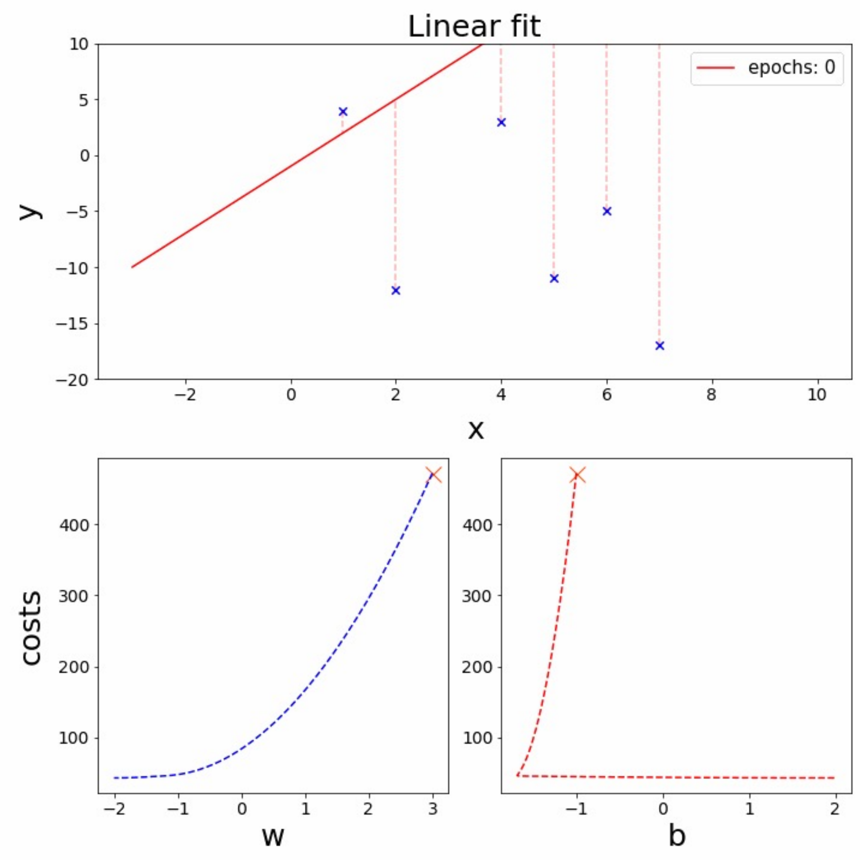
$\mathbf{b} = \bar{\mathbf{X}} \mathbf{a} : n(d+1)$

$\mathbf{c} = \bar{\mathbf{X}} \mathbf{y} : n(d+1)$

Total complexity: $O(Knd)$

Visualization





Approximate Solution: Stochastic GD

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \|\bar{\mathbf{X}}^T \bar{\mathbf{w}} - \mathbf{y}\|_2^2 = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i\right)^2$$

Gradient per data sample: $\frac{\partial (\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i)_2^2}{\partial \bar{\mathbf{w}}} = 2\bar{\mathbf{x}}_i \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i\right)$

Stochastic gradient descent repeats:

For i=1 to n,

1. Compute gradient: $\mathbf{g}_t^i = 2\bar{\mathbf{x}}_i \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}}_t - y_i\right)$
2. Update parameters: $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \alpha_t \mathbf{g}_t^i$

Complexity per iteration/sample: $O(d)$

Intuitive Interpretation:

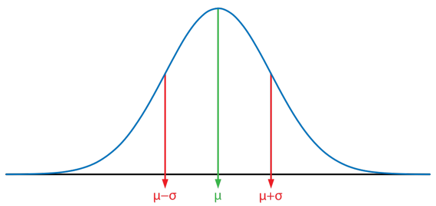
- Magnitude of the update is proportional to the error term $\bar{\mathbf{x}}_i^T \bar{\mathbf{w}}_t - y_i$
- If a training example has a prediction nearly matches the actual label, there is little need to change the parameters

Empirically, stochastic GD gets close to the minimum value much faster than (batch) GD, in particular when the training data is large!

Probabilistic Interpretation

$$y_i = \bar{\mathbf{x}}_i^T \bar{\mathbf{w}} + \epsilon_i, \forall i \in \{1, \cdots, n\}$$

IID: $\epsilon_i \sim \mathcal{N}(0, \sigma^2), \forall i$



$$p(y_i | \bar{\mathbf{x}}_i; \bar{\mathbf{w}}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{\left(y_i - \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}\right)^2}{2\sigma^2} \right) \Longleftarrow p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{\epsilon_i^2}{2\sigma^2} \right)$$

Given the feature matrix X of all samples xi and w, what is the distribution of the labels y?

$$p(\bar{\mathbf{y}} | \bar{\mathbf{X}}; \bar{\mathbf{w}}) \quad := \mathcal{L}(\bar{\mathbf{w}})$$

Independent assumption ϵ_i: $\mathcal{L}(\bar{\mathbf{w}}) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i; \bar{\mathbf{w}}) \; = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{\left(y_i - \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}\right)^2}{2\sigma^2} \right)$

Maximum (log)-likelihood: choose w so as to make the data as high probability as possible.

$$\max \log \mathcal{L}(\bar{\mathbf{w}}) \quad \Longrightarrow \quad \min \sum_{i=1}^{n} \left(y_i - \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}\right)^2$$

Minimizing the least-square loss equals to maximizing the (log)-likelihood under Gaussian noise assum.

