Lec11 Machine Learning
11-2 Linear Model

Yang Shu

School of Data Science and Engineering

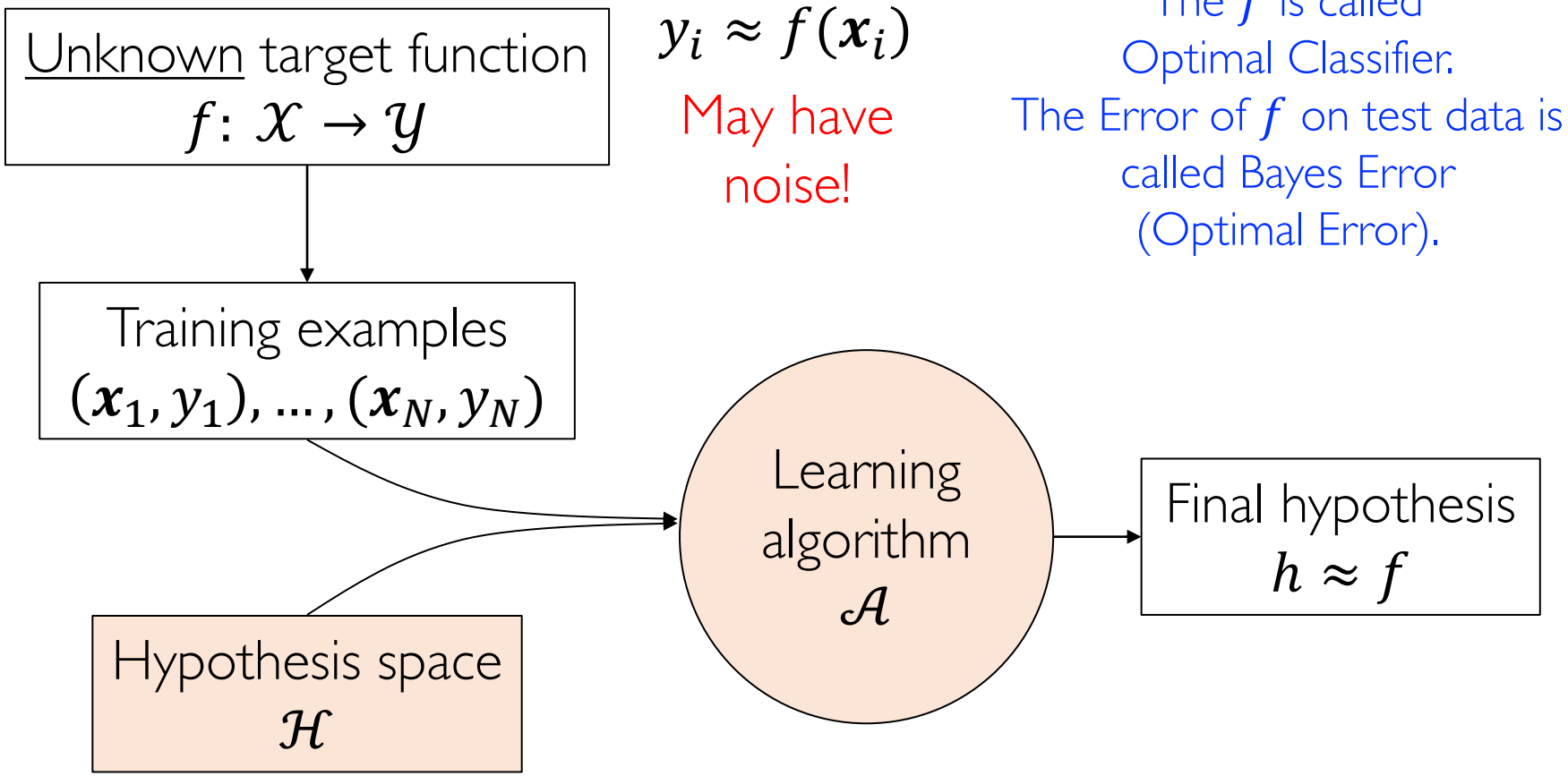
yshu@dase.ecnu.edu.cn

[Acknowledgement: Slides are adapted from Machine Learning Course, Mingsheng Long, THU]

