Regression

Zhiyao Duan

Associate Professor of ECE and CS

University of Rochester

Some figures are copied from the following books

- **LWLS** - Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, *Machine Learning: A First Course for Engineers and Scientists*, Cambridge University Press, 2022.
- **WBK** - Jeremy Watt, Reza Borhani, Aggelos K. Katsaggelos, Machine Learning Refined: Foundations, Algorithms, and Applications (1st Edition), Cambridge University Press, 2016.

Linear Regression

- Regression: given N training examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, learn $f: \boldsymbol{x} \mapsto y$, where $y \in \mathbb{R}$ (numerical), $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ (d-dimensional feature vector)
- This mapping is usually not exact on training data

$$y = f(\boldsymbol{x}) + \epsilon = \hat{y} + \epsilon$$

where ϵ is the approximation error (also called noise)

- Linear Regression: if $f(\cdot)$ is a linear function

$$f(x) = w_0 + w_1 x_1 + \dots + w_d x_d = \begin{bmatrix} w_0 & w_1 & \dots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \boldsymbol{w}^T \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}$$

- **For convenience, we will view x as the expanded feature vector**

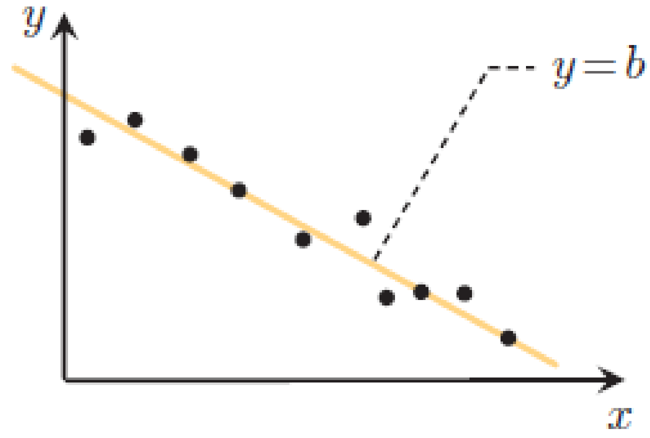
$$\boldsymbol{x} = (x_0, x_1, \dots, x_d)^T, \text{ where } x_0 = 1$$

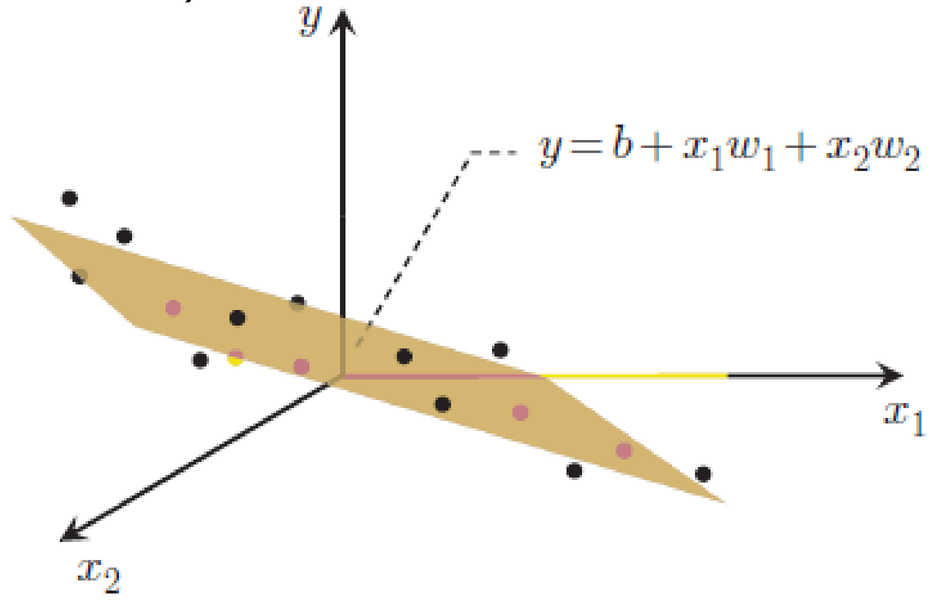
- Then we have $y = \boldsymbol{w}^T \boldsymbol{x} + \epsilon$, and $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$

Geometry of Linear Regression

- Examples when $d = 1$ and $d = 2$

(Fig. 3.1 in WBK 1st edition)





- How to learn $\boldsymbol{w}$ from training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$?
  - Optimization: quantify the approximation error and minimize it

Linear Regression with Squared Error

- On training data, we have $y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)} = \boldsymbol{w}^T \boldsymbol{x}^{(i)} + \epsilon^{(i)}$.
  In matrix form:

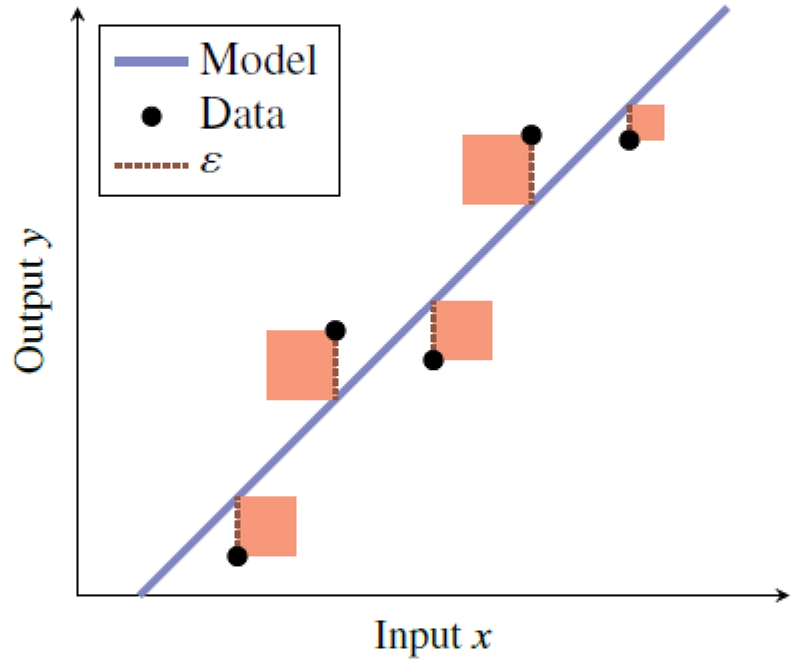
$$\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} {\boldsymbol{x}^{(1)}}^T \\ \vdots \\ {\boldsymbol{x}^{(N)}}^T \end{bmatrix} \boldsymbol{w} + \begin{bmatrix} \epsilon^{(1)} \\ \vdots \\ \epsilon^{(N)} \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