Summary

Least square loss $$L(\bar{\mathbf{w}}) = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i\right)^2 = \|\bar{\mathbf{X}}\bar{\mathbf{w}} - \mathbf{y}\|_2^2$$

Unconstrained optimization $$\bar{\mathbf{w}}^\star = \arg\min_{\bar{\mathbf{w}}} L(\bar{\mathbf{w}})$$

Probabilistic interpretation

- Minimizing least square ⇔ maximum likelihood (Gaussian noise)

Solutions

- Analytical: Normal equation
- Approximate: (Stochastic) gradient/steepest descent

Python example (housing price estimation)

Regularized Linear Regression

$\bar{\mathbf{X}}\bar{\mathbf{X}}^T$ full rank

What if it is not full rank?

L2 regularization

Ridge regression

Linear regression has a dense solution

How to obtain a spare solution?

LASSO

Solve Least Squares in Python (Housing Price Prediction)

1. Load Training/Test Data

```
from keras.datasets import boston_housing


(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

print('shape of x_train: ' + str(x_train.shape))
print('shape of x_test: ' + str(x_test.shape))
print('shape of y_train: ' + str(y_train.shape))
print('shape of y_test: ' + str(y_test.shape))
```

```
shape of x_train: (404, 13)
shape of x_test: (102, 13)
shape of y_train: (404,)
shape of y_test: (102,)
```

2. Add all “1” Feature

```
import numpy

n, d = x_train.shape
xbar_train = numpy.concatenate((x_train, numpy.ones((n,
1))),axis=1)

print('shape of x_train: ' + str(x_train.shape))
print('shape of xbar_train: ' + str(xbar_train.shape))
```

```
shape of x_train: (404, 13)
shape of xbar_train: (404, 14)
```

3. Solve the Least Square

Analytical solution: $\bar{\mathbf{w}}^{\star} = \boxed{(\bar{\mathbf{X}}^T \bar{\mathbf{X}})}^{-1} \bar{\mathbf{X}}^T \mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Square

Analytical solution: $\bar{\mathbf{w}}^{\star} = \boxed{(\bar{\mathbf{x}}^T\bar{\mathbf{x}})^{-1}}\bar{\mathbf{x}}^T\mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Square

Analytical solution: $\bar{\mathbf{w}}^{\star} = \boxed{\left(\bar{\mathbf{X}}^T \bar{\mathbf{X}}\right)^{-1}} \boxed{\bar{\mathbf{X}}^T \mathbf{y}}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Square

Training data Mean Squared Error (MSE): $\frac{1}{n}\|\bar{\mathbf{X}}\bar{\mathbf{w}}^{\star} - \mathbf{y}\|_2^2$

```
y_lsr = numpy.dot(xbar_train, w)
diff = y_lsr - y_train
mse = numpy.mean(diff * diff)
print('Train MSE: ' + str(mse))
```

```
Train MSE: 22.004800838348142
```

Linear Regression for Housing Price

Features of a House x_t

Extend it to $\bar{\mathbf{x}}_t = [\mathbf{x}_t; 1]$

Linear Regressor

$\bar{\mathbf{w}}^\star$

Train

Predict

Price:

$(\bar{\mathbf{w}}^\star)^T \bar{\mathbf{x}}_t$= \$500K

label, y_i

features, x_i

Price	Selling price of the property ($1,000)
Beds	Number of bedrooms in the house
Baths	Number of bathrooms in the house
Square Feet	Size of the house in square feet
Miles to Resort	Miles from the property to the downtown resort area
Miles to Base	Miles from the property to the base of the ski resort's mountain
Acres	Lot size in number of acres
Cars	Number of cars that will fit into the garage
Years Old	Age of the house, in years, at the time it was listed
DoM	Number of days the house was on the market before it sold

4. Make Prediction for Test Samples

- Add a "1" feature to test feature matrix: $\mathbf{X}_t \rightarrow \bar{\mathbf{X}}_t$
- Make prediction by: $\mathbf{y}_t^{pred} = \bar{\mathbf{X}}_t \bar{\mathbf{w}}^\star$
- MSE on testing data: $\frac{1}{n_t} \|\mathbf{y}_t^{pred} - \mathbf{y}_t\|_2^2$

```
n_test, _ = x_test.shape
xbar_test = numpy.concatenate((x_test, numpy.ones((n_test, 1))), axis=1)
y_pred = numpy.dot(xbar_test, w)
```

```
# mean squared error (testing)
diff = y_pred - y_test
mse = numpy.mean(diff * diff)
print('Test MSE: ' + str(mse))

Test MSE: 23.195599256409857
```

Training MSE is **22.0**

4. Make Prediction for Test Samples

Test MSE is **23.2**

Also known as out-of-sample error.

The gap is an indicator of generalization.

Training MSE is **22.0**

Also known as in-sample error.

Acknowledgement

Some slides are from **Shusen Wang**
https://github.com/wangshusen/DeepLearning