DDA3020 Machine Learning
Lecture 05 Linear Regression

Baoyuan Wu
School of Data Science, CUHK-SZ

Jan 24/29/31, 2024

Outline

Notations

- A **set** is an unordered collection of unique elements.
- We denote a set as a calligraphic capital character, for example, $\mathcal{S}$.

- A set of numbers can be finite (include a fixed amount of values). In this case, it is denoted using accolades, for example, $\{1, 3, 18, 23, 235\}$ or $\{x_1, x_2, x_3, ..., x_d\}$.
- A set can also be infinite.

Notations

- A set can be infinite and include all values in some interval.
- If a set includes all values between a and b, including a and b, it is denoted using brackets as $[a, b]$.
- If the set does not include the values a and b, such a set is denoted using parentheses like this: (a, b).
- For example, the set [0, 1] includes such values as 0, 0.0001, 0.25, 0.784, 0.9995, and 1.0.
- A special set denoted $\mathcal{R}$ (or R, $\mathbb{R}$) includes all real numbers from minus infinity to plus infinity.

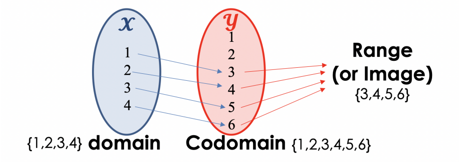
Notations

- Intersection of two sets:
  $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cap \mathcal{S}_2$
  Example: $\{1, 3, 5, 8\} \cap \{1, 8, 4\} = \{1, 8\}$

- Union of two sets:
  $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$
  Example: $\{1, 3, 5, 8\} \cup \{1, 8, 4\} = \{1, 3, 4, 5, 8\}$

Functions, derivative and gradient

- A **function** is a relation that associates each element x of a set $\mathcal{X}$, the **domain** of the function, to a single element y of another set $\mathcal{Y}$, the **codomain** of the function.
- A function usually has a name. If the function is called f, this relation is denoted $y = f(x)$ (read f of x), the element x is the argument or input of the function, and y is the value of the function or the output.
- The symbol that is used for representing the input is the variable of the function (we often say that f is a function of the variable x).



Functions, derivative and gradient

- A **scalar function** can also have vector argument such as, $y = f(\mathbf{x})$, or a scalar argument ($y = f(x)$).
- A **vector function**, denoted as $\mathbf{y} = \mathbf{f}(\mathbf{x})$, is a function that returns $\mathbf{y}$, which can have either a vector argument ($\mathbf{y} = \mathbf{f}(\mathbf{x})$) or a scalar argument ($\mathbf{y} = \mathbf{f}(x)$).

Notation

- The notation $f : \mathbb{R}^d \to \mathbb{R}$ means that f is a function that maps real d-vectors to real numbers, *i.e.*, it is a scalar-valued function of d dimension vectors.
- If $\mathbf{x}$ is a d-vector, then $f(\mathbf{x})$, which is a scalar, denotes the value of the function f at $\mathbf{x}$. In the notation $f(\mathbf{x})$, $\mathbf{x}$ is referred to as the argument of the function

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d)$$

Linear and Affine functions

- To describe a function $f : \mathbb{R}^d \to \mathbb{R}$ we have to specify what its value is for any possible argument $\mathbf{x} \in \mathbb{R}^d$.
- For example, we can define a function $f : \mathbb{R}^4 \to \mathbb{R}$ by

$$f(\mathbf{x}) = x_1 + x_2 - 2x_3 - x_4$$

Linear and Affine functions

Another example: The inner product function

- Suppose there is a d-vector. We can define a scalar valued function f of d-vectors, given by

$$f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} = a_1 x_1 + a_2 x_2 + \ldots + a_d x_d$$

  for any d-vector $\mathbf{x}$.
- This function gives the inner product of its d-vector argument $\mathbf{x}$ with some (fixed) d-vector $\mathbf{a}$.
- We can also think of f as forming a weighted sum of the elements of $\mathbf{x}$; the elements of $\mathbf{a}$ give the weights used in forming the weighted sum.

Linear and Affine functions

- A function $f : \mathbb{R}^d \to \mathbb{R}$ is **linear** if it satisfies the following two properties:
  - **Homogeneity**: For any d-vector $\mathbf{x}$ and a scalar α, $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$.
  - **Additivity**: For any d-vectors $\mathbf{x}$ and $\mathbf{y}$, $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$

$$f(\mathbf{x}) = \mathbf{a}^\top\mathbf{x} = a_1x_1 + a_2x_2 + \ldots + a_dx_d$$

  for any d-vector $\mathbf{x}$.
- **Homogeneity** states that scaling the (vector) argument is the same as scaling the function value.
- **Additivity** says that adding (vector) arguments is the same as adding the function values.

Linear and Affine functions

Superposition and linearity

- The inner product function f defined before satisfies the linearity property

$$\begin{aligned} f(\alpha\mathbf{x} + \beta\mathbf{y}) &= \mathbf{a}^\top(\alpha\mathbf{x} + \beta\mathbf{y}) \\ &= \mathbf{a}^\top(\alpha\mathbf{x}) + \mathbf{a}^\top(\beta\mathbf{y}) \\ &= \alpha(\mathbf{a}^\top\mathbf{x}) + \beta(\mathbf{a}^\top\mathbf{y}) \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{y}) \end{aligned}$$

  for all d-vectors $\mathbf{x}, \mathbf{y}$, and all scalars α, β.
- This property is called **superposition** (which consists of homogeneity and additivity).
- A **function** that satisfies the superposition property is called **linear**

Linear and Affine functions

- If a function f is linear, superposition extends to linear combinations of any number of vectors:

$$f(\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k) = \alpha_1 f(\mathbf{x}_1) + \ldots + \alpha_k f(\mathbf{x}_k)$$

for any d-vectors $\mathbf{x}_1$, ..., $\mathbf{x}_k$ and any scalars α_1,...,α_k

Functions, derivative and gradient

Linear and Affine functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ is affine if and only if it can be expressed as $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + b$ for some d-vector $\mathbf{a}$ and a scalar b, which is sometimes called the offset.

Example:

$$f(\mathbf{x}) = 2.3 - 2x_1 + 1.3x_2 - x_3$$

is affine, with $b = 2.3$, $\mathbf{a} = \begin{bmatrix} -2 \\ 1.3 \\ -1 \end{bmatrix}$.

Functions, derivative and gradient

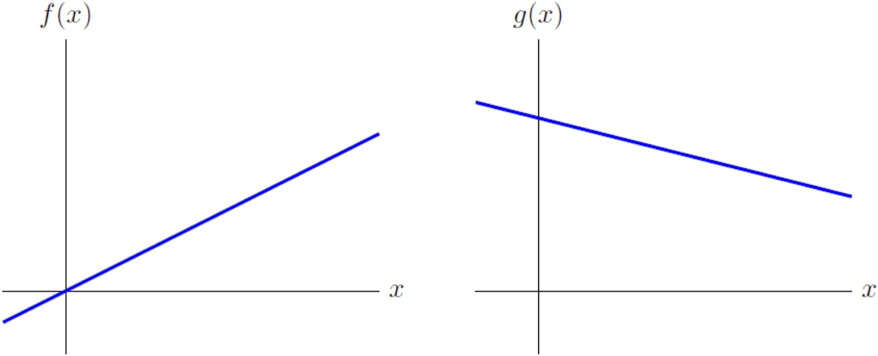


Figure 2.1 *Left.* The function f is linear. *Right.* The function g is affine, but not linear.