- Define squared error loss:

$$L(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right)^2 = \frac{1}{N} \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w} \|_2^2$$

  – Also called the Least Squares loss



Minimizing Least Squares Loss

$$\min_{\boldsymbol{w}} \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)^2 = \min_{\boldsymbol{w}} \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

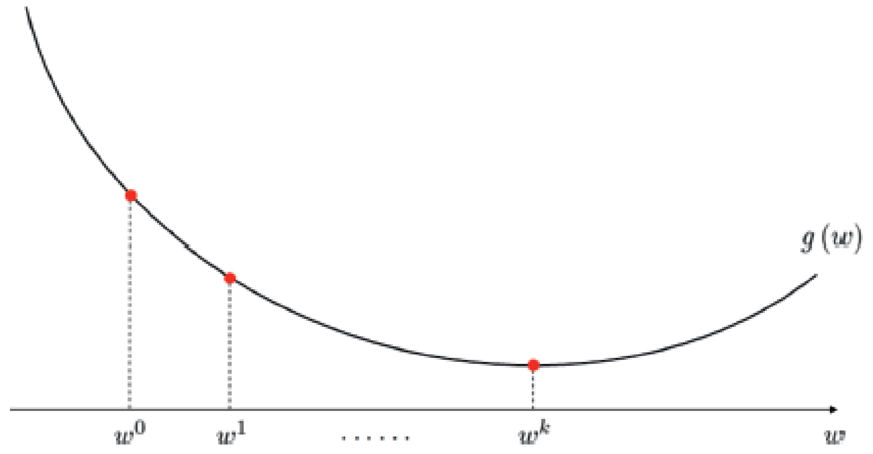
- Computer the gradient w.r.t. $\boldsymbol{w}$:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \frac{2}{N}\sum_{i=1}^{N} -\boldsymbol{x}^{(i)}\left(\mathrm{y}^{(i)} - \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right) = \frac{2}{N}\left(-\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}\right)$$

- Gradient descent: update $\boldsymbol{w}$ along the negative gradient direction

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\nabla_{\boldsymbol{w}} L(\boldsymbol{w})$$

with step size $\eta > 0$, usually small



(Fig. 2.6 in WBK 1st edition)

- Always converges, but may take a long time
  - See Figures 5.4 and 5.5 in WBK

Minimizing Least Squares with Pseudo-inverse

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right)^2 = \min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- This is a quadratic function of $\boldsymbol{w}$
  - Differentiable everywhere, gradient is 0 at a minimum
  - Convex: if there is a minimum, then it is a global minimum

- Let gradient equal to zero, we get the normal equation:

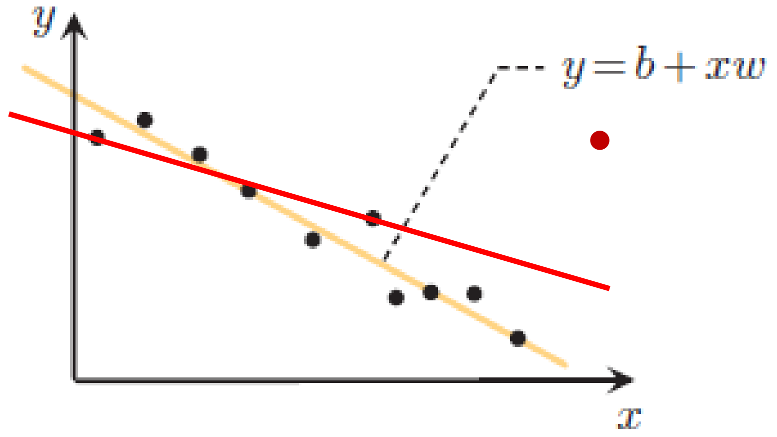
$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \frac{2}{N}(-\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}) = 0$$

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

- If $\boldsymbol{X}^T\boldsymbol{X}$ is invertible (think about when?), then $\boldsymbol{w} = \underbrace{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T}_{\text{Pseudo-inverse of } \boldsymbol{X}}\boldsymbol{y}$

Linear Regression with Absolute Error

- The least squares loss is commonly used, but is susceptible to overfitting outliers (Why?)



(Adapted from Fig. 3.1 in WBK 1st edition)

- Consider the absolute error loss (also called absolute deviations loss)

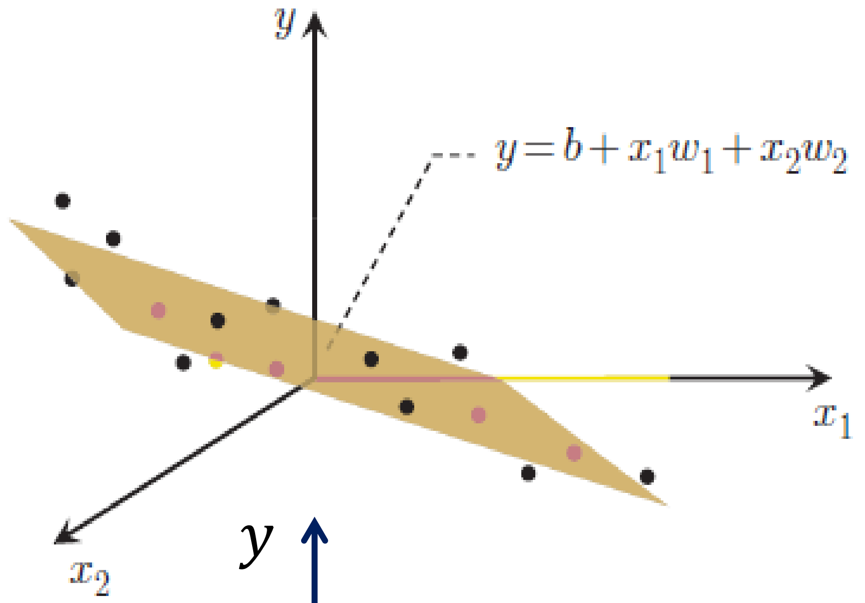
$$L(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\left|y^{(i)} - \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right| = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_1$$