Lec11-2 Linear Model

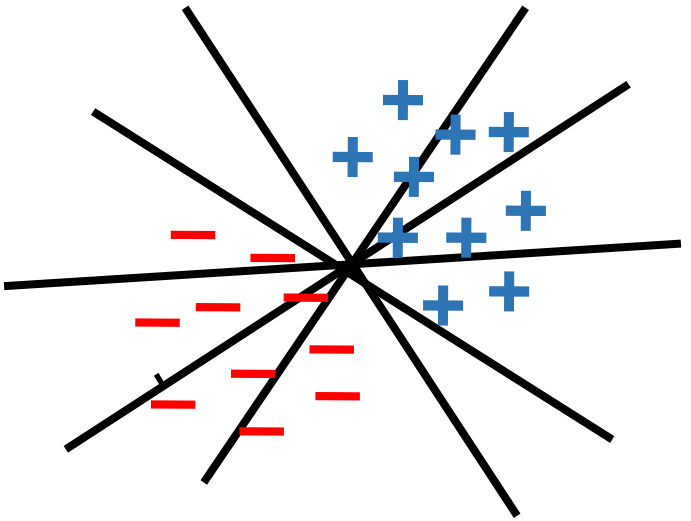
- **Linear Regression**
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- Linear Classification
  - Logistic Regression
  - Softmax Regression

Components of Learning



Hypothesis Space

- A hypothesis space $\mathcal{H}$ is a set of functions that maps $\mathcal{X} \to \mathcal{Y}$.
  - It is the collection of prediction functions we are choosing from.
- We want hypothesis space that…
  - Includes only those functions that have desired regularity (正则性)
    - Continuity (连续性)
    - Smoothness (光滑性)
    - Simplicity (简单性)
  - Easy to work with
- An example hypothesis space:
  - All linear hyperplanes for classification



How to find the best?

Loss Function

- Loss function: $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ measures the difference between a prediction $h(\boldsymbol{x})$ and an actual output y:

$$\ell(y, h(\boldsymbol{x})) = (y - h(\boldsymbol{x}))^2 \text{ (Squared loss for Regression)}$$

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}[y \neq h(\boldsymbol{x})] \text{ (01-loss for Classification)}$$

- The canonical training procedure of machine learning:

Fit dataset with best hypothesis

$$\hat{\epsilon}(h) = \min_{\theta} \sum_{i=1}^{m} \ell(h_\theta(\boldsymbol{x}_i), y_i)$$

θ is parameters of the hypothesis h

- Virtually every machine learning algorithm has this form, just specify
  - What is the hypothesis function?
  - What is the loss function?
  - How do we solve the training problem?

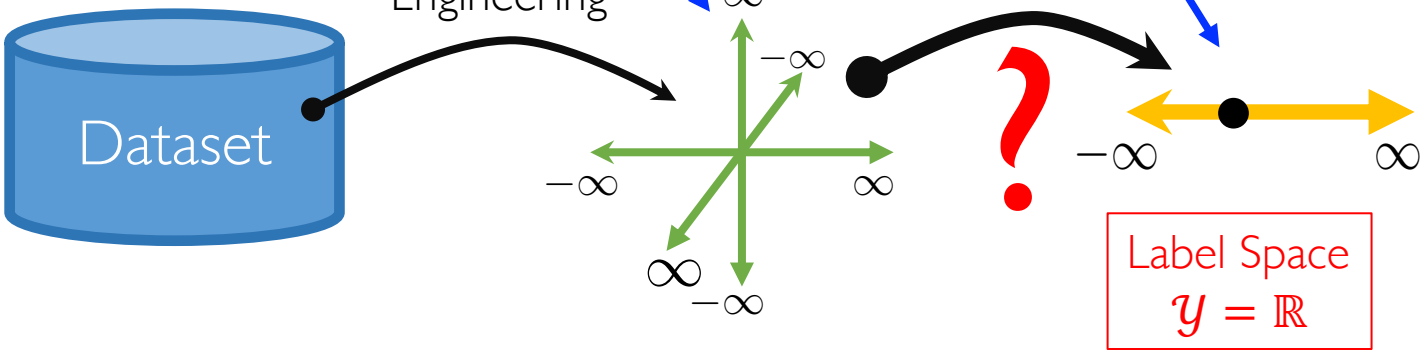
An Example: Electricity Prediction

- **Problem:** Predict the peak power consumption in summer.

Primary key. Cannot be used as the features.

Why?