Functions, derivative and gradient

- We say that $f(x)$ has a **local minimum** at $x = c$ if $f(x) \geq f(c)$ for every x in some open interval around $x = c$.
- An **interval** is a set of real numbers with the property that any number that lies between two numbers in the set is also included in the set.
- An **open interval** does not include its endpoints and is denoted using parentheses. For example, $(0, 1)$ means "all numbers greater than 0 and less than 1"
- The minimal value among all the **local minima** is called the **global minima**. See illustration in the Figure in next page.

Functions, derivative and gradient

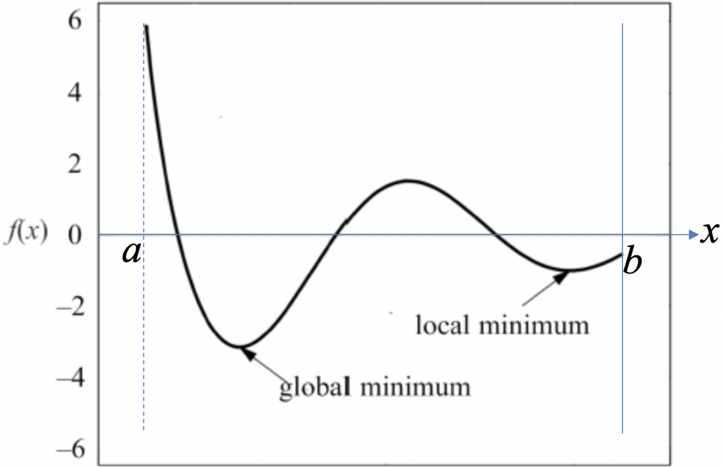


Figure: Local and global minima of a function. $a < x < b$

$\max$ **vs** $\arg\max$

- Given a set of values $\mathcal{A} = \{a_1, a_2, ..., a_m\}$, the operator $\max_{a \in \mathcal{A}} f(a)$ returns the highest value $f(a)$ for all elements in the set $\mathcal{A}$.
- On the other hand, the operator $\arg\max_{a \in \mathcal{A}} f(a)$ returns the element of the set $\mathcal{A}$ that maximizes $f(a)$.
- Sometimes, when the set is implicit or explicit, we can write

$$\max_a f(a) \quad \text{or} \quad \arg\max_a f(a)$$

- Operator $\min$ and $\arg\min$ operates in a similar manner.
- Note: **arg max** returns a value from the **domain** of the function and **max** returns from the **range (codomain)** of the function

Functions, derivative and gradient

Derivative and Gradient

- A derivative f' of a function f is a function or a value that describes how fast f grows (or decreases).
- If the derivative is a constant value, like 5 or -3, then the function grows (or decreases) constantly at any point x of its domain.
- If the derivative f' is positive at some point x, then the function f grows at this point.
- If the derivative f' is negative at some point x, then the function f decreases at this point.
- The **derivative of zero** at x means that the **function's slope** at x is **horizontal**.

Functions, derivative and gradient

Partial Derivative

- **Differentiation of a scalar function *w.r.t.* a vector**
- If $f(\mathbf{w})$ is a scalar function of d variables, $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $f(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times 1$ vector.

$$\frac{df(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

This is referred to as the **gradient** of $f(\mathbf{w})$ and written as $\nabla_{\mathbf{w}} f$.

Functions, derivative and gradient

Partial Derivative

- **Differentiation of a vector function *w.r.t.* a vector**
- If $\mathbf{f}(\mathbf{w})$ is a vector function of size $h \times 1$ and $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $\mathbf{f}(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times h$ vector.

$$\frac{d\mathbf{f}(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \cdots & \frac{\partial f_h}{\partial w_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial w_d} & \cdots & \frac{\partial f_h}{\partial w_d} \end{bmatrix}$$

- This is referred to as the **Jacobian** matrix of $\mathbf{f}(\mathbf{w})$, *i.e.*,

$$\mathbf{J} = \frac{d\mathbf{f}(\mathbf{w})}{d\mathbf{w}},$$

$$\mathbf{J}_{ij} = \frac{\partial f_j}{\partial w_i}.$$

Functions, derivative and gradient

Some Vector-Matrix Differentiation Formulations

$$\frac{d(\mathbf{X}^\top \mathbf{w})}{d\mathbf{w}} = \mathbf{X}, \text{where } \mathbf{X} \text{ is not a function of } \mathbf{w}$$
$$\frac{d(\mathbf{y}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = \mathbf{X}^\top \mathbf{y}$$
$$\frac{d(\mathbf{w}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = (\mathbf{X} + \mathbf{X}^\top)\mathbf{w}$$

- Note that we adopt the denominator layout derivative. If you use the numerator layout derivative, then all above results will be transposed.
- Both types are OK, but keep it **consistent** in all derivatives.
- Please refer to the following wiki page for more details: `https://en.wikipedia.org/wiki/Matrix_calculus`

Linear regression

Dataset:

- We have a collection of m labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, with $\mathbf{x}_i \in \mathcal{X}$ being the d-dimensional feature vector of the i-th example, and $y_i \in \mathcal{Y}$ being a real-valued target.

Linear hypothesis function:

- We want to build a linear model $f_{\mathbf{w},b}(\mathbf{x})$, *i.e.*, linear hypothesis function,

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b,$$

  where $\mathbf{w}$ is a d-dimensional vector of parameters, and the bias parameter b is a real number.
- Note: $f_{\mathbf{w},b}$ is called **linear** due to the linearity *w.r.t.* the parameter vector $[b; \mathbf{w}]$, rather than *w.r.t.* the feature vector $\mathbf{x}$.

Task of linear regression:

- Using the linear model $f_{\mathbf{w},b}$ to approximate the ground-truth target function $t : \mathcal{X} \to \mathcal{Y}$.
- Note: If $\mathcal{Y}$ is a **finite and discrete** set, then the task corresponds to a classification problem; if $\mathcal{Y}$ is a **continuous** space, then the task corresponds to a regression problem.

Learning objective function

- To find the optimal values for $\mathbf{w}^*$ and b^* which minimizes the following expression:
$$\frac{1}{m}\sum_{i=1}^{m}\left(f_{\mathbf{w},b}\left(\mathbf{x}_i\right)-y_i\right)^2$$
- In mathematics, the expression we minimize or maximize is called an objective function, or, simply, an objective.

Linear regression: deterministic perspective

- The expression $\left(f_{\mathbf{w},b}\left(\mathbf{x}_i\right) - y_i\right)^2$ in the above objective is called the loss function. It's a measure of penalty for mis-classification of example i.
- This particular choice of the loss function is called squared error loss.
- All model-based learning algorithms have a loss function and what we do to find the best model is we try to minimize the objective known as the cost function.
- In linear regression, the cost function is given by the average loss, also called the empirical risk.

Linear regression: probabilistic perspective

- We assume that the relationship between the input variable/feature $\mathbf{x}$ and the output variable y is

$$y = \mathbf{w}^\top \mathbf{x} + e, \text{ where } e \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

where e is called observation noise or residual error, and it is independent with any specific input $\mathbf{x}$.