  - Not differentiable at $y^{(i)} - \boldsymbol{w}^T\boldsymbol{x}^{(i)} = 0, \forall i$
  - Second derivative is always 0 if it exists
  - Convex

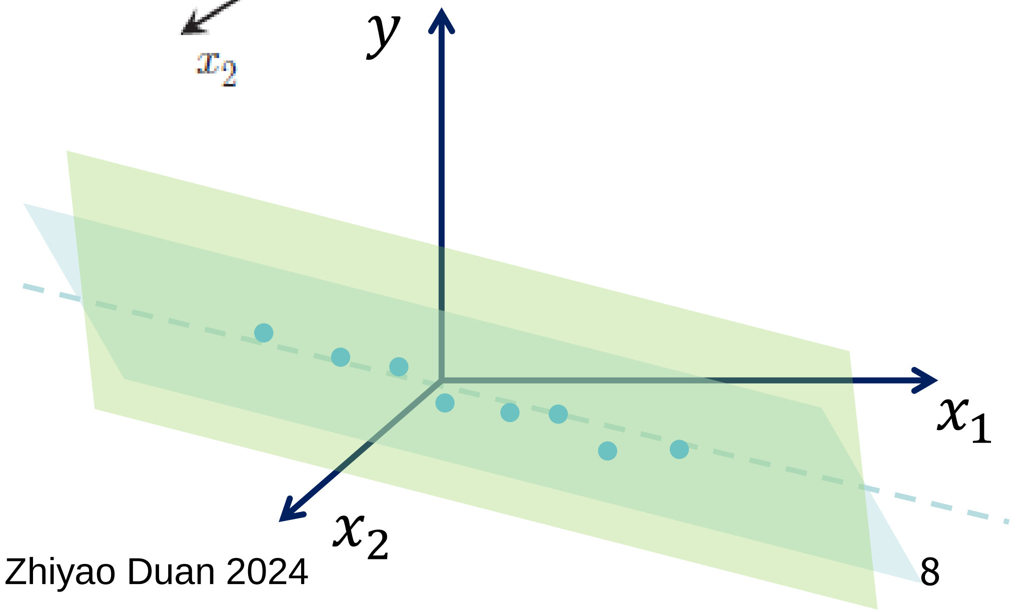
Least Squares → Ridge Regression

- Remember the normal equation for least squares
$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$
- If $N < d + 1$ or dimensions of $\boldsymbol{X}$ are linearly dependent, then $\boldsymbol{X}^T\boldsymbol{X}$ is singular, and there are infinitely many solutions for $\boldsymbol{w}$.
  - Which one to choose?

(Fig. 3.1 in WBK 1st edition)



- Ridge Regression
$$L(\boldsymbol{w}) = \frac{1}{N}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 + \lambda||\boldsymbol{w}||_2^2$$
  - This is L2 regularization
  - Preference inductive bias: prefers small $\boldsymbol{w}$



Ridge Regression

$$\min_{\boldsymbol{w}} \frac{1}{N} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 + \lambda ||\boldsymbol{w}||_2^2$$

- This is still a quadratic function of $\boldsymbol{w}$
- $\lambda \geq 0$ controls the strength of regularization
- Computer gradient w.r.t. $\boldsymbol{w}$ and let it equal zero:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \frac{2}{N}(-\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}) + 2\lambda\boldsymbol{w} = 0$$

- New normal equation:

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

- We can solve $\boldsymbol{w}$:

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

  – The term $N\lambda\boldsymbol{I}$ makes $\boldsymbol{X}^T\boldsymbol{X}$ less singular

- This L2 regularization idea is widely used in many machine learning models

LASSO

- Least Absolute Shrinkage and Selection Operator (LASSO)
  - Uses L1 regularization

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_1$$

- Still a convex function of $\boldsymbol{w}$, but is not differentiable at $w_k = 0, \forall k \in \{0, \cdots, d\}$
- Has no analytical solution as ridge regression does

- Coordinate Descent algorithm
  - Optimize only one parameter w_k in each iteration

- Compared to ridge regression, LASSO results in a sparse vector $\boldsymbol{w}$, where only a few elements are not zero.

Why does L1 regularization creates sparsity?

