AI6102: Machine Learning Methodologies & Applications

L3: Linear Models: Regression

Dacheng Tao
dacheng.tao@ntu.edu.sg
Nanyang Technological University, Singapore

Recall: Supervised Learning

In mathematics

- Given: a set of N labeled data $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i$ is m-dimensional vector of numerical values, and y_i is a scalar
- We aim to learn a mapping $f: \boldsymbol{x} \rightarrow y$ by requiring $f(\boldsymbol{x}_i) = y_i$
- The learned mapping f is expected to make precise predictions on any unseen $\boldsymbol{x}^*$ as $f(\boldsymbol{x}^*)$

Hypothesis

- A mapping or function $f\colon \boldsymbol{x} \to y$ can be considered as an element of some space of possible functions $\mathcal{H}\colon\ \mathbb{R}^m \to \mathbb{R}$, often called hypothesis space
- Supervised learning aims to find a hypothesis $f \in \mathcal{H}$ from training data $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, s.t. for any test $\boldsymbol{x}^*$, $f(\boldsymbol{x}^*) = y^*$

Assumption

- The training data instances $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$, are independent and identically distributed (i.i.d.), and drawn from an unknown joint probability distribution $P_{tr}(\boldsymbol{x}, y)$
- Unseen test data instances $\{(\boldsymbol{x}^*, y^*)\}$ are also i.i.d, and drawn from an unknown joint probability distribution $P_{ts}(\boldsymbol{x}, y)$
- The two joint probability distributions are the same: $P_{tr}(\boldsymbol{x}, y) = P_{ts}(\boldsymbol{x}, y)$

Loss Function

- Denote by $\hat{y} = f(\boldsymbol{x})$ the prediction of the function $f(\cdot)$ on a data instance $\boldsymbol{x}$, and y is the ground-truth output of $\boldsymbol{x}$
- Let $\ell: \mathbb{R}\times\mathbb{R} \rightarrow \mathbb{R}_+ \geq 0$ be a loss function to measure the difference between the ground-truth output y and the prediction $\hat{y}$ of a hypothesis $f(\cdot)$ on any input data instance $\boldsymbol{x}$

$$\ell(f(\boldsymbol{x}), y) = \ell(\hat{y}, y)$$

Risk Minimization

- The risk associated with a hypothesis $f(\cdot)$ is defined as the expectation of the loss function over all possible input-output pairs drawn from a joint probabilistic distribution $P(\boldsymbol{x}, y)$:

$$R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}[\ell(f(\boldsymbol{x}), y)]$$

- Recall: in supervised learning, the learned hypothesis $f(\cdot)$ is expected to make precise predictions on any test data instance $\boldsymbol{x}^*$, i.e., $f(\boldsymbol{x}^*) = y^*$

$$f^* = \arg\min_f R_{ts}(f) = \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim P_{ts}}[\ell(f(\boldsymbol{x}), y)]$$

! Test data is unseen in training! And even in the test phase, y is not observed!

Risk Minimization (cont.)

$$
\begin{aligned}
f^* &= \arg\min_f R_{ts}(f) \\
&= \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim P_{ts}}[\ell(f(\boldsymbol{x}), y)] \\
&= \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim P_{ts}}\left[\frac{P_{tr}(\boldsymbol{x},y)}{P_{tr}(\boldsymbol{x},y)}\ell(f(\boldsymbol{x}), y)\right] \\
&= \arg\min_f \int_y \int_{\boldsymbol{x}} P_{ts}(\boldsymbol{x},y)\left(\frac{P_{tr}(\boldsymbol{x},y)}{P_{tr}(\boldsymbol{x},y)}\ell(f(\boldsymbol{x}), y)\right) d\boldsymbol{x}dy
\end{aligned}
$$

Definition of expectation

$$
\boxed{\mathbb{E}_{\boldsymbol{x}\sim P}[g(\boldsymbol{x})] = \int_{\boldsymbol{x}} P(\boldsymbol{x})g(\boldsymbol{x})\,d\boldsymbol{x} = \int_{x_1} \dots \int_{x_m} P(\boldsymbol{x})g(\boldsymbol{x})\,dx_1 \dots dx_m}
$$

Risk Minimization (cont.)

$$f^* = \arg\min_f R_{ts}(f)$$

$$= \arg\min_f \int_y \int_{\boldsymbol{x}} \boxed{P_{ts}(\boldsymbol{x}, y)} \left(\frac{\boxed{P_{tr}(\boldsymbol{x}, y)}}{P_{tr}(\boldsymbol{x}, y)} \ell(f(\boldsymbol{x}), y) \right) d\boldsymbol{x}dy$$

Assumption
$P_{tr}(\boldsymbol{x}, y) = P_{ts}(\boldsymbol{x}, y)$

$= 1$

$$= \arg\min_f \int_y \int_{\boldsymbol{x}} P_{tr}(\boldsymbol{x}, y) \left(\boxed{\frac{P_{ts}(\boldsymbol{x}, y)}{P_{tr}(\boldsymbol{x}, y)}} \ell(f(\boldsymbol{x}), y) \right) d\boldsymbol{x}dy$$

$$= \arg\min_f \int_y \int_{\boldsymbol{x}} P_{tr}(\boldsymbol{x}, y) \ell(f(\boldsymbol{x}), y) \, d\boldsymbol{x}dy$$

Definition of expectation

$$= \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim P_{tr}}[\ell(f(\boldsymbol{x}), y)]$$

Empirical Risk Minimization

$$f^* = \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim P_{tr}}[\ell(f(\boldsymbol{x}), y)] = \arg\min_f R_{tr}(f)$$

- The distribution $P_{tr}(\boldsymbol{x}, y)$ is unknown, thus we are not able to sample (infinite) input-output pairs $\{(\boldsymbol{x}, y)\}$ to learn a hypothesis $f(\cdot)$!
- In practice, only a finite number of training pairs are available, $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
- Approximate the expected risk by empirical risk:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim P_{tr}}[\ell(f(\boldsymbol{x}), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i) = \hat{R}_{tr}(f)$$

Empirical Risk Minimization (cont.)

- In practice, the hypothesis f can be learned by minimizing the empirical risk

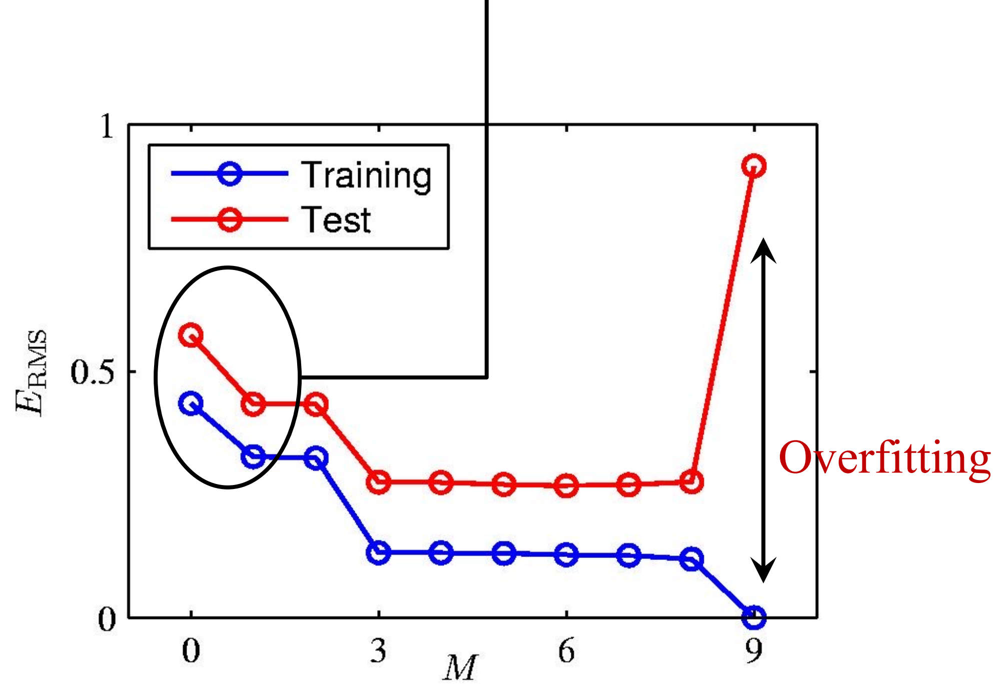
$$\hat{f} = \arg\min_{f} \hat{R}_{tr}(f) = \arg\min_{f} \frac{1}{N}\sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i)$$