- Thus, the output y can also be seen as a random variable, and its conditional probability is formulated as

$$p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma^2) \tag{2}$$

Linear regression: probabilistic perspective

Maximum log-likelihood estimation:

- The parameter $\mathbf{w}$ can be learned by maximum log-likelihood estimation (MLE), given the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, as follows

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w}} \log \mathcal{L}(\mathbf{w}; D), \tag{3}$$

$$\begin{aligned} \log \mathcal{L}(\mathbf{w}; D) &= \log \Big(\prod_{i=1}^m p(y_i | \mathbf{x}_i, \mathbf{w}) \Big) = \sum_{i=1}^m \log \mathcal{N}(\mathbf{w}^\top \mathbf{x}_i, \sigma^2) \\ &= -m \log(\sigma (2\pi)^{\frac{1}{2}}) - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2. \end{aligned} \tag{4}$$

- Removing the constants *w.r.t.* $\mathbf{w}$,

$$\mathbf{w}_{MLE} = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2, \tag{5}$$

which is exactly the same with the cost function from the deterministic perspective.

Linear regression with analytical solution

Learning (Training)

- Consider the set of feature vector $\mathbf{x}_i$ and target output y_i indexed by $i = 1, \ldots, m$, then a linear model $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$ can be packed as

$$f_{\mathbf{w},b}(\mathbf{X}) \quad \Leftrightarrow \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Learning model — Learning target vector

$$= \begin{bmatrix} \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ \mathbf{x}_m^\top \mathbf{w} \end{bmatrix}$$

$$\text{where } \mathbf{x}_i^\top \mathbf{w} = \quad [1, x_1, \ldots, x_d]_i \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

Note: The **bias term** is responsible for **shifting the line/plane** up or down.

Least Squares Regression

- In vector-matrix notation, the squared error loss function can be written compactly using $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$:

$$\begin{aligned} J(\mathbf{w}) &= \mathbf{e}^\top \mathbf{e} \\ &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \left(\mathbf{w}^\top \mathbf{X}^\top - \mathbf{y}^\top\right)(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{w}^\top \mathbf{X}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{y}^\top \mathbf{y} \\ &= \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{y}^\top \mathbf{y} \end{aligned}$$

Note: when $f_{\mathbf{w},b}(\mathbf{X}) = \mathbf{X}\mathbf{w}$, then

$$\sum_{i=1}^{m} \left(f_{\mathbf{w},b}\left(\mathbf{x}_i\right) - y_i\right)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}).$$

Linear Regression

Differentiating J($\mathbf{w}$) with respect to $\mathbf{w}$ and setting the result to 0 :

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}) &= \mathbf{0} \\
\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{y}^\top \mathbf{y}\right) &= \mathbf{0} \\
\Rightarrow 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y} &= \mathbf{0} \\
\Rightarrow 2\mathbf{X}^\top \mathbf{X}\mathbf{w} &= 2\mathbf{X}^\top \mathbf{y}
\end{aligned}$$

If $\mathbf{X}^\top \mathbf{X}$ is invertible, then

$$\begin{aligned}
&\text{Learning:} \quad \widehat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \\
&\text{Prediction:} \quad f_{\mathbf{w},b}\left(\mathbf{X}_{\text{new}}\right) = \mathbf{X}_{\text{new}} \widehat{\mathbf{w}}
\end{aligned}$$

Linear Regression

Example 1

$\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$

$\{x = -9\} \rightarrow \{y = -6\}$
$\{x = -7\} \rightarrow \{y = -6\}$
$\{x = -5\} \rightarrow \{y = -4\}$
$\{x = 1\} \rightarrow \{y = -1\}$
$\{x = 5\} \rightarrow \{y = 1\}$
$\{x = 9\} \rightarrow \{y = 4\}$

$$\underbrace{\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix}}_{\mathbf{y}}$$

(Least squares approximation).

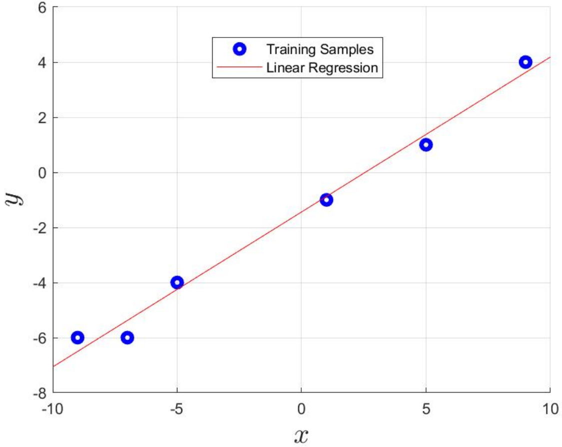
This set of linear equations has NO exact solution.

However ($\mathbf{X}^T\mathbf{X}$ is invertible)

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$

Linear regression



$$\hat{y} = \mathbf{X}\widehat{\mathbf{w}} = \mathbf{X}\begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$

$$y = -1.4375 + 0.5625x$$

Prediction:

$$\{x = -1\} \rightarrow \{y = ?\}$$

$$\hat{y} = [1 \; -1]\begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix} = -2$$

Linear Regression for one-dimensional examples.

Example 2 $\{(\mathbf{x}_i,\ y_i)\}_{i=1}^{m}$

$\{x_1 = 1,\ x_2 = 1,\ \ x_3 = 1\} \rightarrow \{y = 1\}$
$\{x_1 = 1,\ x_2 = -1, x_3 = 1\} \rightarrow \{y = 0\}$
$\{x_1 = 1,\ x_2 = 1,\ \ x_3 = 3\} \rightarrow \{y = 2\}$
$\{x_1 = 1,\ x_2 = 1,\ \ x_3 = 0\} \rightarrow \{y = -1\}$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}}_{\mathbf{y}} \qquad \mathbf{w}^T\mathbf{X}^T$$

This set of linear equations has NO exact solution.

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

However ($\mathbf{X}^T\mathbf{X}$ is invertible)

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix} \approx \begin{bmatrix} -0.75 \\ 0.18 \\ 0.93 \end{bmatrix}$$

(Least squares approximation).

Linear Regression

Prediction:

$$\{x_1 = 1, x_2 = 6, x_3 = 8\} \rightarrow \{y = ?\}$$
$$\{x_1 = 1, x_2 = 0, x_3 = -1\} \rightarrow \{y = ?\}$$

$$\hat{y} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.75 \\ 0.18 \\ 0.93 \end{bmatrix}$$
$$= \begin{bmatrix} 7.7500 \\ -1.6786 \end{bmatrix}$$

Linear regression solved by gradient descent

- The linear regression is formulated to the following optimization problem

$$\bar{\mathbf{w}}^* = \arg\min_{\bar{\mathbf{w}}} J(\bar{\mathbf{w}}), \quad J(\bar{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x}_i^\top \mathbf{w} + b - y_i)^2 = \frac{1}{2}(\mathbf{X}\bar{\mathbf{w}} - \mathbf{y})^2, \quad (6)$$

  where $\mathbf{X} = [(1, \mathbf{x}_1^\top); \cdots; (1, \mathbf{x}_m^\top)] \in \mathbb{R}^{m\times(d+1)}$, and $\bar{\mathbf{w}} = [b; \mathbf{w}] \in \mathbb{R}^{(d+1)\times 1}$. Note: for clarity, we will also use $\mathbf{w}$ to represent $\bar{\mathbf{w}}$, when b is not explicitly written.