Date	High Temperature (F)	Peak Demand (GW)
2011-06-01	84.0	2.651
2011-06-02	73.0	2.081
2011-06-03	75.2	1.844
	…	…

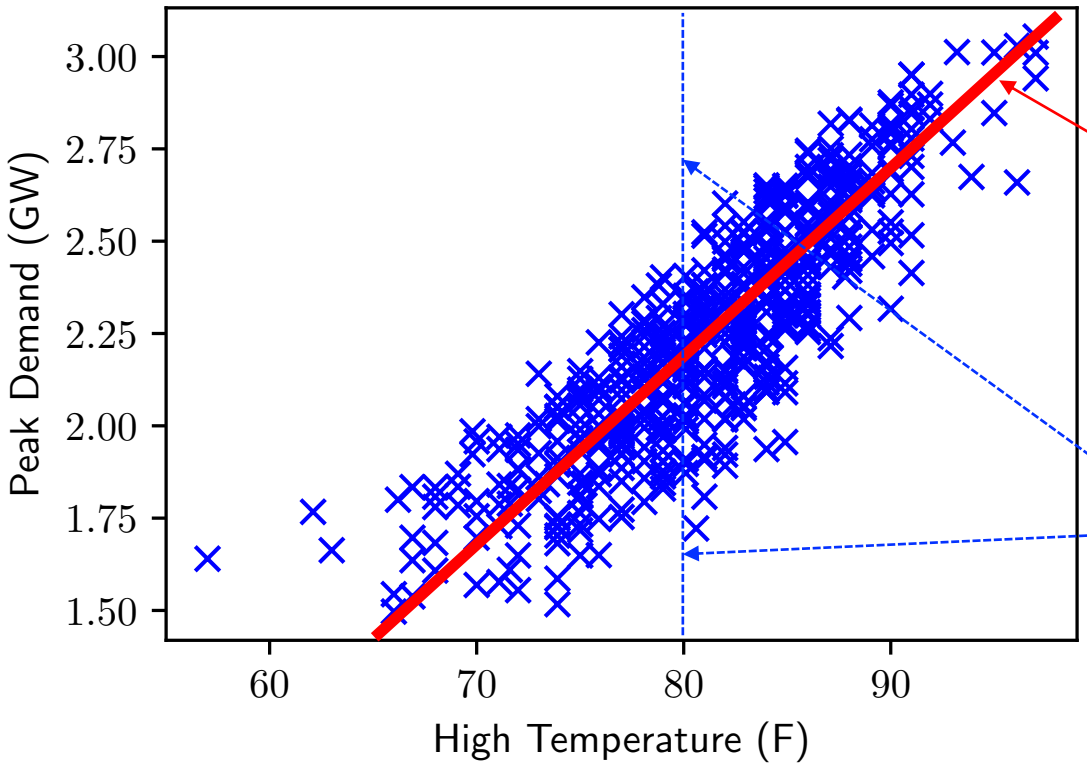


How to map from x to y?

A Regression Problem!

What Model to Choose?

- **Exploratory Data Analysis (EDA)** to ease model selection.
  - Plot *high temperature* vs. *peak demand* in summer (June-August)

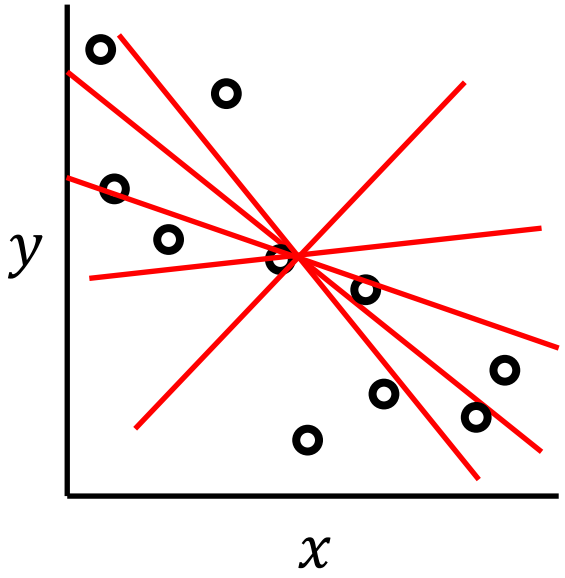


A linear model seems good!

Why different peak demands for the same high temperature?

Linear Regression: Hypothesis Space

- Training data: $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$
  - Feature vector: $\boldsymbol{x} \in \mathbb{R}^d$, response: $y \in \mathbb{R}$
- Add a constant dimension to feature vector $\boldsymbol{x}$:



Equivalent to add intercept in the linear models.