- Given a training data set, N is a constant, thus for convenience in presentation, $\frac{1}{N}$ is dropped

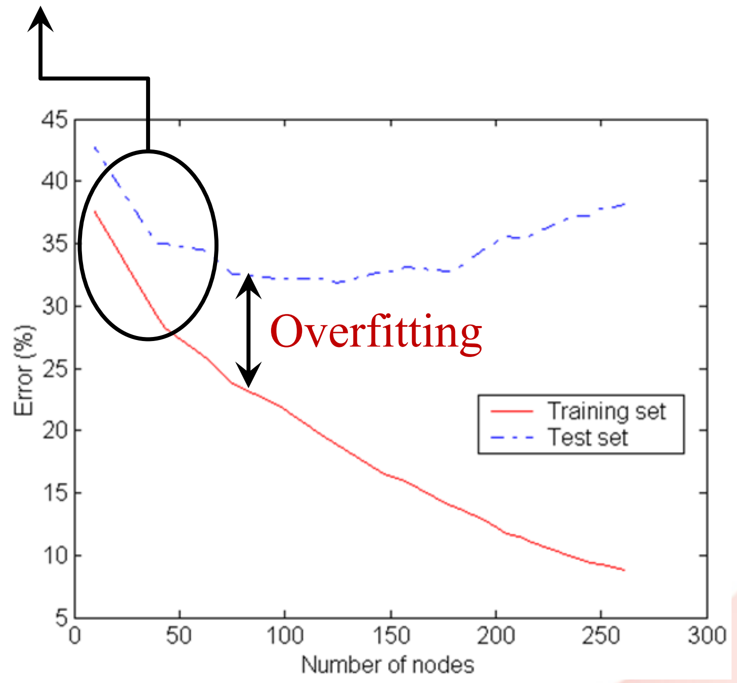
$$\hat{f} = \arg\min_{f} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i)$$

Overfitting Revisit

Underfitting: when model is too simple, both training and test error are large



Polynomial curve fitting



Decision tree classification

Overfitting: when test error begins to increase even though training error continues to decrease

Overfitting v.s. Model Complexity

- Observations of the two examples:
  - Increasing model complexity could make training error or training loss to keep being decreased
  - When model complexity keeps increasing, test error or test loss will increase after some point
- After a model is learned, we can use validation set to evaluate it to reduce the risk of overfitting (tuning hyper-parameters)
- Can we reduce the risk of overfitting when learning a model?

Occam’s Razor Principle

- Given two models of similar performance, we should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it is fitted accidentally by noise in data
  - Overfitting results in models that are more complex than necessary
- Therefore, we should include model complexity when learning a model

Structural Risk Minimization (cont.)

- Empirical Risk Minimization

$$\hat{f} = \arg\min_{f} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i)$$

- Structural Risk Minimization

$$\hat{f} = \arg\min_{f} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i) + \boxed{\lambda\Omega(f)}$$

  - $\Omega(f)$ is known as a penalty or regularization term to control the model complexity of f
  - $\lambda > 0$ is a trade-off hyper-parameter

Structural Risk Minimization (cont.)

$$\hat{f} = \arg\min_{f} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i) + \lambda\Omega(f)$$

- How to learn f?
  - Design a specific form of f in terms of some parameters, denoted by a vector $\boldsymbol{\theta} \in \mathbb{R}^{t\times 1}$, i.e., $f(x; \boldsymbol{\theta})$
  - The parameterized $f(x; \boldsymbol{\theta})$ defines a family of functions with different values of $\boldsymbol{\theta}$
  - Learning f is equivalent to learning the values of $\boldsymbol{\theta}$

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

Structural Risk Minimization (cont.)

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

- Popular regularization terms include
  - the squared L2 norm: $\|\boldsymbol{\theta}\|_2^2$
    - $\|\boldsymbol{\theta}\|_2^2 = \sum_{i=1}^{t} \theta_i^2$
    - Tends to prefer a model with a smaller value for each parameter θ_i
  - the L1 norm: $\|\boldsymbol{\theta}\|_1$
    - $\|\boldsymbol{\theta}\|_1 = \sum_{i=1}^{t} |\theta_i|$
    - Tends to prefer a model with a smaller value for each parameter θ_i, and fewer parameters with non-zero values
    - Induce sparsity, i.e., some θ_i's tend to be zeros

Linear Models: Regression

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

- In general, for regression, $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ is defined as

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}\cdot\boldsymbol{x} + b$$

$\boldsymbol{\theta}$ is a concatenation of $\boldsymbol{w}$ and b

- Given $\boldsymbol{x}_i$, the prediction of $f(\boldsymbol{x};\boldsymbol{\theta})$ is the linear combination of its m feature values with weights $\boldsymbol{w}$ plus a bias term b

$$\boldsymbol{x}_i \boxed{\begin{array}{c} x_{1i} \\ \hline x_{2i} \\ \hline \dots \\ \hline x_{mi} \end{array}} \qquad \boldsymbol{w} \boxed{\begin{array}{c} w_1 \\ \hline w_2 \\ \hline \dots \\ \hline w_m \end{array}}$$

$$\hat{y} = f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}\cdot\boldsymbol{x} + b = \sum_{i=1}^{m} x_i w_i + b$$

Linear Models: Classification

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

- In general, for classification, $f(\boldsymbol{x}_i; \boldsymbol{w})$ is defined as

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = h(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

where $h(z)$ is function to map continuous values to discrete values (denoting different categories)

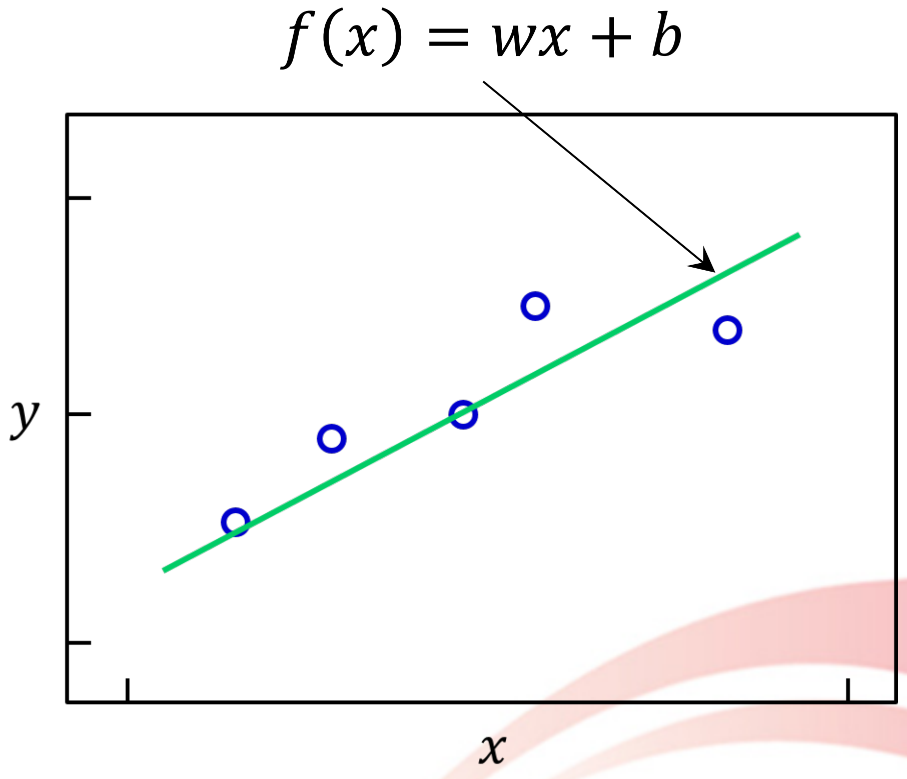
- For example,

$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

Next Lecture

Linear Regression: One-Dimension

- Each instance is represented by only one input feature
- To learn a linear function $f(x)$ in terms of w and b (both are scalars) from $\{x_i, y_i\}, i = 1, \dots, N$



1D Linear Regression (cont.)