- $\mathbf{w}$ can be updated by gradient descent algorithm,

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}, \ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) \quad (7)$$

  where α is called step-size or learning rate.

- Does gradient descent always converge to the optimal solution? (Plot the update trajectory of gradient descent on loss curve)

Closed-form solution vs. gradient descent

Closed-form solution $\bar{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$	**Gradient descent** $\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} - \alpha \mathbf{X}^\top(\mathbf{X}\bar{\mathbf{w}} - \mathbf{y})$
No hyper-parameter	Needs to choose α
No need to iterate	Needs many iterations
Complexity $O(d^3 + md^2)$	Complexity $O(T \times md^2)$
Slow if d is very large	Works well when d is large

Thus, you can choose between above two solutions according to the dimensionality of your training data:

- When the training data is very high-dimensional, *i.e.*, d is very **large**, then it is better to choose **gradient descent algorithm**
- When the training data is not high-dimensional, *i.e.*, d is very **small**, then it is better to choose **closed-form solution**

Linear regression with single output

When considering the entire set of data indexed by $i = 1, \ldots, m$, a linear model $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$ can be packed as

$$\begin{aligned} f_{\mathbf{w},b}(\mathbf{X}) &= \mathbf{X}\mathbf{w} \leftarrow \text{Scalar function} \\ &= \begin{bmatrix} \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ \mathbf{x}_m^\top \mathbf{w} \end{bmatrix} \quad \text{where} \quad \mathbf{x}_i^\top \mathbf{w} = [1, x_{i,1}, \ldots, x_{i,d}] \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \end{aligned} \tag{8}$$

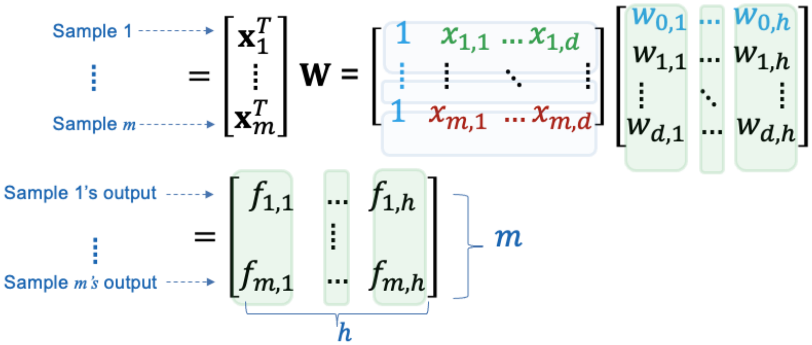
Primal solution: $\widehat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$

Note: The **bias term** is responsible for **shifting the line/plane** up or down.

Linear regression with multiple outputs

- When considering the entire set of data indexed by $i = 1, \dots, m$, a linear model $\mathbf{f}_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{W} + \mathbf{b}^\top$ can be packed as

$$\mathbf{f}_{\mathbf{w},b}(\mathbf{X}) = \mathbf{X}\mathbf{W}$$



Linear regression with multiple outputs

In matrix-matrix notation, the squared error loss function can be written compactly using $\mathbf{E} = \mathbf{X}\mathbf{W} - \mathbf{Y}$:

$$
\begin{aligned}
\mathrm{J}(\mathbf{W}) &= \operatorname{trace}\left(\mathbf{E}^\top \mathbf{E}\right) \\
&= \operatorname{trace}\left[(\mathbf{X}\mathbf{W} - \mathbf{Y})^\top (\mathbf{X}\mathbf{W} - \mathbf{Y})\right]
\end{aligned}
$$

If $\mathbf{X}^\top \mathbf{X}$ is invertible, then

$$
\begin{aligned}
\text{Learning: } \quad & \widehat{\mathbf{W}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-\mathbf{1}} \mathbf{X}^\top \mathbf{Y} \\
\text{Prediction: } \quad & \mathbf{f}_{\mathbf{w},b}\left(\mathbf{X}_{\text{new}}\right) = \mathbf{X}_{\text{new}} \widehat{\mathbf{W}}
\end{aligned}
$$

Assumption: the error terms $\mathbf{e}_k^\top \mathbf{e}_k$ are independent for all $k = 1, \ldots, h$

Linear regression with multiple outputs

$$
\begin{aligned}
J(\mathbf{W}) &= \text{trace}(\mathbf{E}^T\mathbf{E}) \\
&= \text{trace}\left(\begin{bmatrix} \mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_h^T \end{bmatrix} [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_h]\right) \\
&= \text{trace}\left(\begin{bmatrix} \mathbf{e}_1^T\mathbf{e}_1 & \mathbf{e}_1^T\mathbf{e}_2 & \dots & \mathbf{e}_1^T\mathbf{e}_h \\ \mathbf{e}_2^T\mathbf{e}_1 & \mathbf{e}_2^T\mathbf{e}_2 & \dots & \mathbf{e}_2^T\mathbf{e}_h \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{e}_h^T\mathbf{e}_1 & \mathbf{e}_h^T\mathbf{e}_2 & \dots & \mathbf{e}_h^T\mathbf{e}_h \end{bmatrix}\right) = \sum_{k=1}^{h} \mathbf{e}_k^T\mathbf{e}_k
\end{aligned}
$$

Linear regression

$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$

Example

$$\{x_1 = 1, x_2 = 1, x_3 = 1\} \rightarrow \{y_1 = 1, \quad y_2 = 0\}$$
$$\{x_1 = 1, x_2 = -1, x_3 = 1\} \rightarrow \{y_1 = 0, \quad y_2 = 1\}$$
$$\{x_1 = 1, x_2 = 1, \quad x_3 = 3\} \rightarrow \{y_1 = 2, y_2 = -1\}$$
$$\{x_1 = 1, x_2 = 1, \quad x_3 = 0\} \rightarrow \{y_1 = -1, y_2 = 3\}$$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix}}_{\mathbf{Y}}$$

This set of linear equations has NO exact solution.

$$\hat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

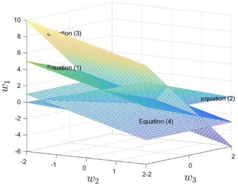
However ($\mathbf{X}^T\mathbf{X}$ is invertible)

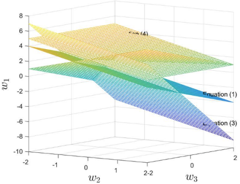
$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

(Least squares approximation).

Linear regression with multiple outputs

Example:





Prediction:

$$\{x_1 = 1, x_2 = 6, \quad x_3 = 8\} \rightarrow \{y_1 = ?, y_2 = ?\}$$
$$\{x_1 = 1, x_2 = 0, x_3 = -1\} \rightarrow \{y_1 = ?, y_2 = ?\}$$

$$\hat{\mathbf{Y}} = \mathbf{X}_t \widehat{\mathbf{W}}$$

$$= \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

$$= \begin{bmatrix} 7.75 & -7.25 \\ -1.6786 & 3.4643 \end{bmatrix}$$

Linear regression for classification

Dataset:

- We have a collection of m labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, with $\mathbf{x}_i \in \mathcal{X}$ being the d-dimensional feature vector of the i-th example, and $y_i \in \mathcal{Y}$ being a real-valued target.

