Machine Learning & Signals Learning

3 Linear Least-Squares

Goals: re-write LS in the matrix notation and extend it to the multi-variate.

מטרה: רישום ע"י וקטורים/מטריצות

3.1 Uni-variate LS

מטרה: המרה של חיזוי לינארי תוך רישום מטריצי

To improve the mathematical representation, vector notation can be used. This time, the points $\{x_k, y_k\}_{k=1}^{M}$ are organized into vectors, with a few additional ones, as follows,

מספר עמודות M×1 מספר שורות — וקטור של אחדות — משקלים — w = weight משקל

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \mathbf{1}_M = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^M, \ \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{1}$$

The resulting model notation is

$w_0 + w_1 x_1 = \hat{y}_1$
$w_0 + w_1 x_2 = y_2$
$\vdots$

וקטור של חיזוי ←

$$\hat{\mathbf{y}} = f(\mathbf{X}; \mathbf{w}) = \mathbf{1}_M \underset{\text{קבוע}}{w_0} + \mathbf{x} w_1 = \mathbf{X}\mathbf{w}, \tag{2}$$

מטריצה של משנים

where $\mathbf{X} = [\mathbf{1}_M \quad \mathbf{x}] \in \mathbb{R}^{M \times 2}$ and $\mathbf{w} = [w_0 \quad w_1]^T$.

The model error is

וקטור שגיאות

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} \tag{3}$$

$\|a\| = \sqrt{a_0^2 + a_1^2 + \ldots a_n^2}$

The corresponding SSE loss function (2.4) is

$$\begin{aligned} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) &= \mathbf{e}^T \mathbf{e} && = \|\mathbf{e}\|^2 \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) && = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) && = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \end{aligned} \tag{4}$$

The corresponding optimal minimum (Eq. (2.7)) results from the solution of normal equation (in matrix form)

$$\nabla_{\mathbf{w}} \mathcal{L}(\cdot) = -\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \tag{5}$$

and is given by

$$\begin{gathered} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w} = 0 \\ \mathbf{X}^T \mathbf{y} = (\mathbf{X}^T \mathbf{X}) \mathbf{w} \end{gathered}$$

Finally, vector notation normal equation solution is

$$\boxed{\mathbf{w}_{opt} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}} \tag{6}$$

Example 3.1: For centered ($\bar{x} = \bar{y} = 0$) variables with $w_0 = 0$,

$$w_1 = \frac{\sum_{i=1}^{M} x_i y_i}{\sum_{i=1}^{M} x_i^2}$$

3.2 Multivariate LS

מטרה: יותר מעמודה אחת של X

3.2.1 Notation

דוגמה: דירה: שטח מספר חדרים קומה

For the multivariate N-dimensional formulation,

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{M,1} & x_{M,2} & \cdots & x_{M,N} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_M^T \end{bmatrix} = \underbrace{[\mathbf{1}_M \quad \tilde{\mathbf{x}}_1 \quad \tilde{\mathbf{x}}_2 \quad \cdots \quad \tilde{\mathbf{x}}_N]}_{X} \in \mathbb{R}^{M\times(N+1)} \quad (7$$

$$\mathbf{w} = [w_0 \quad w_1 \quad \cdots \quad w_N]^T \in \mathbb{R}^{N+1} \quad (8$$

where:

שורה = [illegible]

- each row $\mathbf{x}_i^T$ is termed *sample*, $\mathbf{x}_i = [1, x_{i,1}, \ldots, x_{i,N}]^T \in \mathbb{R}^{N+1}$, $i = 1, \ldots, M$.
- each column $\tilde{\mathbf{x}}_j \in \mathbb{R}^M$, $j = 1, \ldots, N$ is termed *feature vector*. $\longrightarrow$ עמודה

The normal equation solution (3.6) is the same for $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, independently from the number of feature vectors.

Dataset

All the data rows in $(\mathbf{X}, \mathbf{y})$ are called dataset with M samples (or entries) and N features.