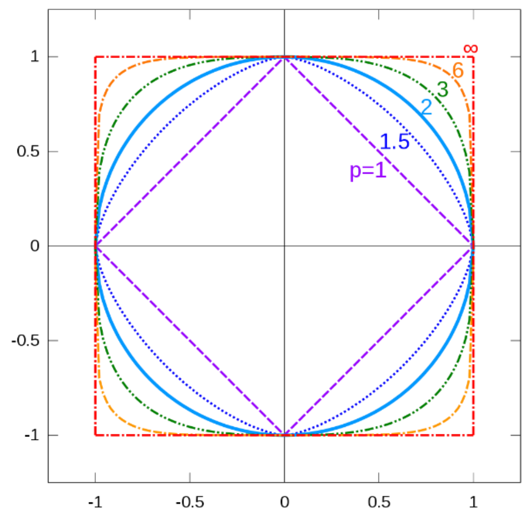
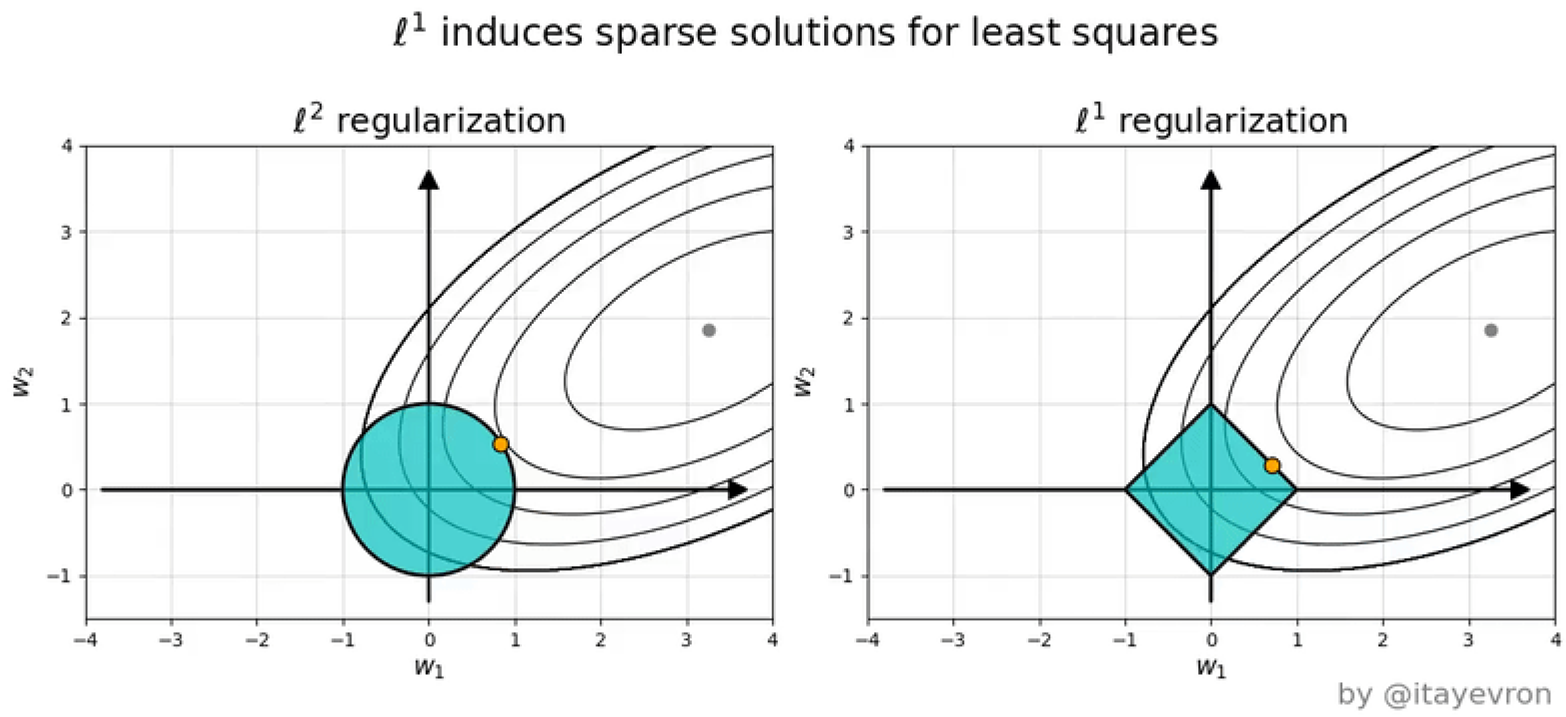


Figure from
https://en.wikipedia.org/wiki/Lp_space



(figure from https://satishkumarmoparthi.medium.com/why-l1-norm-creates-sparsity-compared-with-l2-norm-3c6fa9c607f4)

Ridge Regression vs. LASSO

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$$

- Regularization, prefers small $\boldsymbol{w}$
- Convex

- Has analytic solution
- Solve $\boldsymbol{w}$ through normal equation
- Results in non-sparse $\boldsymbol{w}$
  - Many elements are small but non-zero

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_1$$

- Regularization, prefers small $\boldsymbol{w}$
- Convex

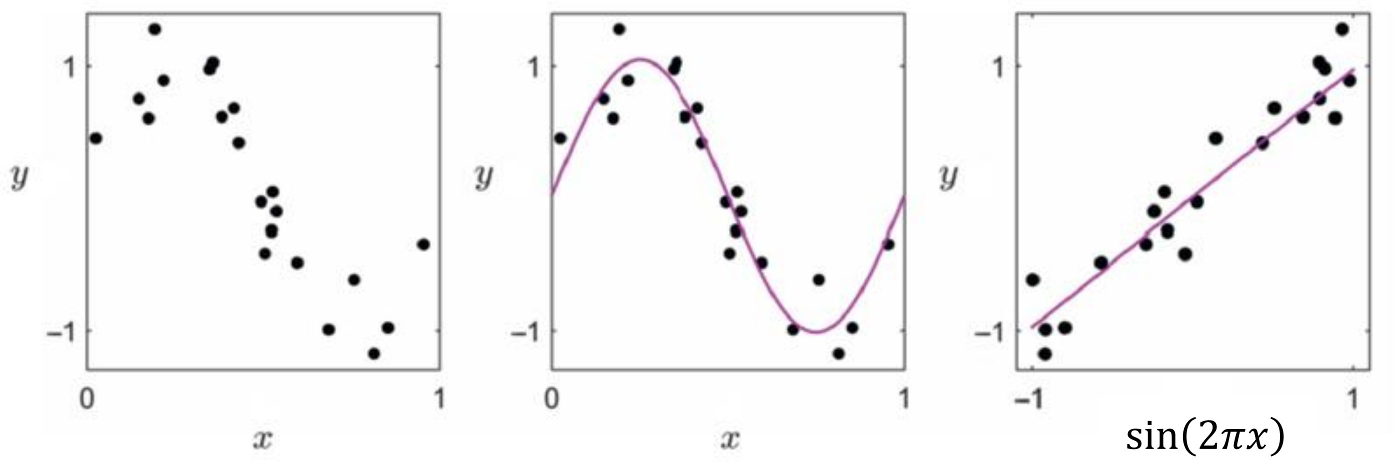
- No analytic solution
- Solve $\boldsymbol{w}$ through coordinate descent
- Results in sparse $\boldsymbol{w}$
  - Many elements are zero

Nonlinear Regression

- In many cases the mapping between $\boldsymbol{x}$ and y is nonlinear
- One may try to derive nonlinear features $\boldsymbol{\phi}(\boldsymbol{x}) = [1, \phi_1(\boldsymbol{x}), \cdots, \phi_p(\boldsymbol{x})]^T$, and then try linear regression from $\boldsymbol{\phi}(\boldsymbol{x})$ to y

$$y = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon$$

- This is nonlinear regression through linear regression and a nonlinear feature mapping



(Fig. 3.7 in WBK 1st edition)