Linear hypothesis function:

- We want to build a linear model $f_{\mathbf{w},b}(\mathbf{x})$, *i.e.*, linear hypothesis function,

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b,$$

  where $\mathbf{w}$ is a d-dimensional vector of parameters, and the bias parameter b is a real number.
- Note: $f_{\mathbf{w},b}$ is called linear due to the linearity *w.r.t.* the parameter vector $[b; \mathbf{w}]$, rather than *w.r.t.* the feature vector $\mathbf{x}$.

Task of linear regression:

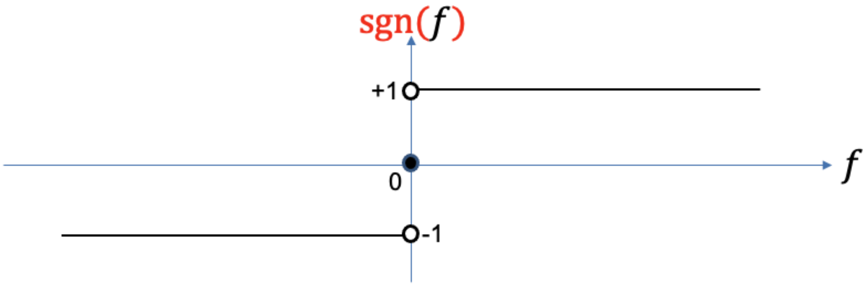
- Using the linear model $f_{\mathbf{w},b}$ to approximate the ground-truth target function $t : \mathcal{X} \to \mathcal{Y}$.
- Note: If $\mathcal{Y}$ is a **finite and discrete** set, then the task corresponds to a **classification** problem; if $\mathcal{Y}$ is a **continuous** space, then the task corresponds to a regression problem.

Linear regression for classification

Binary Classification: If $\mathbf{X}^\top \mathbf{X}$ is invertible, then

$$\text{Learning:} \quad \widehat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-\mathbf{1}} \mathbf{X}^\top \mathbf{y}, \quad y_i \in \{-1, +1\}, i = 1, \ldots, m$$

$$\text{Prediction:} \; f_{\mathbf{w},b}\left(\mathbf{x}_{new}\right) = \text{sgn}\left(\mathbf{x}_{new}^\top \widehat{\mathbf{w}}\right) \left(\text{ for each row } \mathbf{x}_{new}^\top \text{ of } \mathbf{X}_{new}\right)$$



Linear regression for classification

Example

$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$\{x = -9\} \rightarrow \{y = -1\}$
$\{x = -7\} \rightarrow \{y = -1\}$
$\{x = -5\} \rightarrow \{y = -1\}$
$\{x = 1\} \rightarrow \{y = +1\}$
$\{x = 5\} \rightarrow \{y = +1\}$
$\{x = 9\} \rightarrow \{y = +1\}$

$$\underbrace{\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{\mathbf{y}}$$

This set of linear equations has NO exact solution.

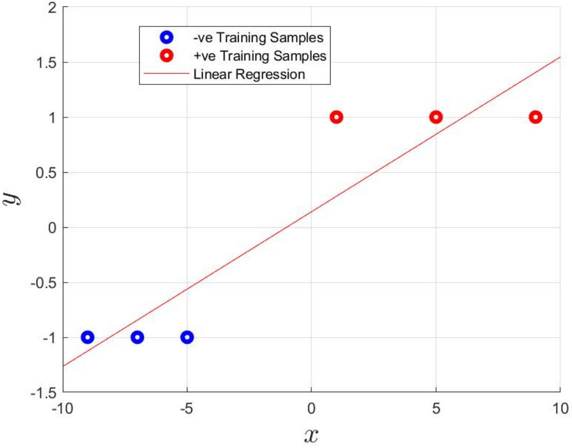
However ($\mathbf{X}^T\mathbf{X}$ is invertible)

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}$$

(Least squares approximation).

Linear regression for classification



$$\hat{y} = \text{sgn}(\mathbf{X}\widehat{\mathbf{w}})$$
$$= \text{sgn}\left(\mathbf{X}\begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}\right)$$
$$y = 0.1406 + 0.1406x$$

Prediction:

$$\{x = -2\} \rightarrow \{y = ?\}$$
$$\hat{y} = \text{sgn}\left([1 \; -2]\begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}\right)$$
$$= \text{sgn}(-0.1406) = -1$$

Linear regression for one-dimensional classification.

Linear regression for classification

Linear Methods for Multi-Category Classification:

If $\mathbf{X}^\top \mathbf{X}$ is invertible, then

Learning: $\widehat{\mathbf{W}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}, \quad \mathbf{Y} \in \mathbf{R}^{m \times C}$

Prediction: $f_{\mathbf{w},b}\left(\mathbf{x}_{\text{new}}\right) = \arg\max_{i=1,\ldots,C}(\mathbf{x}_{\text{new}}^\top \widehat{\mathbf{W}})$(for each row $\mathbf{x}_{\text{new}}^\top$ of $\mathbf{X}_{\text{new}}$)

Each row (of i=1, ..., m) in Y has a one-hot **assignment:**

e.g., target for class-1 is labelled as $\mathbf{y}_i^\top = [1, 0, 0, \ldots, 0]$ for the i th sample,

target for class-2 is labelled as $\mathbf{y}_i^\top = [0, 1, 0, \ldots, 0]$ for the i th sample,

target for class-C is labelled as $\mathbf{y}_i^\top = [0, 0, \ldots, 0, 1]$ for the i th sample.

Linear regression for classification

Example $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$

$\{x_1 = 1, x_2 = 1, \ \ x_3 = 1\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$
$\{x_1 = 1, x_2 = -1, x_3 = 1\} \rightarrow \{y_1 = 0, y_2 = 1, y_3 = 0\}$
$\{x_1 = 1, x_2 = 1, \ \ x_3 = 3\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$
$\{x_1 = 1, x_2 = 1, \ \ x_3 = 0\} \rightarrow \{y_1 = 0, y_2 = 0, y_3 = 1\}$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Y}}$$

This set of linear equations has NO exact solution.

$$\widehat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

However ($\mathbf{X}^T\mathbf{X}$ is invertible) (Least squares approximation)

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Linear regression for classification

Example

Prediction:

$$\{x_1 = 1, x_2 = 6, \quad x_3 = 8\} \rightarrow \{ \text{ class } 1, 2, \text{ or } 3 \text{ ? } \}$$
$$\{x_1 = 1, x_2 = 0, x_3 = -1\} \rightarrow \{ \text{ class } 1, 2, \text{ or } 3 \text{ ? } \}$$

$$\widehat{\mathbf{Y}} = \mathbf{X}_t \widehat{\mathbf{W}} = \arg \max_{i=1,\ldots,C} \left(\begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix} \right)$$

$$= \arg \max_{i=1,\ldots,C} \left(\begin{bmatrix} 4 & -2.50 & -0.50 \\ -0.2587 & 0.50 & 0.7857 \end{bmatrix} \right)$$

Position of the largest number determines the class label

$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{array}{l} \rightarrow \text{Class-1} \\ \rightarrow \text{Class-3} \end{array}$$

Ridge regression

Motivation 1:

Recall the learning computation: $\widehat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$.