3.2.2 Adjusted Coefficient of Determination

Goal: Provide a comparison metric for multivariate linear models that accounts for different numbers or subsets of features.

While Section 2.3.1 discusses the standard R^2 metric, the regular coefficient of determination has a critical flaw in multivariate regression: it mathematically cannot decrease when new features are added to the model, even if those features are entirely uninformative.

To address this, the *adjusted* R^2 ($\bar{R}^2$) is introduced. It uses a penalty term based on the number of features N relative to the number of samples M. It is given by:

$$\bar{R}^2 = 1 - \left(1 - R^2\right)\frac{M-1}{M-N-1} \tag{9}$$

The primary advantage of $\bar{R}^2$ is that it penalizes the model for adding irrelevant variables. It will only increase if a new feature improves the model's predictive power more than would be expected by chance.

3.3 Properties

3.3.1 Moore–Penrose inverse (pseudo-inverse)

The Moore–Penrose inverse is the generalization of the ordinary matrix inverse for non-rectangular matrices. Assuming $\mathbf{X}$ has full column rank, its left pseudo-inverse is given by:

$$\mathbf{X}^{+} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \in \mathbb{R}^{(N+1)\times M}, \tag{10}$$

such that multiplying it by $\mathbf{X}$ yields the identity matrix

$$\mathbf{X}^{+}\mathbf{X} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X} = \mathbf{I}_{N+1}$$

Note that a direct implementation of $\mathbf{X}^{+}$ may have numerical stability issues due to inversion of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$. All the modern programming languages have numerically-stable and efficient implementation of pseudo-inverse calculations.

Using this notation, the optimal weight vector in (3.6) can be compactly written as

$$\mathbf{w}_{opt} = \mathbf{X}^{+}\mathbf{y} \tag{11}$$

3.3.2 Projection matrix

Using the pseudo-inverse notation, the model's output (the predicted values) can be written directly in terms of the target vector $\mathbf{y}$:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{X}\mathbf{X}^{+}\mathbf{y} = \mathbf{P}\mathbf{y}\end{aligned} \tag{12}$$

where

$$\mathbf{P} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \in \mathbb{R}^{M\times M} \tag{13}$$

is termed the *projection matrix*. Geometrically, $\mathbf{P}$ projects the target vector $\mathbf{y}$ onto the subspace spanned by the columns of $\mathbf{X}$ (the column space).

Important properties of the projection matrix $\mathbf{P}$ include:

- Symmetry, $\mathbf{P} = \mathbf{P}^T$,
- Idempotence, $\mathbf{P} = \mathbf{P}^2$,

Proof.

$$\begin{aligned}\mathbf{P}^2 &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \\ &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T = \mathbf{P}\end{aligned} \tag{14}$$

• Orthogonality: The projection $\mathbf{P}$ is orthogonal to its complement $(\mathbf{I}-\mathbf{P})$, i.e. $\mathbf{P} \perp (\mathbf{I}-\mathbf{P})$.

Proof. $\mathbf{P}(\mathbf{I}-\mathbf{P}) = \mathbf{P} - \mathbf{P}^2 = \mathbf{0}$.

• Complementary projection, $\mathbf{I}-\mathbf{P}$ is also projection matrix.

• The residual error vector $\mathbf{e}$ is simply the projection of $\mathbf{y}$ onto the orthogonal complement of the column space of $\mathbf{X}$

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{P}\mathbf{y} = (\mathbf{I}-\mathbf{P})\mathbf{y}. \tag{15}$$

3.3.3 Error properties

Error and feature orthogonality

שגיאה $\perp$ נתונים

אין קשר לינארי

$$\mathbf{e} \perp \mathbf{X} \Rightarrow \mathbf{X}^T\mathbf{e} = \mathbf{0}_{N+1} \tag{16}$$