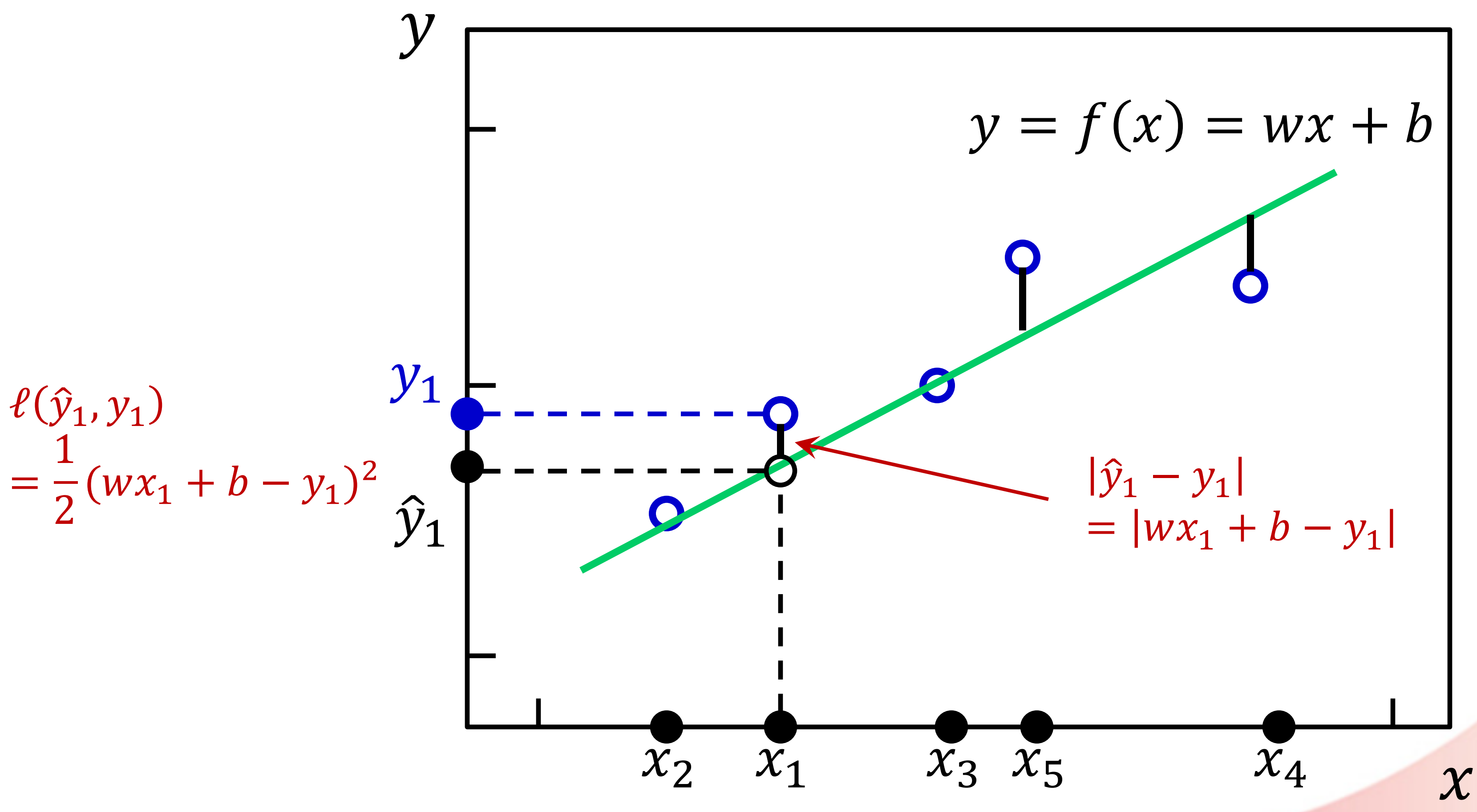
$$[\hat{w}, \hat{b}] = \arg\min_{[w,b]} \sum_{i=1}^{N} \ell(\hat{y}_i, y_i)$$

Drop the regularization term for simplicity at first

where $\hat{y}_i = wx_i + b$

- The loss function $\ell(\hat{y}_i, y_i)$ is to measure the difference between $\hat{y}_i$ and y_i
  - For regression, the magnitude of the difference, i.e., $|\hat{y}_i - y_i|$
- To make the resultant optimization problem easier to solve
  - We expect the loss function has some good properties, e.g., differentiable everywhere
  - The square of magnitude, $|\hat{y}_i - y_i|^2 = (\hat{y}_i - y_i)^2$ or $\frac{1}{2}(\hat{y}_i - y_i)^2$

Regression Loss Function



$$\frac{1}{2}\sum_{i=1}^{5}\ell(\hat{y}_i, y_i) = \frac{1}{2}\sum_{i=1}^{5}(wx_i + b - y_i)^2$$

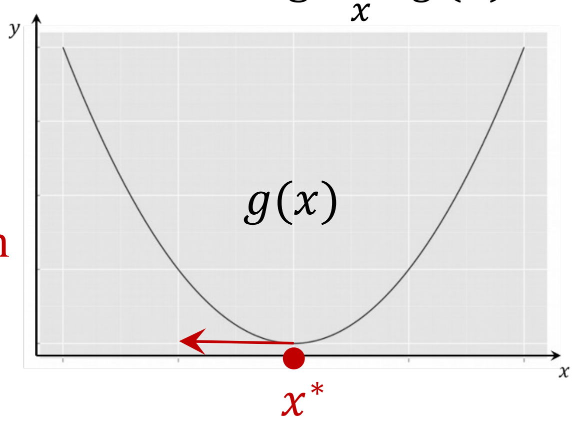
Optimization

- Learn w and b by minimizing the square loss

$$[\hat{w}, \hat{b}] = \arg\min_{[w,b]} \boxed{\frac{1}{2}\sum_{i=1}^{N}(wx_i + b - y_i)^2}$$

  - The objective of the optimization problem
  - The objective is convex

$$x^* = \arg\min_x g(x)$$



- Unconstrained optimization problem
- Set the derivatives of the objective w.r.t. w and b to zero, respectively
- $\hat{w}$ and $\hat{b}$ can be obtained by solving the equations

Closed-form Solution

$$\left\{\begin{array}{l} \dfrac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(wx_i+b-y_i)^2\right)}{\partial w}=0 \\ \\ \dfrac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(wx_i+b-y_i)^2\right)}{\partial b}=0 \end{array}\right.$$

$$\left\{\begin{array}{l} \sum_{i=1}^{N}(wx_i+b-y_i)x_i=0 \\ \\ \sum_{i=1}^{N}(wx_i+b-y_i)=0 \end{array}\right.$$

Chain rule of calculus

$$\boxed{\begin{array}{c} y=g(x) \\ z=f(y)=f\big(g(x)\big) \\ \dfrac{\partial z}{\partial x}=\dfrac{\partial z}{\partial y}\dfrac{\partial y}{\partial x} \end{array}}$$

$$\boxed{\begin{aligned} & z_i=wx_i+b-y_i \\ & \frac{\partial {z_i}^2}{\partial w}=\frac{\partial {z_i}^2}{\partial z_i}\frac{\partial z_i}{\partial w} \\ & =2z_i\frac{\partial(wx_i+b-y_i)}{\partial w} \\ & =2z_ix_i \end{aligned}}$$

Closed-form Solution (cont.)

$$\begin{cases} \sum_{i=1}^{N}(wx_i + b - y_i)x_i = 0 \\ \sum_{i=1}^{N}(wx_i + b - y_i) = 0 \end{cases}$$

$$\sum_{i=1}^{N}(wx_i - y_i) + Nb = 0$$

$$\begin{cases} \sum_{i=1}^{N}(wx_i + b - y_i)x_i = 0 \\ b = \frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i) \end{cases}$$

Closed-form Solution (cont.)

$$\begin{cases} \sum_{i=1}^{N}(wx_i + b - y_i)x_i = 0 \\ b = \frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i) \end{cases}$$

$$\begin{cases} \sum_{i=1}^{N}(wx_i + b - y_i)x_i = 0 \\ b = \boxed{\frac{1}{N}\sum_{i=1}^{N} y_i} - w\boxed{\frac{1}{N}\sum_{i=1}^{N} x_i} = \boxed{\bar{y}} - w\boxed{\bar{x}} \end{cases}$$

Closed-form Solution (cont.)

$$\begin{tikzcd}
\left\{\begin{array}{l} \displaystyle\sum_{i=1}^{N}(wx_i + b - y_i)x_i = 0 \\[2em] b = \bar{y} - w\bar{x} \end{array}\right. \arrow[d, bend right] \\
\left\{\begin{array}{l} \displaystyle\sum_{i=1}^{N}(wx_i + \bar{y} - w\bar{x} - y_i)x_i = 0 \\[2em] b = \bar{y} - w\bar{x} \end{array}\right.
\end{tikzcd}$$

Closed-form Solution (cont.)