We cannot guarantee that the matrix $\mathbf{X}^\top \mathbf{X}$ is invertible.

Ridge regression

Here on, we shall focus on single output y in derivations in the sequel.

$$\min_{\mathbf{w},b} \sum_{i=1}^{m} \left(f_{\mathbf{w},b}\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \bar{\mathbf{w}}^\top \bar{\mathbf{w}}, \text{ where } \bar{\mathbf{w}} = \hat{\mathbf{I}}_d \mathbf{w} = [0, w_1, w_2, \ldots, w_d]^\top,$$

$\hat{\mathbf{I}}_d \in \mathbb{R}^{(d+1)\times(d+1)}$ is defined by setting the $(1,1)$ entry in the $d+1$ dimensional identity matrix $\hat{\mathbf{I}}_{d+1}$ as 0. Note: The bias/offset b is NOT included in the ℓ_2 regularization term, as it just affects the function's height, not its complexity.

Linear Model: $\min_{\mathbf{w},b} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \bar{\mathbf{w}}^\top \bar{\mathbf{w}}$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \bar{\mathbf{w}}^\top \bar{\mathbf{w}} = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y} + 2\lambda \hat{\mathbf{I}}_d \mathbf{w} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \hat{\mathbf{I}}_d \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

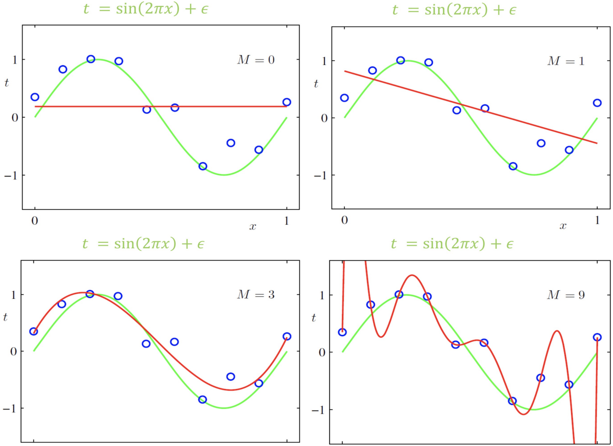
$$\Rightarrow \left(\mathbf{X}^\top \mathbf{X} + \lambda \hat{\mathbf{I}}_d\right) \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

$$\Rightarrow \mathbf{w} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \hat{\mathbf{I}}_d\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

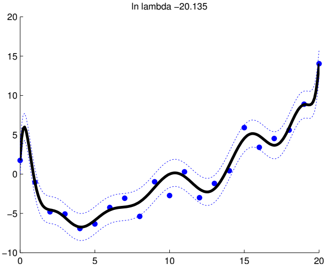
Note that $\left(\mathbf{X}^\top \mathbf{X} + \lambda \hat{\mathbf{I}}_d\right)$ is guaranteed to be invertible, given $\lambda > 0$.

Motivation 2:

- Overfitting is an important challenge for linear regression, as shown below. Note: M in the figure denotes the degree of polynomial hypothesis function.
- If ovefitting, the prediction performance on testing data will be very poor. How to alleviate ovefitting?



Ridge regression



- Let's see one simple example, we use a **polynomial function** (introduced later) with 14 degree to fit $m = 21$ data points. The learned curve is very "wiggly" (see above).
- The parameter values of this curve are as follows

$$6.56, -36.934, -109.25, 543.452, 1022.561, -3046.224, -3768.013, 8524.54,$$
$$6607.897, -12640.058, -5530.188, 9479.73, 1774, 639, -2821.526$$

- There are many large positive/negative values, such that a small change of features could lead to significant change of output.

Ridge regression

- How to get smaller parameter values?
- We can assume that the parameter $\mathbf{w}$ (excluding the bias b) follow a zero-mean Gaussian prior

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \tau^2 \mathbf{I}) \tag{9}$$

- For clarity, **we omit the bias b in the following derivation**.
- Utilizing this prior, we obtain the maximum a posteriori (MAP) estimation

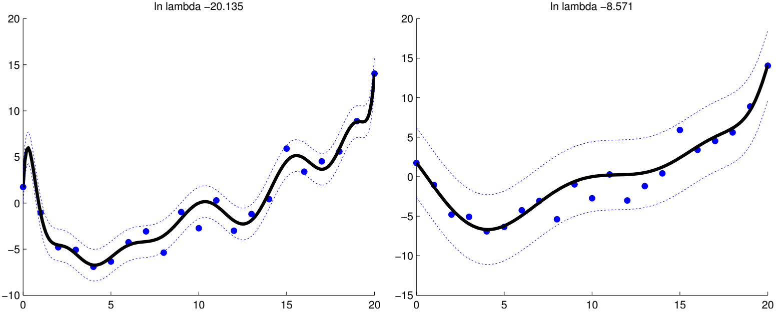
$$\begin{aligned}
\mathbf{w}_{MAP} &= \arg\max_{\mathbf{w}} \big[\sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w}) \big] && (10) \\
&= \arg\max_{\mathbf{w}} \big[\sum_{i=1}^{m} \log \mathcal{N}(\mathbf{x}_i^\top \mathbf{w}, \sigma^2) + \log \mathcal{N}(\mathbf{w}|\mathbf{0}, \tau^2 \mathbf{I}) \big] && (11) \\
&\equiv \arg\min_{\mathbf{w}} \big[\sum_{i=1}^{m} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \lambda \|\mathbf{w}\|_2^2 \big]. && (12)
\end{aligned}$$

- The corresponding closed-form solution is given by

$$\mathbf{w}_{MAP} = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{13}$$

Ridge regression

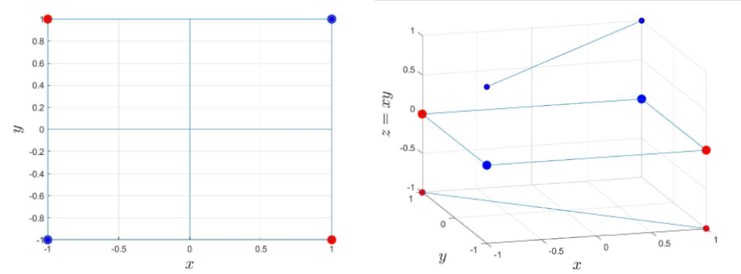
- The above method is also known as ridge regression, or penalized least squares.
- In general, adding a Gaussian prior to the parameters of a model to encourage them to be small is called ℓ_2 **regularization** or **weight decay**.
- As shown below, when we set a larger λ, *i.e.*, more weight on the prior, the resulting curve will be smoother.