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ \ldots \\ x_d \end{pmatrix} \in \mathbb{R}^{d+1}$$

Each hypothesis (high-dim plane) corresponds to a $\boldsymbol{w}$

- Prediction of hypothesis h parametrized by $\boldsymbol{w}$:

$$h(\boldsymbol{x}) = \underbrace{w_0}_{\text{intercept}} + \sum_{j=1}^{d} w_j x_j = \sum_{j=0}^{d} w_j x_j = \underbrace{\boldsymbol{w}}_{\text{normal vector}} \cdot \boldsymbol{x}$$

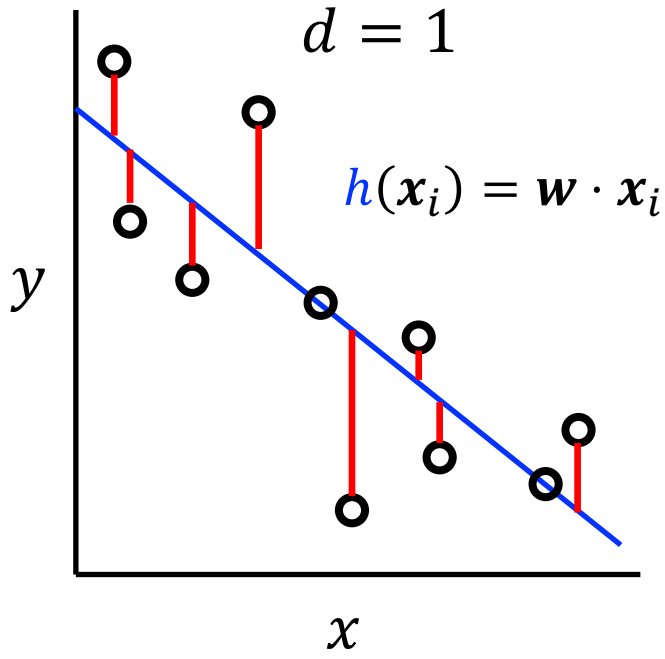
Linear Regression: Loss Function

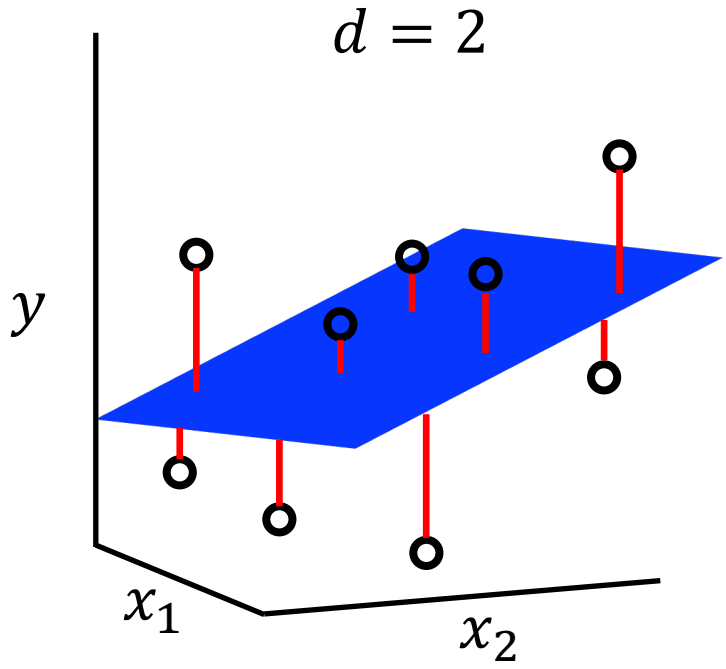
- Use the squared loss (L2 loss) to compute the error on training set:

ˆ standards for *empirical*

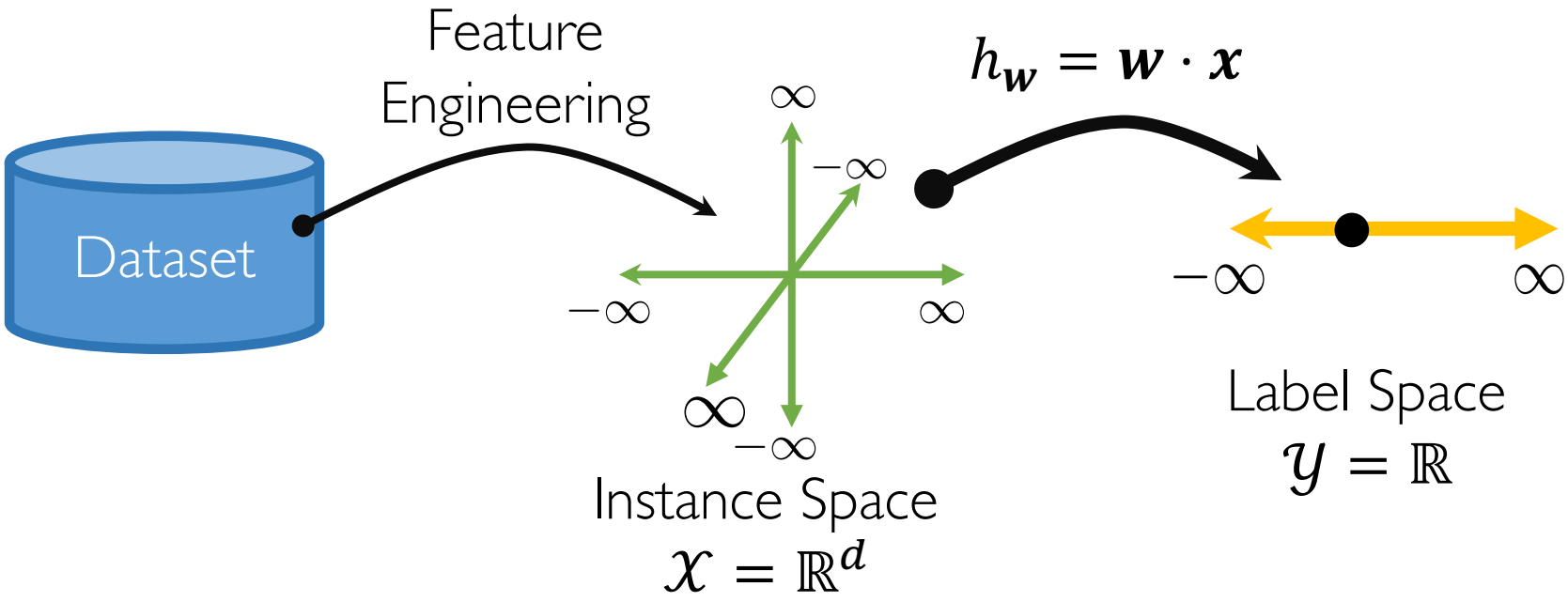
$$\hat{\epsilon}(h) = \sum_{i=1}^{n} (h(\boldsymbol{x}_i) - y_i)^2$$

Training Error





Linear Regression (线性回归)



How do we solve the training problem for the optimal $\boldsymbol{w}$?

Optimization

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

Linear Regression: Analytical Solution

- Training error of linear regression:

$$\hat{\epsilon}(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$
$$= \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

Matrix Cookbook is all you need!

- Put everything in matrix form:

data matrix (design matrix) $n \times (d+1)$

$$\boldsymbol{X} = \begin{bmatrix} - \boldsymbol{x}_1^T - \\ - \boldsymbol{x}_2^T - \\ \vdots \\ - \boldsymbol{x}_n^T - \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

label matrix $n \times 1$

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

Linear Regression: Analytical Solution

- Computing the gradient of $\hat{\epsilon}(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ and setting it to zero yields the optimal parameter $\boldsymbol{w}^*$:

$$\hat{\epsilon}(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

$$\nabla_{\boldsymbol{w}}\hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = \boldsymbol{0}$$

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

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

Normal Equation

In Matrix Cookbook:

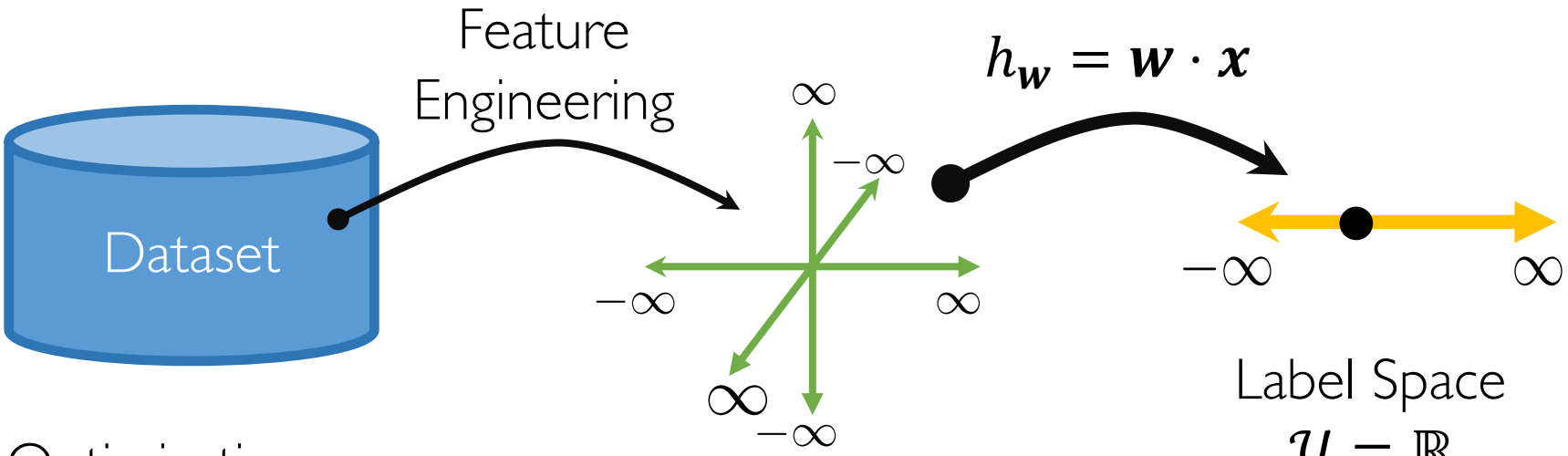
- $\|\boldsymbol{A}\|_F^2 = \mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$
- $\frac{\partial \mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B})}{\partial \boldsymbol{A}} = \boldsymbol{B}$