$$\begin{cases} \sum_{i=1}^{N} (wx_i + \bar{y} - w\bar{x} - y_i)x_i = 0 \\ b = \bar{y} - w\bar{x} \end{cases}$$

$$\Downarrow$$

$$\begin{cases} w\sum_{i=1}^{N} (x_i - \bar{x})x_i = \sum_{i=1}^{N} (y_i - \bar{y})x_i \\ b = \bar{y} - w\bar{x} \end{cases}$$

Closed-form Solution (cont.)

$$\begin{tikzcd}
\begin{cases} w\sum_{i=1}^{N}(x_i-\bar{x})x_i = \sum_{i=1}^{N}(y_i-\bar{y})x_i \\ b = \bar{y}-w\bar{x} \end{cases} \arrow[d, bend right] \\
\begin{cases} w = \dfrac{\sum_{i=1}^{N}(y_i-\bar{y})x_i}{\sum_{i=1}^{N}(x_i-\bar{x})x_i} \\ b = \bar{y}-w\bar{x} \end{cases}
\end{tikzcd}$$

Multi-Dimension Case

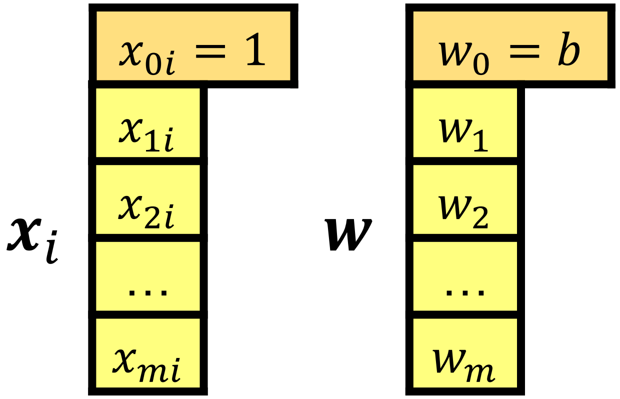
- Each instance has m dimensions, a linear function $f(\boldsymbol{x})$ is defined as

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

- By defining $w_0 = b$, and $x_0 = 1$, $f(\boldsymbol{x})$ can be rewritten as

$$f(\boldsymbol{x}) = \boxed{\boldsymbol{w} \cdot \boldsymbol{x}}$$

Both $\boldsymbol{w}$ and $\boldsymbol{x}$ have $m+1$ dimensions



$$= \sum_{k=0}^{m} x_{ki} w_k$$

$$= \sum_{k=1}^{m} x_{ki} w_k + x_{0i} w_0$$

Optimization

- Learn ***w*** by minimizing the total square loss

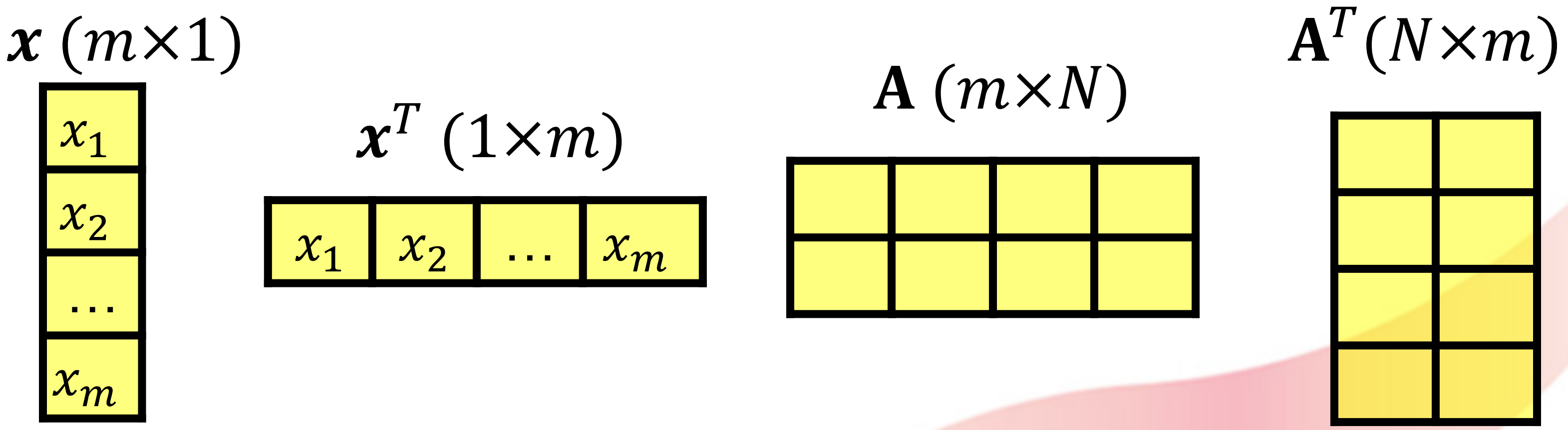
$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2$$

- A closed-form solution can be obtained by setting the derivative of the objective w.r.t. ***w*** to zero, and solving the resultant equations

$$\frac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2\right)}{\partial\boldsymbol{w}} = \mathbf{0}$$

Brief Linear Algebra Review

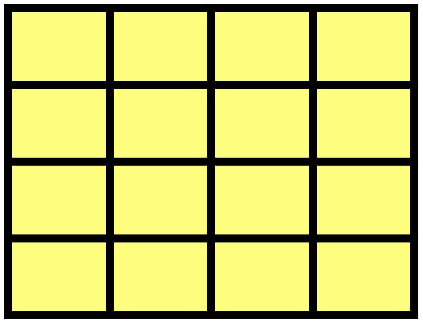
- Linear algebra plays a crucial role in deriving solutions for various machine learning methods
- You are highly recommended to refer to Part I of the Deep Learning book at https://www.deeplearningbook.org/
- Transpose of a vector or matrix


$\boldsymbol{x}\ (m\times 1)$
x_1
x_2
$\dots$
x_m
$\boldsymbol{x}^T\ (1\times m)$
x_1
x_2
$\dots$
x_m
$\mathbf{A}\ (m\times N)$
$\mathbf{A}^T\ (N\times m)$

Matrix/Vector Concepts

- Square matrix
  - If a matrix **A** has the same number of rows and columns, then it is said to be square matrix

$\mathbf{A}\ (m \times m)$



- Symmetric matrix
  - If a **square** matrix **A** satisfies $\mathbf{A} = \mathbf{A}^T$

Matrix Multiplication

- Matrix multiplication is associative
  - $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
- Matrix multiplication is distributive
  - $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$
- Matrix multiplication is NOT commutative in general
  - $\mathbf{AB} \neq \mathbf{BA}$
- The identity matrix, $\mathbf{I}$ $(m \times m)$, is a symmetric matrix with ones on the diagonal and zeros everywhere else
  - If $\mathbf{A}$ is a square matrix $(m \times m)$: $\mathbf{AI} = \mathbf{IA} = \mathbf{A}$
  - If $\mathbf{A}$ is $(N \times m)$: $\mathbf{AI} = \mathbf{A}$
  - If $\mathbf{A}$ is $(m \times N)$: $\mathbf{IA} = \mathbf{A}$

$\mathbf{I}$ $(m \times m)$

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Matrix Operations

- The transpose of $\mathbf{A}^T$:
$$(\mathbf{A}^T)^T = \mathbf{A}$$
- The transpose of $\mathbf{AB}$:
$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$
- The transpose of $\mathbf{A}\boldsymbol{x}$
$$(\mathbf{A}\boldsymbol{x})^T = \boldsymbol{x}^T\mathbf{A}^T$$
- The transpose of $\boldsymbol{x}^T\boldsymbol{y}$
$$(\boldsymbol{x}^T\boldsymbol{y})^T = \boldsymbol{y}^T\boldsymbol{x}$$
- The transpose of a scalar is the scalar itself
$$a^T = a$$

Matrix Operations (cont.)