Polynomial Regression

- Polynomial regression: when $\phi_j(\boldsymbol{x})$ is a polynomial of $\boldsymbol{x}$
  - Example in 1-d:

$$\boldsymbol{\phi}(x) = [1, x, \cdots, x^p]^T$$

$$y = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon$$

$$= w_0 + w_1 x + w_2 x^2 + \cdots + w_p x^p + \epsilon$$

- Then solve linear regression with squared error

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2$$

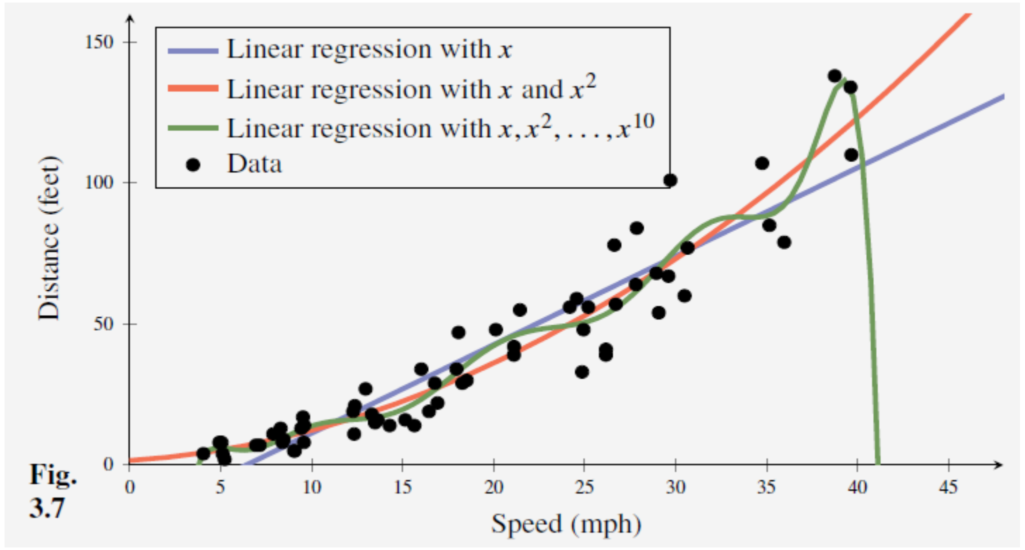


Fig. 3.7

(Fig. 3.7 in LWLS)

- When p is large, it often overfits
- Add regularization to alleviate it
  - Ridge: $\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$
  - LASSO: $\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_1$

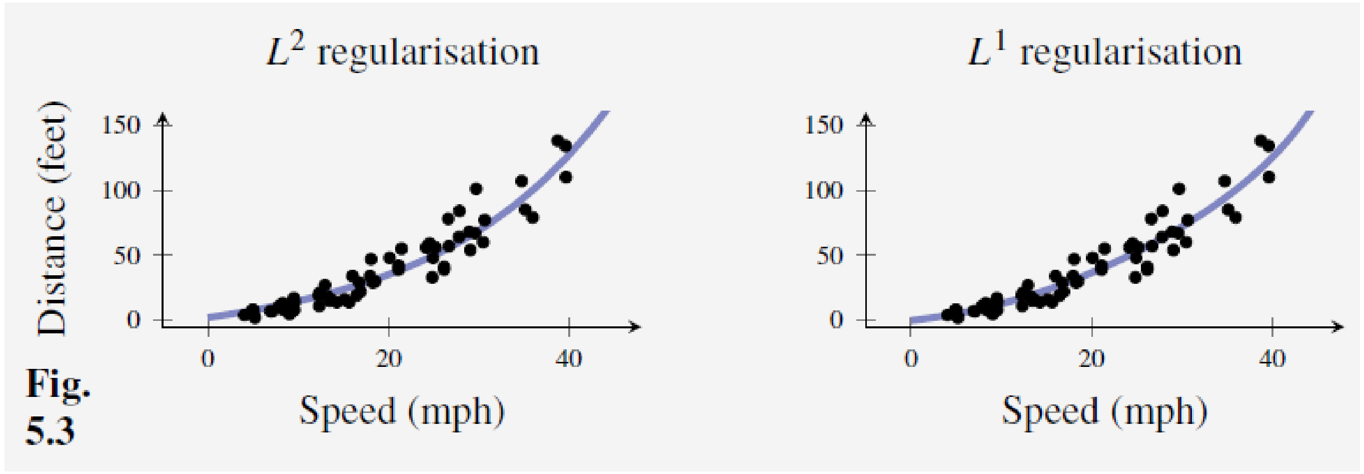


Fig. 5.3

(Fig. 5.3 in LWLS)

Ridge Regression with Nonlinear Features

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$$

original data $\boldsymbol{X} = \begin{bmatrix} {\boldsymbol{x}^{(1)}}^T \\ {\boldsymbol{x}^{(2)}}^T \\ \vdots \\ {\boldsymbol{x}^{(N)}}^T \end{bmatrix}_{N\times(d+1)}$ → nonlinear features $\boldsymbol{\Phi}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{\phi}\left(\boldsymbol{x}^{(1)}\right)^T \\ \boldsymbol{\phi}\left(\boldsymbol{x}^{(2)}\right)^T \\ \vdots \\ \boldsymbol{\phi}\left(\boldsymbol{x}^{(N)}\right)^T \end{bmatrix}_{N\times p}$

- Solving $\boldsymbol{w}$ through normal equation, we have

$$\boldsymbol{w} = \left(\boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{\Phi}(\boldsymbol{X}) + N\lambda \boldsymbol{I}_{p\times p}\right)^{-1} \boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{y}$$

- For test example $\boldsymbol{x}$, use $\boldsymbol{w}$ to predict its y

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{y}^T \boldsymbol{\Phi}(\boldsymbol{X})\left(\boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{\Phi}(\boldsymbol{X}) + N\lambda \boldsymbol{I}_{p\times p}\right)^{-1} \boldsymbol{\phi}(\boldsymbol{x})$$
$$= \boldsymbol{y}^T \left(\boxed{\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T} + N\lambda \boldsymbol{I}_{N\times N}\right)^{-1} \boxed{\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\phi}(\boldsymbol{x})}$$

inner products

- We used the "push through" matrix identity:

$$\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{I})^{-1} = (\boldsymbol{A}\boldsymbol{A}^T + \boldsymbol{I})^{-1}\boldsymbol{A}$$