$$\underbrace{\left(\underbrace{\boldsymbol{X}^T\boldsymbol{X}}_{(d+1)\times(d+1)}\right)^{-1} \times \underbrace{\boldsymbol{X}^T}_{(d+1)\times n}}_{(d+1)\times n}$$

Computational complexity is big!

$$O[d^2(d+n)]$$

Linear Regression



Analytic Solution

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

Unaffordable cost for high-dimensional big data!

Loss Function

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

What should we do in practice?

Outline

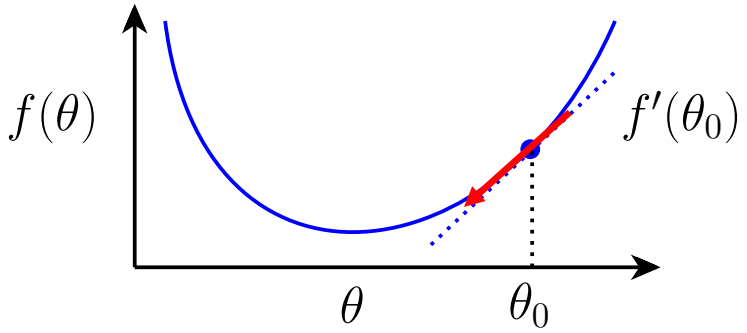
- **Linear Regression**
  - **Optimization**
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- Linear Classification
  - Logistic Regression
  - Softmax Regression

Optimization

- Normal equation suffers from high computing cost.
- Loss functions for other tasks are hard to derive an analytic solution.
- For general differentiable loss function, use Gradient Descent (GD)

Follow the steepest slope

You only need to observe local information for each step.



Work well for convex function

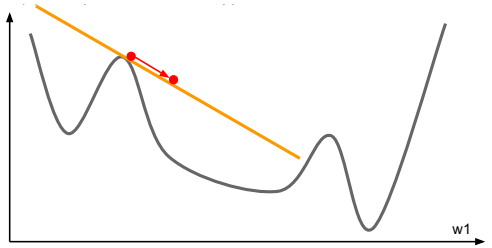
Gradient (梯度)

- Gradient shows direction that function varies fastest.

Same dimension with parameter $\boldsymbol{w}$'s dimension

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

$$g_j = \nabla_{w_j} J(\boldsymbol{w})$$



- First-order Taylor approximation of the objective function:

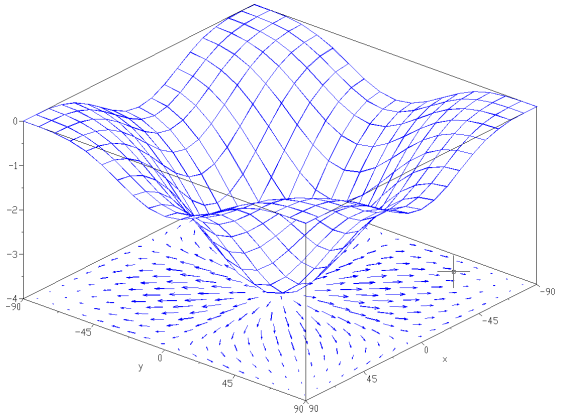
$$J(\boldsymbol{w}) = J(\boldsymbol{w}_0) + (\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{g} + \cdots$$

- Go along gradient for a step with a small rate η:

$$J(\boldsymbol{w} - \eta \boldsymbol{g}) \approx J(\boldsymbol{w}) - \eta \boldsymbol{g}^T \boldsymbol{g}$$

  - Reach a point with smaller loss.