Motivation

- Some data may be not linearly separated, such as the classic **XOR** data, as shown on the bottom left.
- Consequently, the linear regression model doesn't work.
- To tackle it, we could project the original data to the **monomial** axis $x_1 x_2$.
- Then, the XOR becomes linearly separated, as shown on the bottom right.
- Accordingly, we can design a novel linear regression model, as follows

$$f_{\mathbf{w},b}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2, \text{ where } w_0 = b.$$



Polynomial regression

Polynomial expansion

- The linear model $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$ can be written as

$$f_{\mathbf{w},b}(\mathbf{x}) = \sum_{i=0}^{d} x_i w_i = w_0 + \sum_{i=1}^{d} x_i w_i.$$

- By including terms involving the products of pairs of components of $\mathbf{x}$, we obtain a quadratic model:

$$f_{\mathbf{w},b}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j.$$

Polynomial regression

- In general:

$$f_{\mathbf{w},b}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \dots$$

Remarks

- For high dimensional d and high *polynomial order*, the number of polynomial terms becomes explosive! (In fact, this number of terms grows exponentially.)
- Hence, for high dimensional problems, polynomials of order larger than 3 is seldom used.

Polynomial regression

Linear model with basis expansion $\phi(\mathbf{x})$

$$
\begin{aligned}
f_{\mathbf{w},b}(\mathbf{x}) &= w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \dots \\
&= \phi(\mathbf{x})^\top \mathbf{w},
\end{aligned}
$$

where

$$
\begin{aligned}
\phi(\mathbf{x}) &= [1, x_1, \dots, x_d, \dots, x_i x_j, \dots, x_i x_j x_k, \dots]^\top, \\
\mathbf{w} &= [w_0, w_1, \dots, w_d, \dots, w_{ij}, \dots, w_{ijk}, \dots]^\top.
\end{aligned}
$$

Note: $f_{\mathbf{w},b}(\mathbf{x})$ is still a linear function *w.r.t.* $\mathbf{w}$, rather than $\mathbf{x}$. Thus, it is still a **linear model**.

Extending to the case of m data points, *i.e.*, $\mathbf{X} = [\mathbf{x}_1^\top; \dots; \mathbf{x}_m^\top] \in \mathbb{R}^{m \times (d+1)}$, the basis expansion is presented by

$$
\mathbf{P}(\mathbf{X}) = [\phi(\mathbf{x}_1)^\top; \dots; \phi(\mathbf{x}_m)^\top] \in \mathbb{R}^{m \times |\mathbf{w}|}.
$$

Polynomial regression

Example

$$\{\mathbf{x_i}, \mathbf{y_i}\}_{\mathbf{i=1}}^{\mathbf{m}} \quad \begin{array}{l} \{x_1 = 0, x_2 = 0\} \to \{y = 0\} \\ \{x_1 = 1, x_2 = 1\} \to \{y = 1\} \\ \{x_1 = 1, x_2 = 0\} \to \{y = 2\} \\ \{x_1 = 0, x_2 = 1\} \to \{y = 3\} \end{array}$$

$2^{\mathbf{nd}}$ **order** polynomial model

$$f_{\mathbf{w},b}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2$$

$$= \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_{12} \\ w_{11} \\ w_{22} \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Polynomial regression

Ridge regression with original features $\mathbf{X}$:

$$\begin{array}{ll} \text{Learning:} & \hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \\ \text{Prediction:} & f_{\mathbf{w},b}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}} \end{array}$$

Ridge regression with basis expansion $\mathbf{P}(\mathbf{X})$:

$$\begin{array}{ll} \text{Learning:} & \hat{\mathbf{w}} = (\mathbf{P}^\top\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^\top\mathbf{y} \\ \text{Prediction:} & f_{\mathbf{w},b}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}} \end{array}$$

Polynomial regression

For Regression Applications

- Learning: $\hat{\mathbf{w}} = (\mathbf{P}^\top \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^\top \mathbf{y}$, where $\mathbf{y}$ is continuous
- Prediction: $f_{\mathbf{w},b}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new} \hat{\mathbf{w}}$

For Classification Applications

- Learn **discrete** valued $\mathbf{y}$ (binary) or $\mathbf{Y}$ (one-hot)
- Binary Prediction: $f_{\mathbf{w},b}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{sgn}(\mathbf{P}_{new} \hat{\mathbf{w}})$ if $y \in \{-1, +1\}$
- Multi-Category Prediction: $f_{\mathbf{w},b}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{argmax}_{i=1,\ldots,C}(\mathbf{P}_{new} \hat{\mathbf{w}})$

Polynomial regression

Example (Cont'd)

$\{\mathbf{x_i}, \mathbf{y_i}\}_{\mathbf{i=1}}^{\mathbf{m}}$
$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

$2^{\mathbf{nd}}$ **order** polynomial model

$$\mathbf{P} = \begin{bmatrix} 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{P}^\top(\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{y} \text{ (note: under determined linear system)}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 6 & 3 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

Polynomial regression

Example (Cont'd) **Prediction:**

Test point 1: $\{x_1 = 0.1, x_2 = 0.1\} \rightarrow \{y = \text{class} \ -1 \text{ or } \ +1?\}$

Test point 2: $\{x_1 = 0.9, x_2 = 0.9\} \rightarrow \{y = \text{class} \ -1 \text{ or } \ +1?\}$

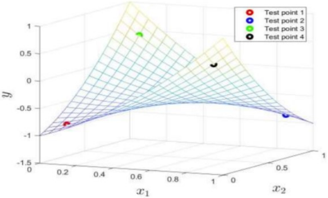
Test point 3: $\{x_1 = 0.1, x_2 = 0.9\} \rightarrow \{y = \text{class} \ -1 \text{ or } \ +1?\}$

$\hat{y} = \mathbf{P}_t \hat{\mathbf{w}}$ Test point 4: $\{x_1 = 0.9, x_2 = 0.1\} \rightarrow \{y = \text{class} \ -1 \text{ or } \ +1?\}$

$$\hat{\mathbf{y}} = \mathbf{sgn}\left(\begin{bmatrix} 1 & 0.1 & 0.1 & 0.01 & 0.01 & 0.01 \\ 1 & 0.9 & 0.9 & 0.81 & 0.81 & 0.81 \\ 1 & 0.1 & 0.9 & 0.09 & 0.01 & 0.81 \\ 1 & 0.9 & 0.1 & 0.09 & 0.81 & 0.01 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix} \right)$$

$$= \text{sgn}\left(\begin{bmatrix} -0.82 \\ -0.82 \\ 0.46 \\ 0.46 \end{bmatrix} \right)$$

$$= \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \dashrightarrow & \text{Class -1} \\ \dashrightarrow & \text{Class -1} \\ \dashrightarrow & \text{Class +1} \\ \dashrightarrow & \text{Class +1} \end{matrix}$$



Lasso regression

- We can replace the Gaussian prior by a Laplacian prior, *i.e.*,

$$p(\mathbf{w}) = \text{Lap}(\mathbf{w}|\mathbf{0}, \lambda) = \frac{1}{2\lambda} \exp\Big(- \frac{|\mathbf{w}|}{\lambda} \Big), \tag{14}$$

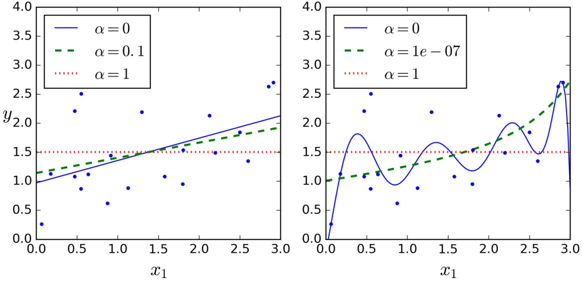
- The combination of the Gaussian distribution of $p(y|\mathbf{x}, \mathbf{w})$ and the Laplacian prior, leading to

$$\mathbf{w}_{MAP} = \arg\max_{\mathbf{w}} \big[\sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w}) \big] \tag{15}$$

$$= \arg\max_{\mathbf{w}} \big[\sum_{i=1}^{m} \log \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma^2) + \text{Lap}(\mathbf{w}|\mathbf{0}, b) \big] \tag{16}$$