Proof. Substituting the projection matrix notation:

$$\begin{aligned}\mathbf{X}^T\mathbf{e} &= \mathbf{X}^T\left(\mathbf{y} - \hat{\mathbf{y}}\right) \\ &= \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{X}^T\mathbf{y} - \mathbf{I}_{N+1}\mathbf{X}^T\mathbf{y} = \mathbf{0}_{N+1}\end{aligned}$$

Error and prediction orthogonality

שגיאה $\perp$ חיזוי

$$\mathbf{e} \perp \hat{\mathbf{y}} \Rightarrow \hat{\mathbf{y}}^T\mathbf{e} = \mathbf{e}^T\hat{\mathbf{y}} = 0 \tag{17}$$

Proof. Using the properties of the projection matrix $\mathbf{P}$ (symmetry and idempotence):

$$\begin{aligned}\hat{\mathbf{y}}^T\mathbf{e} &= (\mathbf{P}\mathbf{y})^T(\mathbf{I}-\mathbf{P})\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}^T(\mathbf{I}-\mathbf{P})\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}^2\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}\mathbf{y} = 0\end{aligned}$$

The interesting outcome of this property is a relation between error and prediction,

$$\underbrace{\|\mathbf{y}\|^2}_{\text{נתונים}} = \underbrace{\|\hat{\mathbf{y}}\|^2}_{\text{חיזוי}} + \underbrace{\|\mathbf{e}\|^2}_{\text{שגיאה}} \tag{18}$$

מה שהצלחתי "להסביר" f(x) לא אקראי שהנחתי בתוך העולם

אקראי

Proof.

$$\begin{aligned}
\|\mathbf{y}\|^2 &= \|\hat{\mathbf{y}} + \mathbf{e}\|^2 \\
&= (\hat{\mathbf{y}} + \mathbf{e})^T(\hat{\mathbf{y}} + \mathbf{e}) \\
&= \hat{\mathbf{y}}^T\hat{\mathbf{y}} + 2\hat{\mathbf{y}}^T\mathbf{e} + \mathbf{e}^T\mathbf{e} \\
&= \|\hat{\mathbf{y}}\|^2 + 0 + \|\mathbf{e}\|^2 \qquad (\text{since } \hat{\mathbf{y}}^T\mathbf{e} = 0)
\end{aligned}$$

Average error

If the model includes a bias/intercept term (w_0), the mean of the residuals is exactly zero,

ממוצע של שגיאות = 0
(ללא ריבוע)

$$\bar{\mathbf{e}} = \frac{1}{M}\sum_{k=1}^{M} e_k = \frac{1}{M}\mathbf{1}^T\mathbf{e} = 0 \tag{19}$$

Proof. From Eq. (3.16), we know $\mathbf{X}^T\mathbf{e} = \mathbf{0}_{N+1}$. Because the first column of $\mathbf{X}$ is $\mathbf{1}_M$ (the bias feature),

$$\mathbf{X}^T\mathbf{e} = \begin{bmatrix} \mathbf{1}_M^T \\ \tilde{\mathbf{x}}_1^T \\ \vdots \\ \tilde{\mathbf{x}}_N^T \end{bmatrix}\mathbf{e} = \begin{bmatrix} \mathbf{1}_M^T\mathbf{e} \\ \vdots \end{bmatrix} = \mathbf{0}_{N+1} \implies \mathbf{1}_M^T\mathbf{e} = 0$$

An interesting consequence of this is that the mean of the true values equals the mean of the predicted values

$$\begin{aligned}
\bar{\mathbf{y}} &= \bar{\hat{\mathbf{y}}} \\
&= w_0 + w_1\bar{\tilde{\mathbf{x}}}_1 + \cdots + w_N\bar{\tilde{\mathbf{x}}}_N
\end{aligned} \tag{20}$$

Proof.

$$\begin{aligned}
\bar{\mathbf{y}} &= \overline{\hat{\mathbf{y}} + \mathbf{e}} \\
&= \bar{\hat{\mathbf{y}}} + \bar{\mathbf{e}}
\end{aligned} \tag{21}$$

Following this property, $\bar{\mathbf{y}} = 0$ implies $w_0 = 0$ and $\bar{\mathbf{e}} = \mathbf{0}$.