- For a square matrix $\mathbf{A}$ $(m\times m)$, if it is invertible, then there exists a unique matrix, denoted by $\mathbf{A}^{-1}$, such that
$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$
- If it is not invertible, then such a matrix $\mathbf{A}^{-1}$ does not exist
- Non-square matrices do not have inverses by definition
- Properties of the inverse ($\mathbf{A}$ and $\mathbf{B}$ are invertible)
  - $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
  - $(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
  - $(\mathbf{A}^{-1})^{T} = (\mathbf{A}^{T})^{-1}$

Linear System

- Given the following system of linear equations

$$\begin{gathered} 2x_1 + x_3 = 5 \\ 3x_1 - 4x_2 + 2x_3 = 4 \\ 2x_2 - 3x_3 = -3 \\ -x_1 + 2x_2 - 5x_3 = 1 \end{gathered}$$

- They can be written in a more compact form as

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

$\mathbf{A}$ (4×3)

2	0	1
3	−4	2
0	2	−3
−1	2	−5

$\boldsymbol{x}$ (3×1)

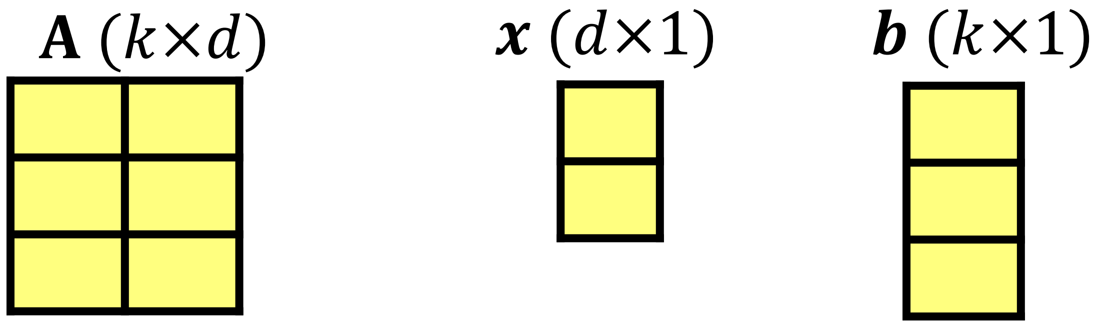
x_1
x_2
x_3

$\boldsymbol{b}$ (4×1)

5
4
−3
1

Linear System (cont.)

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

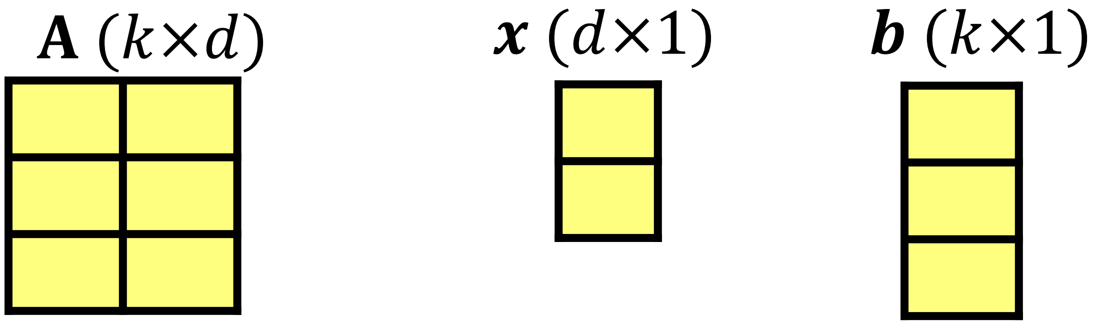

$\mathbf{A}$ $(k \times d)$
$\boldsymbol{x}$ $(d \times 1)$
$\boldsymbol{b}$ $(k \times 1)$

- If **A** is square, and invertible, then we multiply both sides of the equation by $\mathbf{A}^{-1}$ to obtain a **unique** solution

$$\mathbf{A}^{-1}\mathbf{A}\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b} \Longrightarrow \boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

Linear System (cont.)

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$


$\mathbf{A}\ (k\times d)$
$\boldsymbol{x}\ (d\times 1)$
$\boldsymbol{b}\ (k\times 1)$

- If **A** is not invertible, solutions are **not** unique, we can find a solution by using the pseudo inverse (also known as generalized inverse) of **A** instead, denoted by $\mathbf{A}^\dagger$

$$\boldsymbol{x} = \mathbf{A}^\dagger \boldsymbol{b}$$

- Special case: when **A** is square ($k\times k$) but not invertible

$$\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}\mathbf{A}^\dagger = \mathrm{diag}(\lambda_1, \dots, \lambda_k), \text{where } \lambda_i \in \{0,1\}, \text{at least one } \lambda_i = 0$$

$$\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger\mathbf{A}$$

1	0	0	0
0	1	0	0
0	0	**0**	0
0	0	0	**0**

Closed-form Solution for Linear Regression

$$\frac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2\right)}{\partial\boldsymbol{w}} = \mathbf{0}$$

$$\frac{1}{2}\sum_{i=1}^{N}\frac{\partial(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2}{\partial\boldsymbol{w}} = \mathbf{0}$$

$$\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)\boldsymbol{x}_i = \mathbf{0}$$

$$\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i)\boldsymbol{x}_i - \sum_{i=1}^{N}y_i\boldsymbol{x}_i = \mathbf{0}$$

$$\boxed{\begin{aligned} &z_i = \boldsymbol{w}\cdot\boldsymbol{x}_i - y_i \\ &\frac{\partial {z_i}^2}{\partial\boldsymbol{w}} = \frac{\partial {z_i}^2}{\partial z_i}\frac{\partial z_i}{\partial\boldsymbol{w}} = 2z_i\frac{\partial(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)}{\partial\boldsymbol{w}} \\ &\frac{\partial(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)}{\partial\boldsymbol{w}} = \frac{\partial(\boldsymbol{w}\cdot\boldsymbol{x}_i)}{\partial\boldsymbol{w}} - 0 = \boldsymbol{x}_i \end{aligned}}$$

The Matrix Cookbook
https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Closed-form Solution (cont.)

$$a\boldsymbol{x} = \boldsymbol{x}a$$

$$\sum_{i=1}^{N} \overset{\text{scalar}}{\boxed{(\boldsymbol{w} \cdot \boldsymbol{x}_i)}} \boldsymbol{x}_i - \sum_{i=1}^{N} y_i \boldsymbol{x}_i = \mathbf{0}$$

$$\sum_{i=1}^{N} \boxed{\boldsymbol{x}_i (\boldsymbol{w} \cdot \boldsymbol{x}_i)} - \sum_{i=1}^{N} y_i \boldsymbol{x}_i = \mathbf{0}$$

$$\boldsymbol{x}_i (\boldsymbol{w} \cdot \boldsymbol{x}_i) = \boldsymbol{x}_i (\boldsymbol{w}^T \boldsymbol{x}_i) = \boldsymbol{x}_i (\boldsymbol{x}_i^T \boldsymbol{w}) = (\boldsymbol{x}_i \boldsymbol{x}_i^T) \boldsymbol{w}$$

$$\left(\sum_{i=1}^{N} \boxed{(\boldsymbol{x}_i \boldsymbol{x}_i^T)} \right) \boldsymbol{w} - \sum_{i=1}^{N} y_i \boldsymbol{x}_i = \mathbf{0}$$

$\boldsymbol{x}_i \in \mathbb{R}^{(m+1)\times 1}$ and $\boldsymbol{x}_i^T \in \mathbb{R}^{1\times(m+1)}$, thus $\boldsymbol{x}_i \boldsymbol{x}_i^T$ is a $(m+1)$ by $(m+1)$ matrix

Closed-form Solution (cont.)

$$\left(\sum_{i=1}^{N} \boxed{\left(\boldsymbol{x}_i \boldsymbol{x}_i^T\right)}\right)\boldsymbol{w} - \sum_{i=1}^{N} y_i \boldsymbol{x}_i = \boldsymbol{0}$$

$\boldsymbol{x}_i \in \mathbb{R}^{(m+1)\times 1}$ and $\boldsymbol{x}_i^T \in \mathbb{R}^{1\times(m+1)}$, thus
$\boldsymbol{x}_i \boldsymbol{x}_i^T$ is a $(m+1)$ by $(m+1)$ matrix

$(m+1)\times 1$

$$\boldsymbol{x}_i \quad \begin{bmatrix} x_{0i} \\ x_{1i} \\ \ldots \\ x_{mi} \end{bmatrix}$$

$$\boldsymbol{x}_i^T$$

$$\begin{bmatrix} x_{0i} & x_{1i} & \ldots & x_{mi} \end{bmatrix}$$

$1\times(m+1)$

$\boldsymbol{x}_i^T \boldsymbol{x}_i$ Inner product, scalar

$\boldsymbol{x}_i \boldsymbol{x}_i^T$ $(m+1)$ by $(m+1)$ matrix

$$\begin{bmatrix} x_{0i}x_{0i} & x_{0i}x_{1i} & \ldots & x_{0i}x_{mi} \\ x_{1i}x_{0i} & x_{1i}x_{1i} & \ldots & x_{1i}x_{mi} \\ \ldots & \ldots & \ldots & \ldots \\ x_{mi}x_{0i} & x_{mi}x_{1i} & \ldots & x_{mi}x_{mi} \end{bmatrix}$$