Kernel Trick

$$\hat{y} = \boldsymbol{y}^T \left(\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T + N\lambda \boldsymbol{I}_{N\times N}\right)^{-1} \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\phi}(\boldsymbol{x})$$

- The feature mapping $\boldsymbol{\phi}(\cdot)$ enters the prediction only through inner products

- For example, in 1-d regression, let $\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ \sqrt{3}x \\ \sqrt{3}x^2 \\ x^3 \end{bmatrix}$, then $\boldsymbol{\phi}(x)^T\boldsymbol{\phi}(x') = (1 + xx')^3$. To make a prediction on a test example, we only need to compute $(1 + xx')^3$ for different x, but not $\boldsymbol{\phi}(x)$

- We could directly define these inner products instead of the nonlinear mapping $\boldsymbol{\phi}(\cdot)$ itself

Kernel Trick

- Kernel: $\kappa(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}$ is a function that takes two samples and return a scalar. We assume
  - Symmetric: $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}', \boldsymbol{x})$
  - Positive semidefinite: a corresponding mapping $\boldsymbol{\phi}(\boldsymbol{x})$ always exists but is not unique! (Reproducing Kernel Hilbert Space, RKHS)
  - Scaling $(a \cdot \kappa(\boldsymbol{x}, \boldsymbol{x}')\ \forall a > 0)$, addition $(\kappa_1(\boldsymbol{x}, \boldsymbol{x}') + \kappa_2(\boldsymbol{x}, \boldsymbol{x}'))$ and multiplication $(\kappa_1(\boldsymbol{x}, \boldsymbol{x}')\kappa_2(\boldsymbol{x}, \boldsymbol{x}'))$ preserve positive semidefiniteness

- E.g., Linear kernel: $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
  - A corresponding mapping: $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$
- E.g., Polynomial kernel: $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (c + \boldsymbol{x}^T \boldsymbol{x}')^{p-1}$
  - A corresponding mapping: finite dimensional function $\boldsymbol{\phi}(\boldsymbol{x})$ where each dimension is a polynomial of $\boldsymbol{x}$ up to order $p - 1$

- E.g., Gaussian Radial Basis Function (RBF) kernel: $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2l^2}\right)$
  - Corresponding mapping $\boldsymbol{\phi}(\boldsymbol{x})$ has infinite dimensions
  - "Local" nature: $\kappa(\boldsymbol{x}, \boldsymbol{x}') \to 0$ as $\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \to \infty$

Kernel Ridge Regression

- Objective: $\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$
- Prediction: $\hat{y} = \boldsymbol{y}^T (\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T + N\lambda \boldsymbol{I}_{N\times N})^{-1} \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\phi}(\boldsymbol{x})$

- Define a kernel function $\kappa(\boldsymbol{x}, \boldsymbol{x}')$
- On training set, we computer kernel between any pair of examples

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) = \begin{bmatrix} \kappa\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}\right) \\ \vdots & \cdots & \vdots \\ \kappa\left(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}\right) \end{bmatrix} = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T \text{(Gram matrix)}$$

- On test example $\boldsymbol{x}$, make prediction as

$$\hat{y} = \underbrace{\boldsymbol{y}^T (\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \mathrm{n}\lambda \boldsymbol{I})^{-1}}_{\text{Constant, can be pre-computed and denoted as } \boldsymbol{\alpha}^T\text{, dual variable}} \boldsymbol{K}(\boldsymbol{X},\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X},\boldsymbol{x})$$

Kernel Ridge Regression

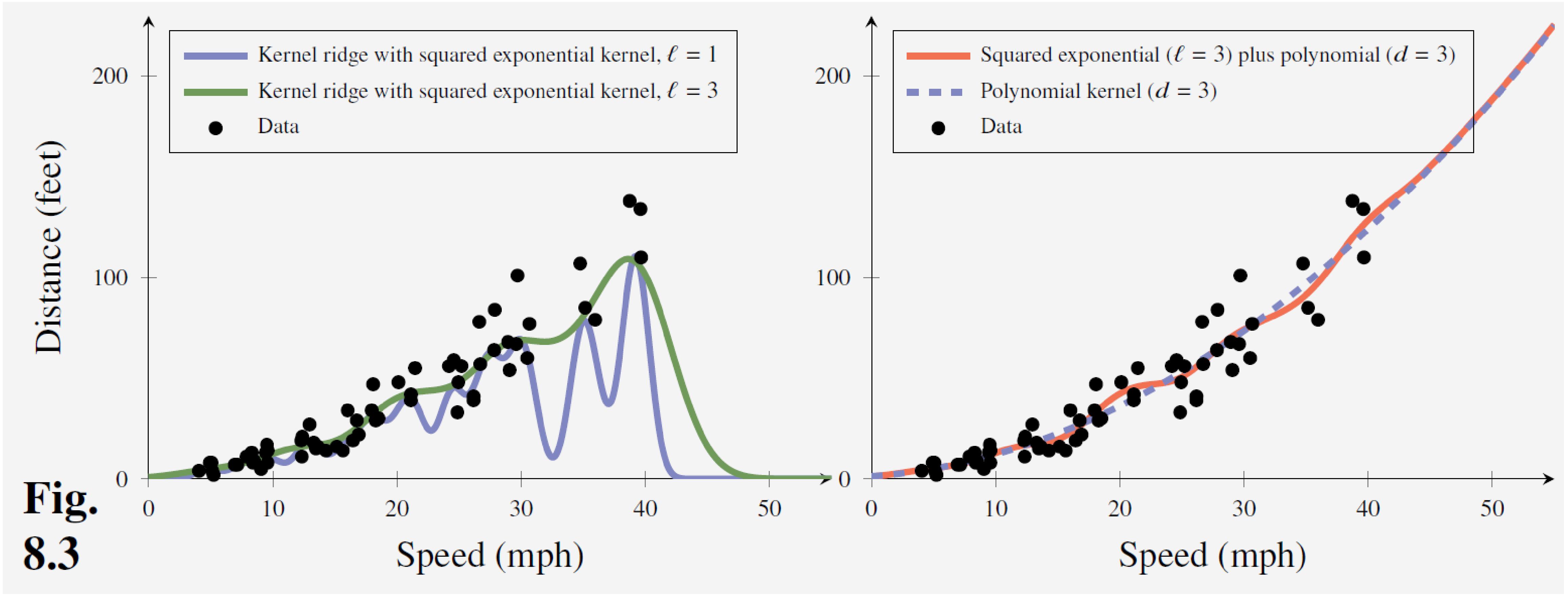


Fig. 8.3

Representer Theorem

- By definition, dual variable

$$\boldsymbol{\alpha} = (\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \mathrm{n}\lambda\boldsymbol{I})^{-1}\boldsymbol{y} = (\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T + \mathrm{n}\lambda\boldsymbol{I})^{-1}\boldsymbol{y}$$