Error distribution

The values of the error vector $\mathbf{e}$ are assumed to be normally distributed, due to Central Limit Theorem (CLT). Typically, this assumption is not needed in ML, but it is important for statistical analysis for small values of M.

Minimum SSE

The reduced expression for the resulting minimal loss is

מינימום שגיאה המתקבל

$$\mathcal{L}_{min} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} \tag{22}$$

Proof.

$$\begin{aligned} sse_{min} &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T\mathbf{e} \\ &= \mathbf{y}^T\mathbf{e} - \hat{\mathbf{y}}^T\mathbf{e} \qquad \leftarrow (17) \\ &= \mathbf{y}^T\mathbf{e} \\ &= \mathbf{y}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned} \tag{23}$$

The MSE or RMSE evaluation from the loss is straightforward by dividing by M and taking the square root, respectively.

סיכום:

3.3.4 Implementation note

The matrix $\mathbf{X}$ is assumed *full-rank*, i.e. columns are linearly independent and the corresponding eigenvalues of $\mathbf{X}^T\mathbf{X}$ are sufficiently large. The straightforward approach is to use numerically optimized algorithms for $\mathbf{w}_{opt}$ solution given $\mathbf{X}$ and $\mathbf{y}$, such as:

1. Matlab: `lsqminnorm`
2. Python: `numpy.linalg.lstsq` and `scipy.linalg.lstsq`

Code example Matlab example.

3.4 Gradient Descent

Goal: Find the minimum of the function:

- First-order derivative based algorithm.
- Local minimum only.

Preface

Let's assume some function $y = f(x)$, with $x, y \in \mathfrak{R}$, differentiable with $\dfrac{dy}{dx} = f'(x)$.

The values of the error vector $\mathbf{e}$ are assumed to be normally distributed, due to Central Limit Theorem (CLT). Typically, this assumption is not needed in ML, but it is important for statistical analysis for small values of M.

Minimum SSE

The reduced expression for the resulting minimal loss is

$$\mathcal{L}_{min} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} \tag{22}$$

Proof.

$$\begin{aligned} sse_{min} &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T\mathbf{e} \\ &= \mathbf{y}^T\mathbf{e} - \hat{\mathbf{y}}^T\mathbf{e} \qquad \leftarrow (17) \\ &= \mathbf{y}^T\mathbf{e} \\ &= \mathbf{y}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned} \tag{23}$$

The MSE or RMSE evaluation from the loss is straightforward by dividing by M and taking the square root, respectively.

3.3.4 Implementation note

The matrix $\mathbf{X}$ is assumed *full-rank*, i.e. columns are linearly independent and the corresponding eigenvalues of $\mathbf{X}^T\mathbf{X}$ are sufficiently large. The straightforward approach is to use numerically optimized algorithms for $\mathbf{w}_{opt}$ solution given $\mathbf{X}$ and $\mathbf{y}$, such as:

1. Matlab: `lsqminnorm`
2. Python: `numpy.linalg.lstsq` and `scipy.linalg.lstsq`

Code example Matlab example.

סיכומי:-

* ניתן להניח $\underbrace{\text{קשר לינארי}}_{\text{קו ישר}}$ בין הנתונים שונים
* חישוב של מקדמים הקו הישר - ניתן לעשות ע"י נוסחא/חישוב מספרי יעיל
* ניתן לקשר בין הנתונים לבין ערך y הנדרש

3.4 Gradient Descent

Goal: Find the minimum of the function:

- First-order derivative based algorithm.
- Local minimum only.

Preface

Let's assume some function $y = f(x)$, with $x, y \in \mathfrak{R}$, differentiable with $\dfrac{dy}{dx} = f'(x)$.