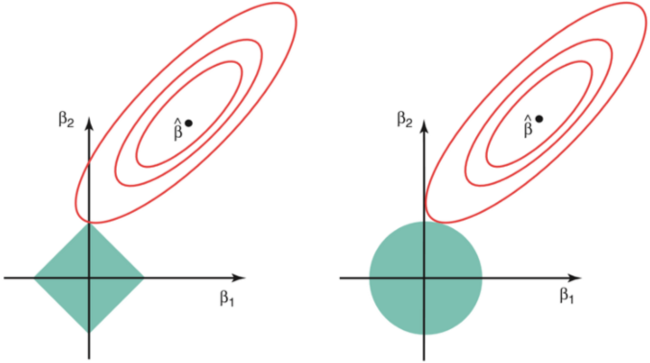
$$\equiv \arg\min_{\mathbf{w}} \big[\sum_{i=1}^{m} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \alpha|\mathbf{w}| \big]. \tag{17}$$

- It is **Lasso regression**, and the regularization is called ℓ_1 **regularization**. It will encourage the sparse parameters.
- As shown below, when we set a larger α, *i.e.*, more weight on the prior, the resulting curve will be smoother.



Geometry of Ridge and Lasso regression

- Geometry of Ridge and Lasso regression. Which one is Ridge?

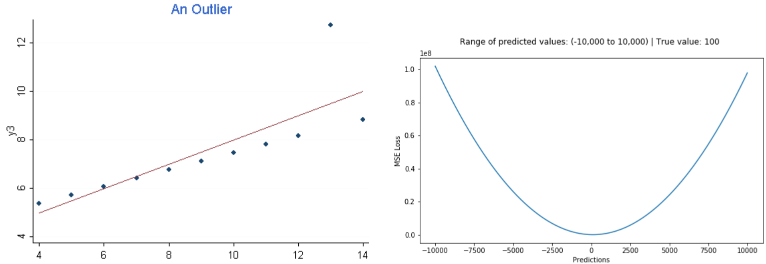


Robust linear regression

- When there is a few outliers in the training data D, which are far from most other points, then learned parameters $\mathbf{w}_{MLE}$ will be significantly influenced, leading to very poor fit.
- Let's see the loss curve of the residual sum of squares (RSS),

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2. \tag{18}$$

- The error increases quadratically along with the residual. To minimize such a large error, the linear model will be significantly changed.
- How to alleviate the significant influence of outliers?

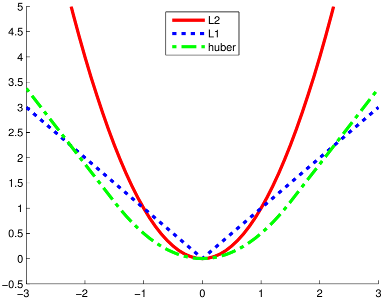


Robust linear regression

- We adopt the ℓ_1 loss to replace the ℓ_2 loss, as follows

$$J(\mathbf{w}) = \sum_{i=1}^{m} |\mathbf{x}_i^\top \mathbf{w} - y_i|. \tag{19}$$

- The curves of ℓ_1 and ℓ_2 losses are shown as follows.
- When the residual is large, the ℓ_1 loss is much smaller than the ℓ_2 loss, such that the influence of outliers could be alleviated.



Robust linear regression

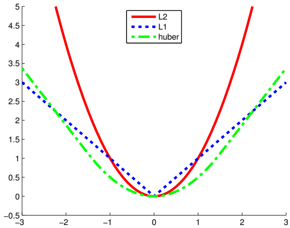
- Actually, the above ℓ_1 loss can also be derived from the probabilistic perspective, by assuming that

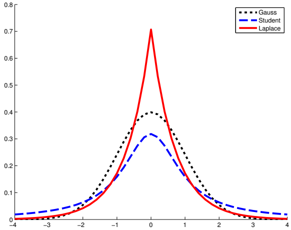
$$p(y|\mathbf{x}, \mathbf{w}, b) = \text{Lap}(y|\mathbf{x}, \mathbf{w}, b) \propto \exp(-\frac{1}{b}|y - \mathbf{w}^\top \mathbf{x}|) \tag{20}$$

- Applying the maximum log-likelihood estimation (MLE), we will obtain

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w}} \log \mathcal{L}(\mathbf{w}; D) = \arg\max_{\mathbf{w}} \sum_{i}^{m} \log p(y|\mathbf{x}, \mathbf{w}) \tag{21}$$

$$\equiv \arg\min_{\mathbf{w}} \frac{1}{b} \sum_{i=1}^{m} |\mathbf{x}_i^\top \mathbf{w} - y_i| \tag{22}$$





Robust linear regression

$$\mathbf{w}_{MLE} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} |\mathbf{x}_i^\top \mathbf{w} - y_i| \tag{23}$$

- However, the ℓ_1 loss function is non-differentiable. The gradient descent algorithm cannot be adopted.
- We can transform it to a linear program, as follows

$$\min_{\mathbf{w}, \boldsymbol{t}} \sum_{i}^{m} t_i \tag{24}$$
$$s.t. \ -t_i \leq \mathbf{x}_i^\top \mathbf{w} - y_i \leq t_i, 1 \leq i \leq m.$$

Please refer to:
`https://math.stackexchange.com/questions/1639716/how-can-l-1-norm-minimization-with`

Robust linear regression

$$\mathbf{w}_{MLE} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} |\mathbf{x}_i^\top \mathbf{w} - y_i| \tag{25}$$

- We can also utilize the following equation:

$$|a| = \min_{\mu>0} \frac{1}{2}(\frac{a^2}{\mu} + \mu) \tag{26}$$

- Then, the above ℓ_1 minimization problem (25) can be reformulated as follows

$$\min_{\mathbf{w}} \min_{\mu_1,\ldots,\mu_m>0} \frac{1}{2}\Big(\frac{(\mathbf{x}_i^\top \mathbf{w} - y_i)^2}{\mu_i} + \mu_i\Big). \tag{27}$$

- It can be iteratively and alternatively optimized as follows:
  - Given $\mathbf{w}$, $\mu_i = |\mathbf{x}_i^\top \mathbf{w} - y_i|, i = 1, \ldots, m$
  - Given μ, $\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} \frac{1}{2}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$
- It is called iteratively reweighted least squares method.

Summary of different variants of linear regressions

Note that the uniform distribution will not change the mode of the likelihood. Thus, MAP estimation with a uniform prior corresponds to MLE.

$p(y\|\mathbf{x},\mathbf{w})$	$p(\mathbf{w})$	regression method
Gaussian	Uniform	Least squares
Gaussian	Gaussian	Ridge regression
Gaussian	Laplace	Lasso regression
Laplace	Uniform	Robust regression
Student	Uniform	Robust regression