Closed-form Solution (cont.)

$$\boxed{\left(\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\right)}\boldsymbol{w} - \sum_{i=1}^{N} y_i\boldsymbol{x}_i = \boldsymbol{0}$$

$$\boldsymbol{x}_i\boldsymbol{x}_i^T$$

$(m+1)$ by $(m+1)$ matrix

$x_{0i}x_{0i}$	$x_{0i}x_{1i}$	$\dots$	$x_{0i}x_{mi}$
$x_{1i}x_{0i}$	$x_{1i}x_{1i}$	$\dots$	$x_{1i}x_{mi}$
$\dots$	$\dots$	$\dots$	$\dots$
$x_{mi}x_{0i}$	$x_{mi}x_{1i}$	$\dots$	$x_{mi}x_{mi}$

$$\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)$$

$\sum_{i=1}^{N} x_{0i}x_{0i}$	$\sum_{i=1}^{N} x_{0i}x_{1i}$	$\dots$	$\sum_{i=1}^{N} x_{0i}x_{mi}$
$\sum_{i=1}^{N} x_{1i}x_{0i}$	$\sum_{i=1}^{N} x_{1i}x_{1i}$	$\dots$	$\sum_{i=1}^{N} x_{1i}x_{mi}$
$\dots$	$\dots$	$\dots$	$\dots$
$\sum_{i=1}^{N} x_{mi}x_{0i}$	$\sum_{i=1}^{N} x_{mi}x_{1i}$	$\dots$	$\sum_{i=1}^{N} x_{mi}x_{mi}$

Closed-form Solution (cont.)

$$\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)$$

$$\begin{bmatrix} \sum_{i=1}^{N} x_{0i}x_{0i} & \sum_{i=1}^{N} x_{0i}x_{1i} & \dots & \sum_{i=1}^{N} x_{0i}x_{mi} \\ \sum_{i=1}^{N} x_{1i}x_{0i} & \sum_{i=1}^{N} x_{1i}x_{1i} & \dots & \sum_{i=1}^{N} x_{1i}x_{mi} \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^{N} x_{mi}x_{0i} & \sum_{i=1}^{N} x_{mi}x_{1i} & \dots & \sum_{i=1}^{N} x_{mi}x_{mi} \end{bmatrix}$$

$$\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T) = \mathbf{X}\mathbf{X}^T$$

$(m+1)$ by N

$$\mathbf{X} = \begin{bmatrix} x_{01} & x_{02} & \dots & x_{0N} \\ x_{11} & x_{12} & \dots & x_{1N} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mN} \end{bmatrix}$$

N by $(m+1)$

$$\mathbf{X}^T = \begin{bmatrix} x_{01} & x_{11} & \dots & x_{m1} \\ x_{02} & x_{12} & \dots & x_{m2} \\ \dots & \dots & \dots & \dots \\ x_{0N} & x_{1N} & \dots & x_{mN} \end{bmatrix}$$

Closed-form Solution (cont.)

$$\left(\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\right)\boldsymbol{w}-\sum_{i=1}^{N}y_i\boldsymbol{x}_i=\mathbf{0}$$

$$\mathbf{X}\mathbf{X}^T\boldsymbol{w}-\boxed{\sum_{i=1}^{N}y_i\boldsymbol{x}_i}=\mathbf{0}$$

$$\mathbf{X}\mathbf{X}^T\boldsymbol{w}-\mathbf{X}\boldsymbol{y}=\mathbf{0}$$

$(m+1)$ by N

$$\mathbf{X}=\begin{array}{|c|c|c|c|}\hline x_{01} & x_{02} & \dots & x_{0N}\\ \hline x_{11} & x_{12} & \dots & x_{1N}\\ \hline \dots & \dots & \dots & \dots\\ \hline x_{m1} & x_{m2} & \dots & x_{mN}\\ \hline\end{array}$$

N by 1

$$\boldsymbol{y}=\begin{array}{|c|}\hline y_1\\ \hline y_2\\ \hline \dots\\ \hline y_N\\ \hline\end{array}$$

$$\sum_{i=1}^{N}y_i\boldsymbol{x}_i=\mathbf{X}\boldsymbol{y}$$

Closed-form Solution (cont.)

$$\mathbf{X}\mathbf{X}^T\boldsymbol{w} - \mathbf{X}\boldsymbol{y} = \mathbf{0}$$

$$\mathbf{X}\mathbf{X}^T\boldsymbol{w} = \mathbf{X}\boldsymbol{y}$$

When $\mathbf{X}\mathbf{X}^T$ is invertible

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

$$\cancel{\mathbf{I}}\boldsymbol{w} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\boldsymbol{y}$$

What if $\mathbf{X}\mathbf{X}^T$ is NOT invertible?

When XX^T is NOT Invertible

- We can use the pseudo inverse (also known as generalized inverse) of $\mathbf{XX}^T$ instead, i.e., $(\mathbf{XX}^T)^\dagger$
- In this case

$$\boldsymbol{w} = (\mathbf{XX}^T)^\dagger \mathbf{X}\boldsymbol{y}$$

Regularized Linear Regression

- Recall structural risk minimization

$$\hat{f} = \arg\min_{f} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i) + \lambda\Omega(f)$$

$\lambda > 0$ tradeoff hyper-parameter

$$\|\boldsymbol{w}\|_2^2 = \boldsymbol{w} \cdot \boldsymbol{w}$$

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w} \cdot \boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2} \boxed{\|\boldsymbol{w}\|_2^2}$$

A regularization term to control the complexity of the model

Also known as Ridge regression

Optimization

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

- Still an unconstrained optimization problem, and the objective is still convex
- A closed-form solution can be obtain by setting the derivative of the objective w.r.t. ***w***, and solving the resultant equation

$$\frac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2\right)}{\partial\boldsymbol{w}} = \mathbf{0}$$

Closed-form Solution

$$\frac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2\right)}{\partial \boldsymbol{w}} = \boldsymbol{0}$$

$$\boxed{\frac{\partial\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2}{\partial \boldsymbol{w}}} + \boxed{\frac{\partial\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2}{\partial \boldsymbol{w}}} = \boldsymbol{0}$$

$$\frac{\partial\|\boldsymbol{w}\|_2^2}{\partial \boldsymbol{w}} = \frac{\partial(\boldsymbol{w}\cdot\boldsymbol{w})}{\partial \boldsymbol{w}} = 2\boldsymbol{w}$$

$$\boxed{\left(\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\right)\boldsymbol{w} - \sum_{i=1}^{N} y_i\boldsymbol{x}_i} + \boxed{\lambda\boldsymbol{w}} = \boldsymbol{0}$$

The matrix cookbook

Closed-form Solution (cont.)

$$\begin{tikzcd}
\boxed{\left(\sum_{i=1}^{N}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\right)\boldsymbol{w} - \sum_{i=1}^{N} y_i\boldsymbol{x}_i} + \lambda\boldsymbol{w} = \mathbf{0} \arrow[d, bend right] \\
\boxed{\mathbf{X}\mathbf{X}^T\boldsymbol{w} - \mathbf{X}\boldsymbol{y}} + \lambda\mathbf{I}\boldsymbol{w} = \mathbf{0} \arrow[d, bend right] \\
(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{w} - \mathbf{X}\boldsymbol{y} = \mathbf{0} \arrow[d, bend right] \\
\boxed{(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})}\boldsymbol{w} = \mathbf{X}\boldsymbol{y} \arrow[r] & \boldsymbol{w} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{X}\boldsymbol{y}
\end{tikzcd}$$

Always invertible as
long as λ is positive

Why $\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}$ Invertible?

- A square matrix is invertible if and only if it does not have a zero eigenvalue
- If a symmetric matrix **A** is positive semidefinite, then all of its eigenvalues are non-negative (≥ 0)
  - When a symmetric matrix **A** ($d \times d$) is said to be positive semidefinite iif for any non-zero column vector $\boldsymbol{x}$ ($d \times 1$), $\boldsymbol{x}^T \mathbf{A} \boldsymbol{x} \geq 0$
- If a symmetric matrix **A** is positive definite, then all of its eigenvalues are positive (> 0)
  - When a symmetric matrix **A** ($d \times d$) is said to be positive definite iif for any non-zero column vector $\boldsymbol{x}$ ($d \times 1$), $\boldsymbol{x}^T \mathbf{A} \boldsymbol{x} > 0$
- A positive definite matrix is invertible (all eigenvalues > 0, and of course non-zero)

Why $\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}$ Invertible? (cont.)