The values of the error vector $\mathbf{e}$ are assumed to be normally distributed, due to Central Limit Theorem (CLT). Typically, this assumption is not needed in ML, but it is important for statistical analysis for small values of M.

Minimum SSE

The reduced expression for the resulting minimal loss is

$$\mathcal{L}_{min} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} \tag{22}$$

Proof.

$$\begin{aligned} sse_{min} &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T\mathbf{e} \\ &= \mathbf{y}^T\mathbf{e} - \hat{\mathbf{y}}^T\mathbf{e} \qquad \leftarrow (17) \\ &= \mathbf{y}^T\mathbf{e} \\ &= \mathbf{y}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned} \tag{23}$$

The MSE or RMSE evaluation from the loss is straightforward by dividing by M and taking the square root, respectively.

3.3.4 Implementation note

The matrix $\mathbf{X}$ is assumed *full-rank*, i.e. columns are linearly independent and the corresponding eigenvalues of $\mathbf{X}^T\mathbf{X}$ are sufficiently large. The straightforward approach is to use numerically optimized algorithms for $\mathbf{w}_{opt}$ solution given $\mathbf{X}$ and $\mathbf{y}$, such as:

1. Matlab: `lsqminnorm`
2. Python: `numpy.linalg.lstsq` and `scipy.linalg.lstsq`

Code example Matlab example.

3.4 Gradient Descent

Goal: Find the minimum of the function:

- First-order derivative based algorithm.
- Local minimum only.

Preface

Let's assume some function $y = f(x)$, with $x, y \in \mathfrak{R}$, differentiable with $\dfrac{dy}{dx} = f'(x)$.

- The interpretation of $f'(x)$ is the slope of $f(x)$ at a point x.
- By the definition of the derivative, for some small ϵ,

$$f(x+\epsilon) \approx f(x) + \epsilon f'(x) \tag{3.24a}$$
$$f(x-\epsilon) \approx f(x) - \epsilon f'(x) \tag{3.24b}$$

- Given the sign of the derivative,

$$\begin{aligned} f(x-\epsilon) &< f(x), & f'(x) &> 0 \\ f(x+\epsilon) &< f(x), & f'(x) &< 0 \end{aligned} \tag{25}$$

- For sufficiently small ϵ,

$$f\left(x - \epsilon \,\mathrm{sign}\left(f'(x)\right)\right) \leq f(x) \tag{26}$$

The idea of the algorithm is to reduce $f(x)$ toward the minimum by moving in the direction opposite to the sign of the derivative $f'(x)$ according to (3.26).

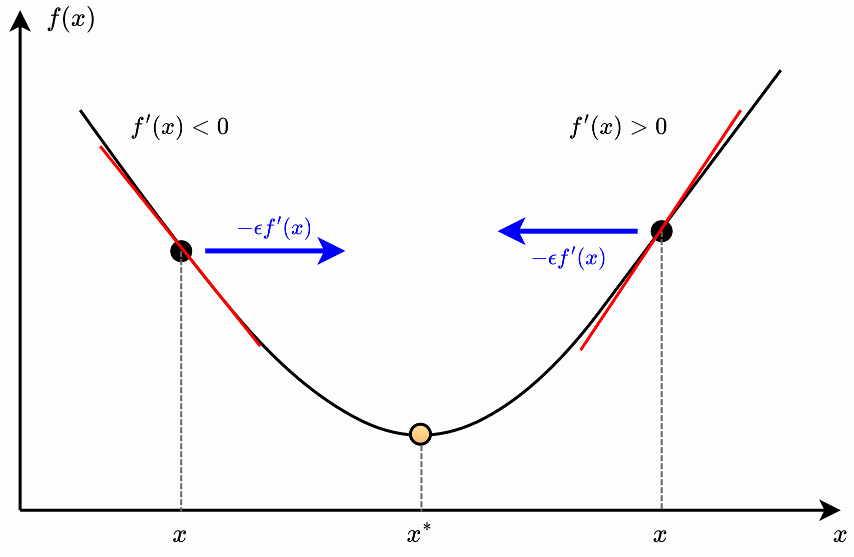

Figure 3.1: Gradient descent intuition: moving in the direction $-\epsilon f'(x)$ reduces the function value toward the minimum.