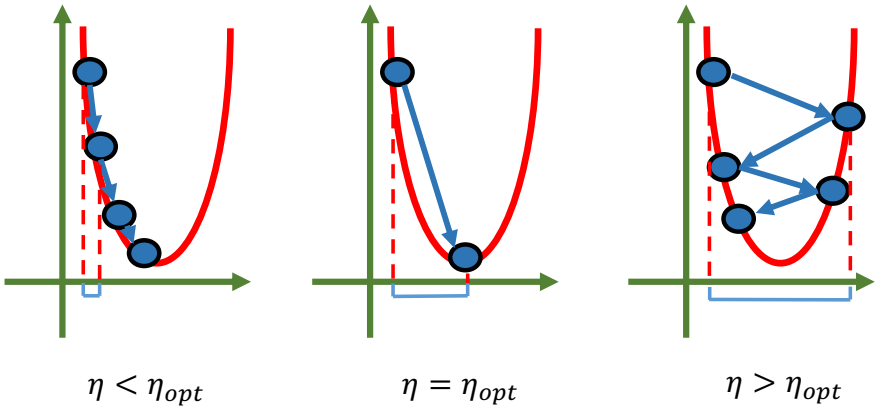
$$-\eta \boldsymbol{g}^T \boldsymbol{g} \leq 0$$



- Repeat this step, and we get the Gradient Descent (GD) algorithm.

Gradient Descent (GD, 梯度下降)

- General optimization algorithm
- For each iteration t ($\leq T$)
  - When $\boldsymbol{w}^t$ does not converge:
  - $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^t}$
- Learning rate η matters!



Squared Loss is convex.

- For linear regression:
  - $\nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$
  - When $\boldsymbol{w}^t$ does not converge:
    - $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - 2\eta \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w}^t - \boldsymbol{y})$

No need for computing inverse matrix!
Complexity (for T iterations): $O(dnT)$

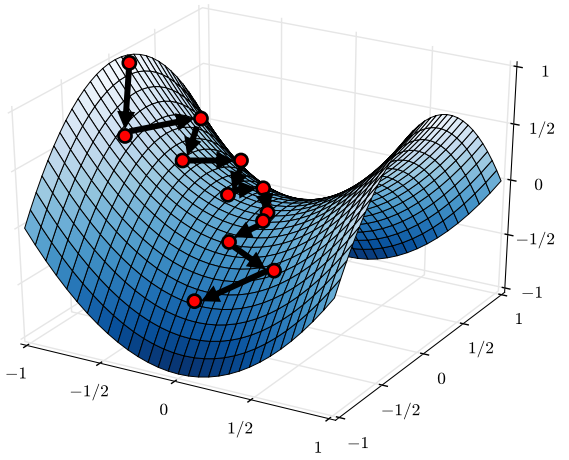
Stochastic Gradient Descent (SGD)

- In each iteration t ($\leq T$):
  - Randomly sample a minibatch of $m \ll n$ points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$
  - Set $J^t(\boldsymbol{w}^t) = \frac{1}{m}\sum_{i=1}^{m} \ell(\boldsymbol{w}^t; \boldsymbol{x}_i, y_i)$
  - Compute gradient on minibatch:

$$\Delta^t = \nabla_{\boldsymbol{w}} J^t(\boldsymbol{w}^t)$$

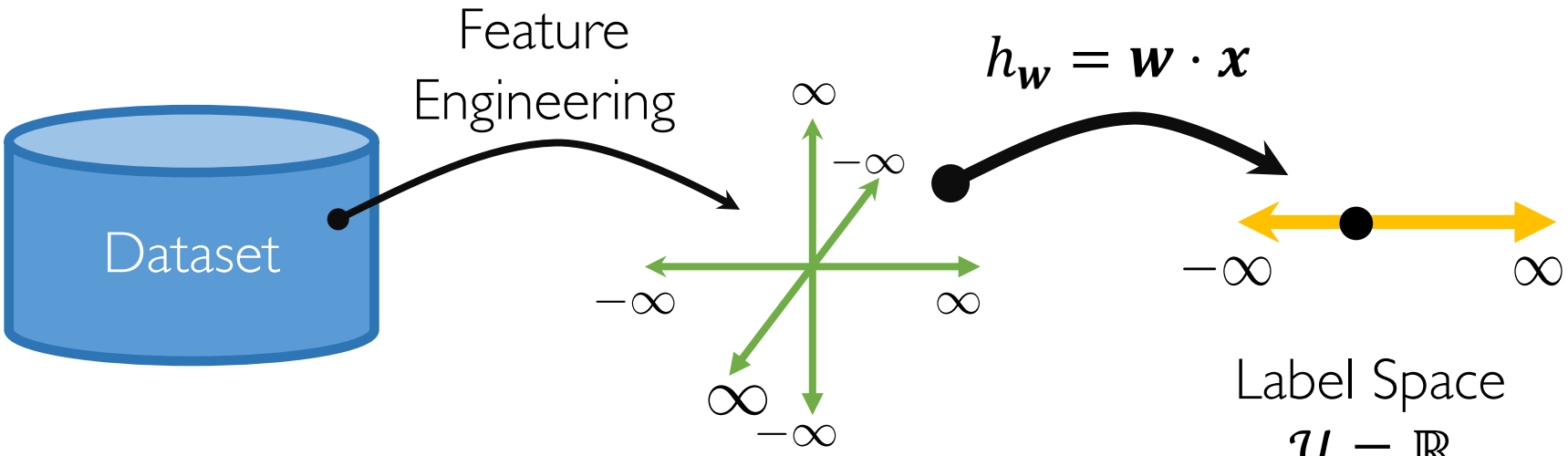
  - Update parameters with learning rate η:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta\Delta^t$$