- For Kernel Ridge Regression, previously we have

Primal variable $\longrightarrow$

$$\begin{aligned}\boldsymbol{w} &= \left(\boldsymbol{\Phi}(\boldsymbol{X})^T\boldsymbol{\Phi}(\boldsymbol{X}) + N\lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{\Phi}(\boldsymbol{X})^T\boldsymbol{y} \\ &= \boldsymbol{\Phi}(\boldsymbol{X})^T(\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Phi}(\boldsymbol{X})^T + N\lambda\boldsymbol{I}_{N\times N})^{-1}\boldsymbol{y} \\ &= \boldsymbol{\Phi}(\boldsymbol{X})^T\boldsymbol{\alpha}\end{aligned}$$

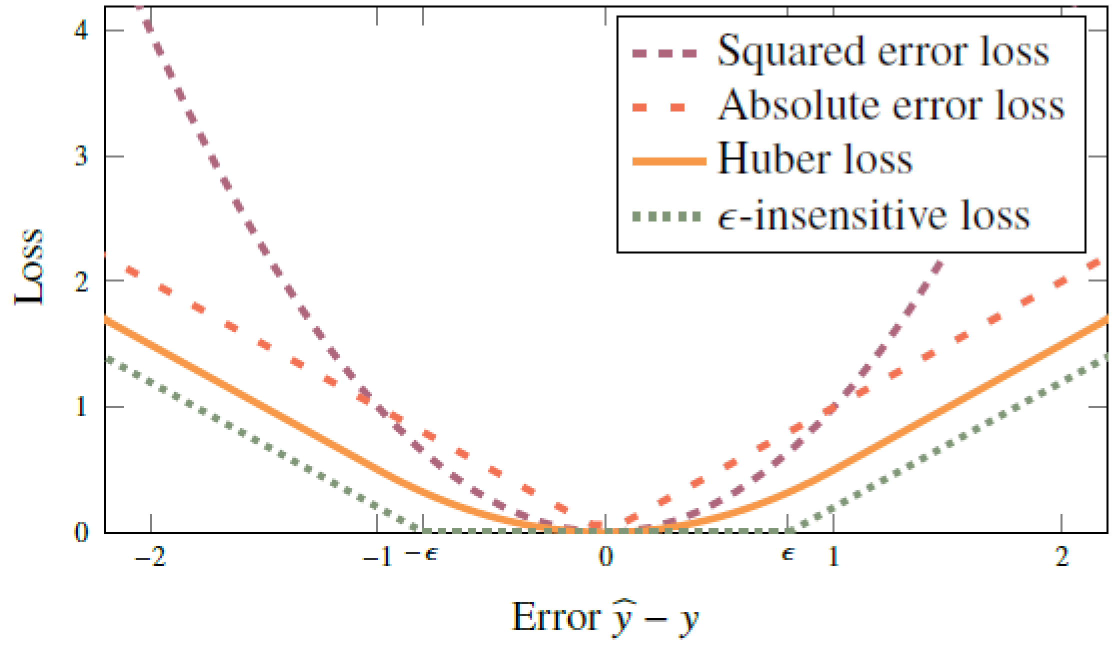
$\longleftarrow$ Dual variable

- This relation between $\boldsymbol{w} \in \mathbb{R}^p$ and $\boldsymbol{\alpha} \in \mathbb{R}^N$, $\boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{X})^T\boldsymbol{\alpha}$, is guaranteed by the representer theorem for (almost) any loss function with L2 regularization
- Thanks to this theorem, we can express a model using dual parameters $\boldsymbol{\alpha}$ and a kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}')$: $\hat{y} = \boldsymbol{\alpha}^T\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x})$, instead of the primal parameters $\boldsymbol{w}$ and a non-linear feature mapping $\boldsymbol{\phi}(\boldsymbol{x})$: $\hat{\boldsymbol{y}} = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})$
  - Note that the dimension of $\boldsymbol{\phi}(\boldsymbol{x})$ is p, which can be very large or even infinite!

Support Vector Regression

- Let's use the ϵ-insensitive loss instead, $\epsilon > 0$, to give some slack

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \max\{0, \left|y^{(i)} - \boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right| - \epsilon\} + \lambda \|\boldsymbol{w}\|_2^2$$



(Fig. 5.1 in LWLS)

Dual Problem

- By the representer theorem, SVR predictions are also of the form
$$\hat{y} = \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x})$$

- And dual variable $\boldsymbol{\alpha}$ is the solution of the dual problem:
$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \boldsymbol{y} + \epsilon \|\boldsymbol{\alpha}\|_1$$
$$\text{subject to } |\alpha_i| \le \frac{1}{2N\lambda}$$
  - No closed-form solution; needs to solve numerically

- $\boldsymbol{\alpha}$ is sparse (Note the L1 regularization term!)
  - This means that the prediction $\hat{y}$ only uses some training examples, but not all!
  - These training examples are called support vectors
  - Note: $\boldsymbol{w}$ is not sparse, as $\boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{\alpha}$