3.4.1 Definition

Gradient descent (GD) - scalar function: For differentiable function $f(x)$, the iterative algorithm

$$x_{n+1} = x_n - \alpha f'(x_n) \tag{27}$$

converges to some local minimum of $f(x)$.

Required parameters are:

- Step-size $\alpha > 0$ is some positive constant or some function of n, i.e. α_n.
- x_0 is an initial guess.

Some of the most common stopping conditions are:

- Reaching the point of slow convergence, $|x_{n+1} - x_n| < \epsilon$.
- Limiting the number of iterations, $n \leq n_0$.

Gradient descent (GD) - vector function: For differentiable multivariate and multidimensional function $f(\mathbf{x}) : \Re^N \to \Re$, the iterative algorithm is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla_{\mathbf{x}} f(\mathbf{x}_n). \tag{28}$$

Each dimension is iteratively reduced according to its derivative.

Notes:

- Easy to implement.
- Requires analytical or numerical derivative.
- Non-trivial selection of the optimal value of α. Inappropriate selection of α results in slower convergence (or divergence in extreme cases). Typically, $\boldsymbol{\alpha}_n$ vector with non-equal values may be desirable.
- Useful only for the function with single (global) minimum, such as MSE minimization.

Minimization of loss function Optimal values of $\mathbf{w}$ may be found by substitution of the gradient of the loss function into (vector) GD,

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{29}$$

GD for LS

Optimal values of $\mathbf{w}$ may be found by substitution of the gradient (Eq. (3.5)) into (3.29),

$$\begin{aligned}
\mathbf{w}_{n+1} &= \mathbf{w}_n - \alpha \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \\
&= \mathbf{w}_n - \frac{\alpha}{M} \mathbf{X}^T \left(\mathbf{X} \mathbf{w}_n - \mathbf{y} \right) \\
&= \mathbf{w}_n \left(\mathbf{I} - \frac{\alpha}{M} \mathbf{X}^T \mathbf{X} \right) + \frac{\alpha}{M} \mathbf{X}^T \mathbf{y}
\end{aligned} \tag{30}$$

The main differences between the normal equation (3.6) and GD solution are:

- Lower computational complexity, since it does not require inversion of $\mathbf{X}^T\mathbf{X}$.
- Option for reduced memory calculation with batch gradient descent.

3.4.2 Full-batch, mini-batch and stochastic GD

The entire dataset is $\{(\mathbf{x}_k, y_k)\}_{k=1}^{M}$. Recall that for LS we can write the loss as a sum over samples,

$$\mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \sum_{k=1}^{M} \ell_k(\mathbf{w}; \mathbf{x}_k, y_k), \tag{31}$$

where $\ell_k(\cdot)$ is a single raw loss between y_k and $\hat{y}_k$.

For the SSE loss,

$$\ell_k(\mathbf{w}) = \left(y_k - \mathbf{x}_k^T \mathbf{w}\right)^2. \tag{32}$$

Full-batch

Full-batch GD uses the gradient of the *whole* loss elements, M. In this case, for LS, the loss gradient takes the form of (3.30).