- $\mathbf{X}\mathbf{X}^T$ is a positive semidefinite
    - Need to prove for any non-zero $(m+1)$- dimensional column vector $\boldsymbol{z}$, $\boldsymbol{z}^T\mathbf{X}\mathbf{X}^T\boldsymbol{z} \geq 0$
    - Proof: Denote $\boldsymbol{y} = \mathbf{X}^T\boldsymbol{z}$. Then $\boldsymbol{z}^T\mathbf{X}\mathbf{X}^T\boldsymbol{z} = \boxed{\boldsymbol{y}^T\boldsymbol{y}}$ $\|\boldsymbol{y}\|_2^2 \geq 0$
- $\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}$ is positive definite if $\lambda > 0$ $\|\boldsymbol{y}\|_2^2 = 0$ iff $\boldsymbol{y} = \mathbf{X}^T\boldsymbol{z} = \mathbf{0}$
    - Need to prove for any non-zero $(m+1)$- dimensional column vector $\boldsymbol{z}$, $\boldsymbol{z}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{z} = \boldsymbol{z}^T\mathbf{X}\mathbf{X}^T\boldsymbol{z} + \lambda\boldsymbol{z}^T\boldsymbol{z} > 0$
    - Proof: $\boldsymbol{z}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{z} = \boxed{\boldsymbol{z}^T\mathbf{X}\mathbf{X}^T\boldsymbol{z}} + \boxed{\lambda\boldsymbol{z}^T\boldsymbol{z}} > 0$

≥ 0

> 0 since $\lambda > 0$
and $\boldsymbol{z}$ is non-zero

Large-scale Issue

$$(m+1)\times(m+1)$$

$$\boldsymbol{w} = \boxed{(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})}^{-1}\mathbf{X}\boldsymbol{y}$$

- The computation complexity of computing an inverse of a $(m+1)\times(m+1)$ matrix is $O\left((m+1)^3\right)$
- When m is large, it is time consuming
- Rather than computing the inverse to obtain a closed-form solution, consider the following linear system

$$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{w} = \mathbf{X}\boldsymbol{y}$$

Linear system: $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

- We can solve it by using various numerical methods, e.g., Gaussian elimination, etc.

Large-scale Issue (cont.)

$$\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

- Alternatively, rather than trying to derive an analytical solution, we can apply numerical methods to iteratively minimize the objective, e.g., gradient descent
- Denote the objective by

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

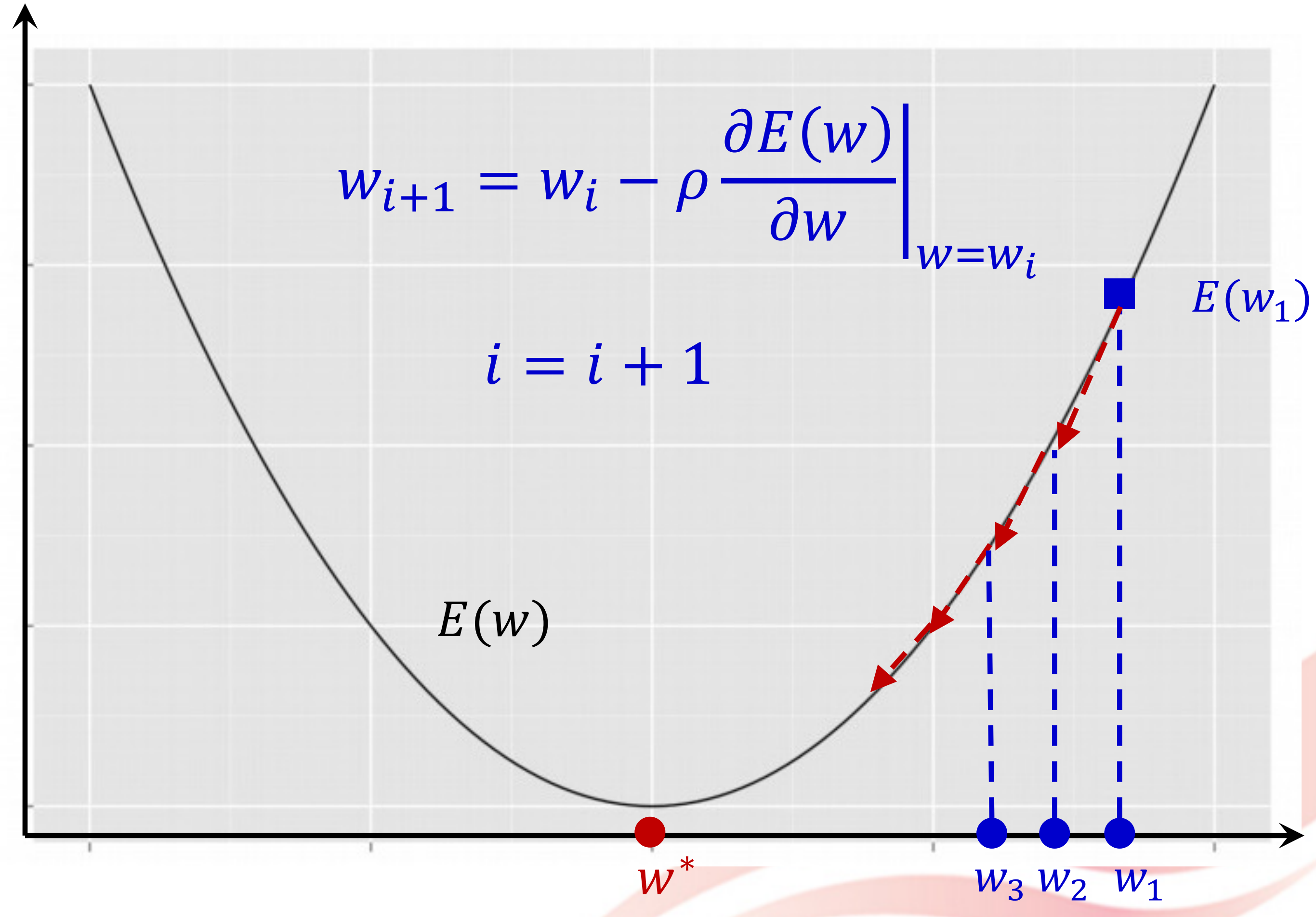
Objective to be minimized

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \boxed{\rho}\,\frac{\partial \boxed{E(\boldsymbol{w})}}{\partial \boldsymbol{w}}$$

Learning rate $\rho \in (0,1]$

Gradient Descent

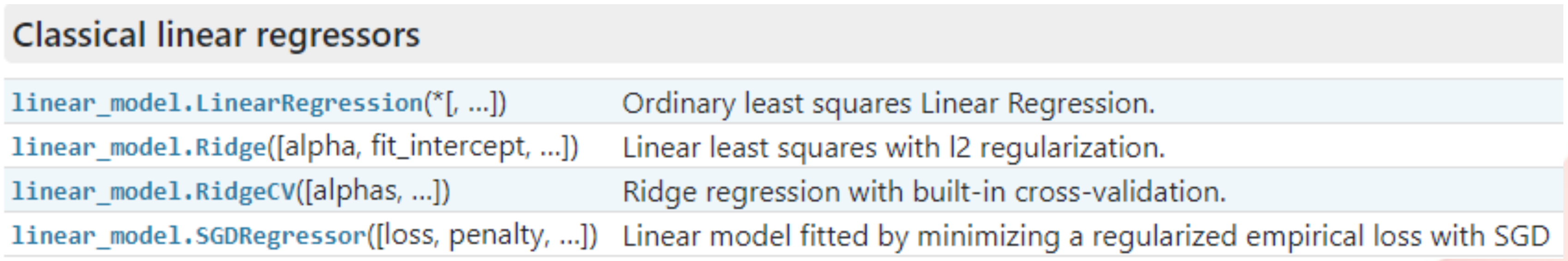
$$w^* = \arg\min_w E(w)$$



$w_{i+1} = w_i - \rho \left.\frac{\partial E(w)}{\partial w}\right|_{w=w_i}$
$i = i + 1$
$E(w_1)$
$E(w)$
w^*
w_3
w_2
w_1
w

Implementation using scikit-learn

- API: sklearn.linear_model: Linear Models
  https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model
  - Classical linear regressors
  - linear_model.LinearRegression → linear regression without regularization
  - linear_model.Ridge → regularized linear regression