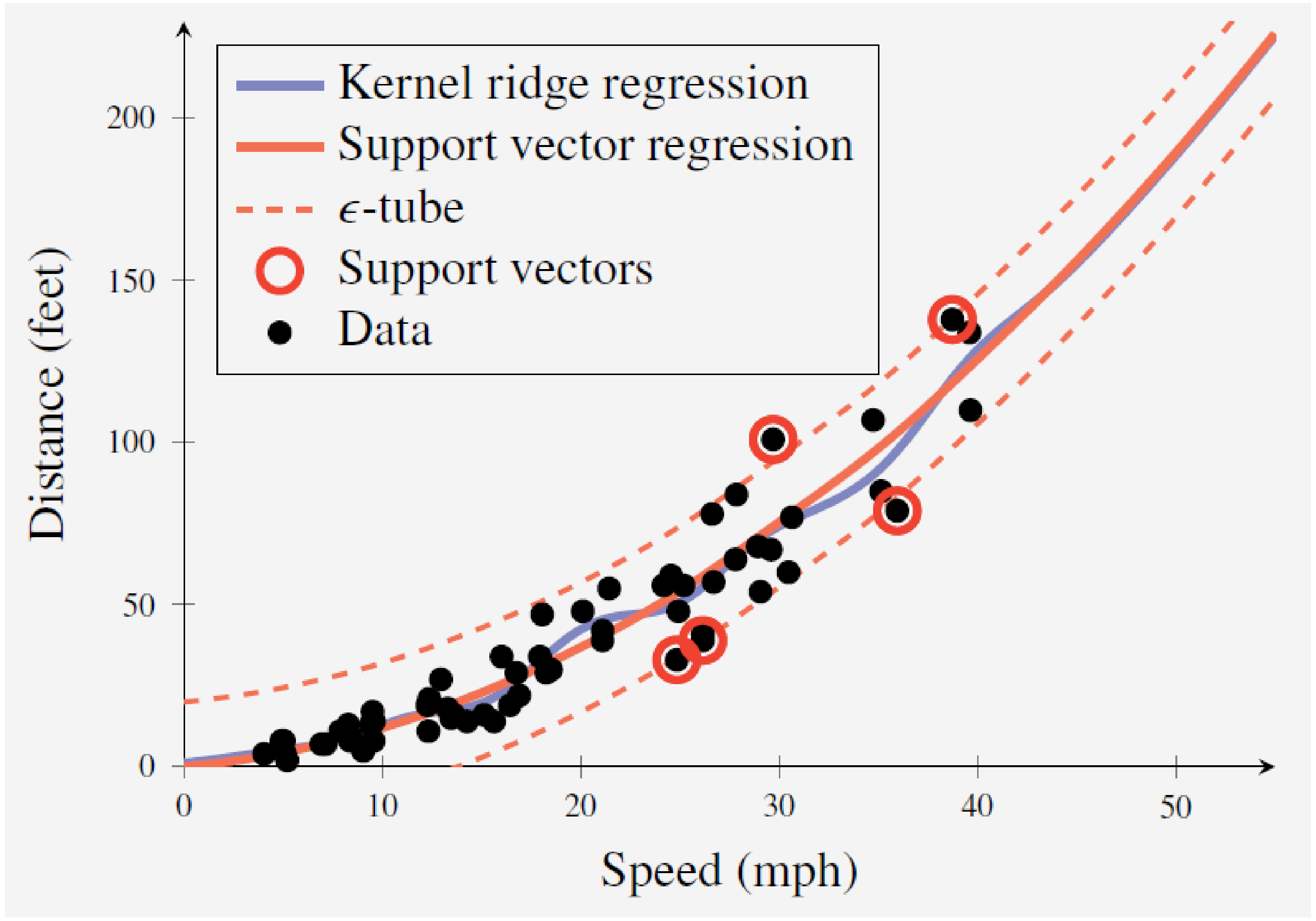
Support Vector Regression

- Support vectors are training examples whose loss is non-zero, i.e.,

$$\left|y^{(i)} - \hat{y}^{(i)}\right| - \epsilon > 0$$

- Larger ϵ → fewer support vectors
  - Less computation during prediction
  - Simpler model, smoother function, stronger regularization

- It is noted that all training examples are used in learning the model (i.e., choosing and weighing support vectors)



Kernel Ridge Regression vs. Support Vector Regression

$$\min_{\boldsymbol{w}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$$

- L2 regularization on primal variable
- Kernel method
- Primal-dual relation: $\boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{\alpha}$
- Solves dual variable $\boldsymbol{\alpha}$ instead of primal variable $\boldsymbol{w}$
- Prediction: $\hat{y} = \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x})$

- Solves $\boldsymbol{\alpha}$ analytically
$$\boldsymbol{\alpha} = (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \mathrm{n}\lambda \boldsymbol{I})^{-1} \boldsymbol{y}$$
- $\boldsymbol{\alpha}$ is not sparse

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} \max\{0, \left|y^{(i)} - \boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right| - \epsilon\} + \lambda \|\boldsymbol{w}\|_2^2$$

- L2 regularization on primal variable
- Kernel method
- Primal-dual relation: $\boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{X})^T \boldsymbol{\alpha}$
- Solves dual variable $\boldsymbol{\alpha}$ instead of primal variable $\boldsymbol{w}$
- Prediction: $\hat{y} = \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x})$

- Solves $\boldsymbol{\alpha}$ numerically
$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \boldsymbol{y} + \epsilon \|\boldsymbol{\alpha}\|_1$$
$$\text{subject to } |\alpha_i| \leq \frac{1}{2N\lambda}$$
- $\boldsymbol{\alpha}$ is sparse

Summary

- Linear regression fits training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with a linear mapping $\boldsymbol{w}^T\boldsymbol{x}$
  - One may use different loss (squared error vs. absolute error) to measure the fitting error

- Regularization on the weights $\boldsymbol{w}$ to prefer small weights
  - L2 regularization: ridge regression → non-sparse weights $\boldsymbol{w}$; Solving normal equation
  - L1 regularization: LASSO → sparse weights $\boldsymbol{w}$; Coordinate descent

- Nonlinear regression through nonlinear feature mapping and linear regression
- Realize nonlinear mapping through kernel trick implicitly
- Solving dual variable $\boldsymbol{\alpha}$ instead of primal variable $\boldsymbol{w}$, thanks to representer theorem

- Kernel ridge regression: non-sparse $\boldsymbol{\alpha}$; computing $\boldsymbol{\alpha}$ directly
- Support vector regression: sparse $\boldsymbol{\alpha}$, support vectors; solving $\boldsymbol{\alpha}$ numerically