Mini-batch GD

Instead of using all M samples, we select (usually at random) at iteration n a subset (mini-batch) of indices

$$\mathcal{B}_n \subset \{1, 2, \ldots, M\}, \qquad |\mathcal{B}_n| = B \ll M.$$

We define the corresponding sub-matrices containing rows indexed by $\mathcal{B}_n$,

$$\mathbf{X}_{\mathcal{B}_n} \in \mathfrak{R}^{B\times(N+1)}, \qquad \mathbf{y}_{\mathcal{B}_n} \in \mathfrak{R}^{B}, \tag{33}$$

The mini-batch loss is

$$\mathcal{L}_{\mathcal{B}_n}(\mathbf{w}) = \frac{1}{B} \sum_{k\in\mathcal{B}_n} \ell_k(\mathbf{w}) = \frac{1}{B} \left\|\mathbf{y}_{\mathcal{B}_n} - \mathbf{X}_{\mathcal{B}_n}\mathbf{w}\right\|^2 \tag{34}$$

and its gradient is

$$\nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{B}_n}(\mathbf{w}) = \frac{1}{B} \mathbf{X}_{\mathcal{B}_n}^T \left(\mathbf{X}_{\mathcal{B}_n}\mathbf{w} - \mathbf{y}_{\mathcal{B}_n}\right). \tag{35}$$

The mini-batch GD update becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \frac{\alpha}{B} \mathbf{X}_{\mathcal{B}_n}^T \left(\mathbf{X}_{\mathcal{B}_n}\mathbf{w}_n - \mathbf{y}_{\mathcal{B}_n}\right), \tag{36}$$

where B is called the *batch size*.

Stochastic gradient descent (SGD) The special extreme case of $B = 1$ is termed *stochastic gradient descent*.

Epoch

The term *epoch* denotes one full pass over the entire dataset. In full-batch GD, one iteration already uses all M samples, so one iteration is equivalent to one epoch. In mini-batch or stochastic GD, an epoch consists of several iterations, each using only a (small) part of the data.

Assume that we sweep once over all M samples without repetition:

- Full-batch GD: $B = M \Rightarrow L = 1$ update per epoch.
- Mini-batch GD: B samples per update results in

  $$L = \left\lceil \frac{M}{B} \right\rceil \tag{37}$$

  batches (updates) per epoch.

- SGD: $B = 1 \Rightarrow L = M$ updates per epoch.

Theoretical relation to the full gradient

Theoretically, mini-batches $\mathcal{B}_n$ are estimators of the full gradient,

$$\mathbb{E}\left[\nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{B}_n}(\mathbf{w})\right] = \frac{1}{L} \sum_{n=1}^{L} \nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{B}_n}(\mathbf{w}) = \frac{1}{M} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}). \tag{38}$$

and follow (on average) the same direction as full-batch GD. Practically, since $\mathbf{w}$ is updated for each n, it results with an additional *noise* as illustrated in Fig. 3.2.

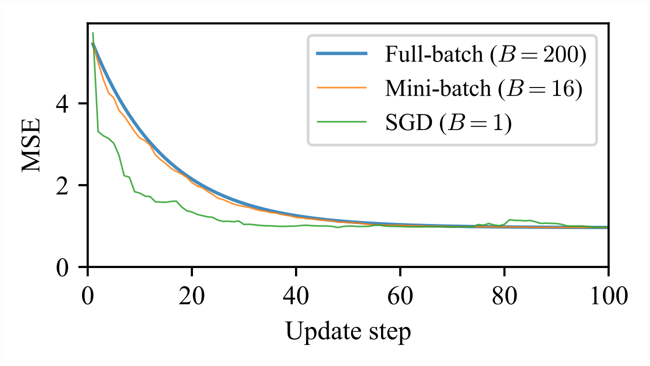

Figure 3.2: Mini-batch gradient noise on a toy LS problem ($y = 2x + 1 + \epsilon$, $M = 100$, $\alpha = 0.04$): full-batch GD (B=M) yields a smooth loss trajectory, while mini-batch (B=16) and SGD (B=1) produce increasingly noisy updates.

Summary

- Mini-batch / SGD can handle huge datasets, since each update uses only B samples in memory.
- Updates are faster (less data per step), and typically allow efficient parallel computation (e.g., on GPUs).
- The noisy gradient may slow convergence and requires careful choice of step-size α (often smaller than in full-batch GD).
- In practice, mini-batch GD (with B between a few tens and a few hundreds) is the most commonly used compromise between speed and stability.