Classical linear regressors

linear_model.LinearRegression(*[, ...])	Ordinary least squares Linear Regression.
linear_model.Ridge([alpha, fit_intercept, ...])	Linear least squares with l2 regularization.
linear_model.RidgeCV([alphas, ...])	Ridge regression with built-in cross-validation.
linear_model.SGDRegressor([loss, penalty, ...])	Linear model fitted by minimizing a regularized empirical loss with SGD

Example

```
>>> from sklearn.linear_model import LinearRegression
```

```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np

>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
```

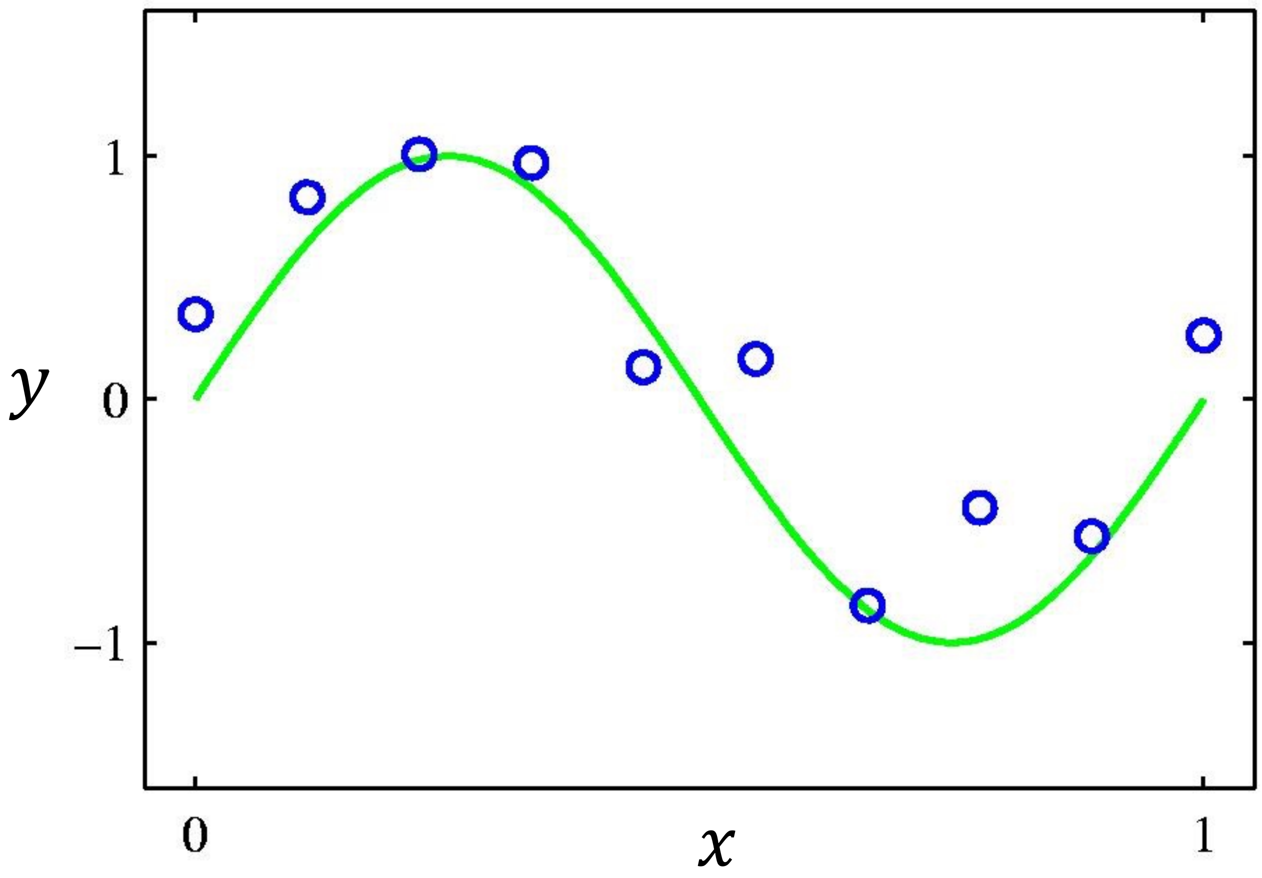
```
>>> rr = Ridge(alpha=0.1)
>>> rr.fit(X, y)
>>> pred_train_rr= rr.predict(X)
```

```
>>> lr = LinearRegression()
>>> lr.fit(X, y)
>>> pred_train_lr= lr.predict(X)
```

Model training and testing

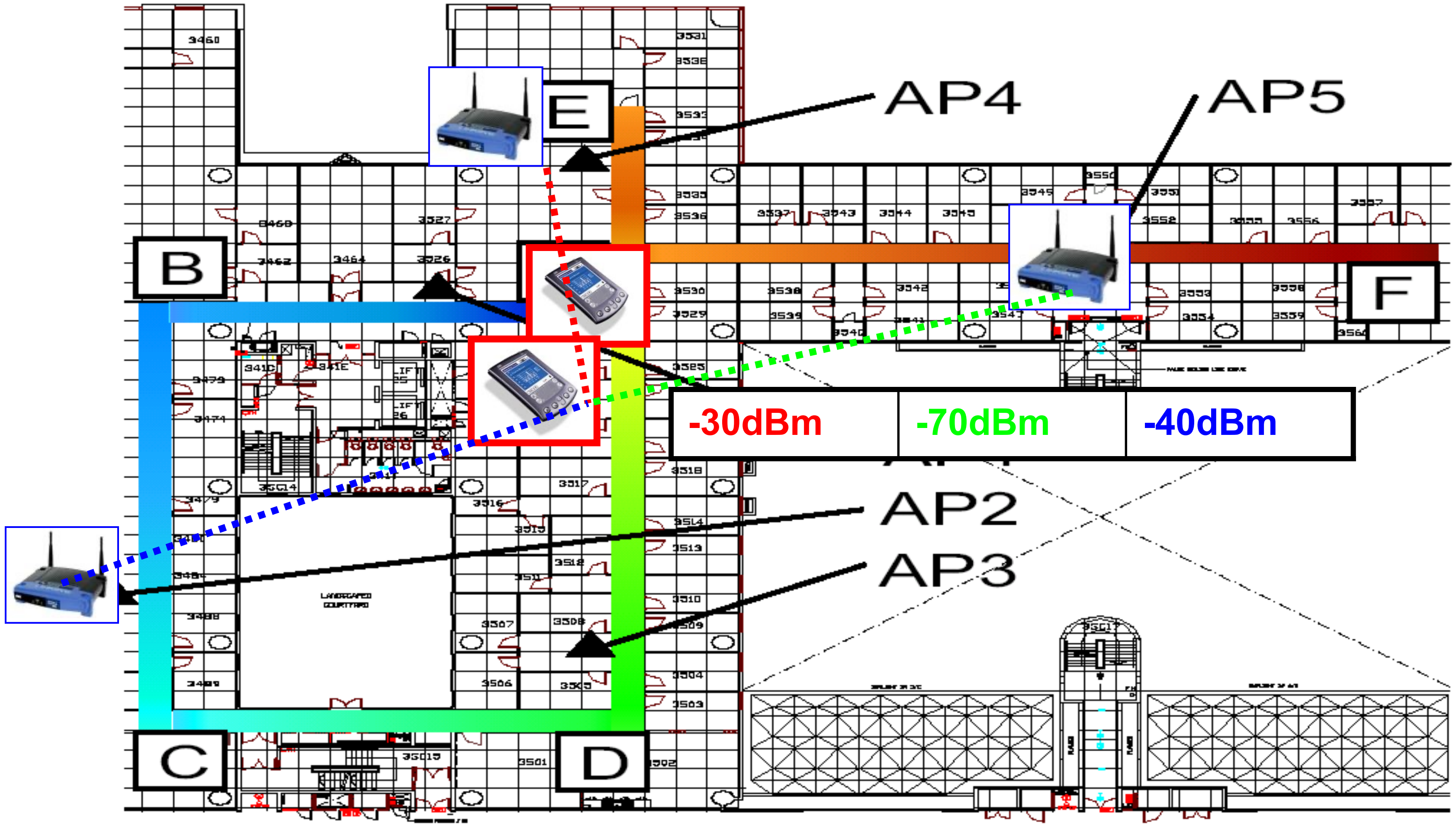
Nonlinear Regression

Kernel methods (L5)



Regression: Real-world Example

Indoor WiFi localization



Thank you!