- For linear regression: $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - 2\eta \boldsymbol{X}_m^T(\boldsymbol{X}_m\boldsymbol{w}^t - \boldsymbol{y})$
  - Computational complexity $O(dmT)$. What if large dimension d?

Linear Regression



Optimization

Gradient Descent

$$\nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Analytic Solution

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

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

Loss Function

Why should we use squared loss?

Outline

- **Linear Regression**
  - Optimization
  - **Statistical View: Maximum Likelihood Estimation**
- Nonlinearization
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- Linear Classification
  - Logistic Regression
  - Softmax Regression

Likelihood in Parametric Models

- Suppose we have a parametric model $\{p(\boldsymbol{z};\theta)|\theta \in \Theta\}$ parametrized by θ and an i.i.d. sample dataset $\mathcal{D} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$
- The likelihood of the estimated parameter $\hat{\theta} \in \Theta$ for sample $\mathcal{D}$ is

$$p(\mathcal{D};\hat{\theta}) \triangleq \prod_{i=1}^{n} p(\boldsymbol{z}_i;\hat{\theta})$$

- Due to numerical stability, we prefer to work with the log-likelihood:

$$\log p(\mathcal{D};\hat{\theta}) \triangleq \sum_{i=1}^{n} \log p(\boldsymbol{z}_i;\hat{\theta})$$

- And sums are easier to work with than products.

Maximum Likelihood Estimation (MLE)

- Maximum Likelihood Estimator (MLE) for estimating the parameter θ in the parametric model $\{p(\boldsymbol{z};\theta)|\theta \in \Theta\}$ is

$$\hat{\theta} \in \underset{\theta\in\Theta}{\operatorname{argmax}} \log p(\mathcal{D};\theta) \Rightarrow \hat{\theta} \in \underset{\theta\in\Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(\boldsymbol{z}_i;\theta)$$

Think of this for all models.

- Finding the MLE is an optimization problem.
- Under some proper distributional assumption:
  - The MLE of a parametric model leads to a particular loss function.
  - Note that: We need to specify the distribution before solving MLE.

Linear Regression: Statistical View

- In the supervised learning model, $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ are i.i.d. samples from some data-generating distribution $p(\boldsymbol{x}, y)$. $\boldsymbol{z}_n$

$$p(\mathcal{D}; \hat{\theta}) \triangleq \prod_{i=1}^{n} p(\boldsymbol{x}_i, y_i; \hat{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i; \hat{\theta}) p(\boldsymbol{x}_i)$$

- For the discriminative model that only models the relation between $\boldsymbol{x}$ and $\boldsymbol{y}$, we assume $\boldsymbol{x}_i$ follows an *empirical* distribution $p(\boldsymbol{x}_i) = \frac{1}{n}$.

$$\prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i; \hat{\theta}) p(\boldsymbol{x}_i) = \prod_{i=1}^{n} \frac{1}{n} p(y_i|\boldsymbol{x}_i; \hat{\theta}) \propto \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i; \hat{\theta})$$

Linear Regression: Statistical View

- In the discriminative model, the likelihood and the log-likelihood of the estimated parameter $\hat{\theta} \in \Theta$ for sample $\mathcal{D}$:

$$p(\mathcal{D}; \theta) \triangleq \prod_{i=1}^{n} p(y_i | \boldsymbol{x}_i; \theta)$$

$$\log p(\mathcal{D}; \theta) = \sum_{i=1}^{n} \log p(y_i | \boldsymbol{x}_i; \theta)$$

- The Maximum Likelihood Estimation (MLE) for discriminative model:

$$\hat{\theta} = \max_{\theta} \sum_{i=1}^{n} \log p(y_i | \boldsymbol{x}_i; \theta)$$

Linear Regression: Statistical View

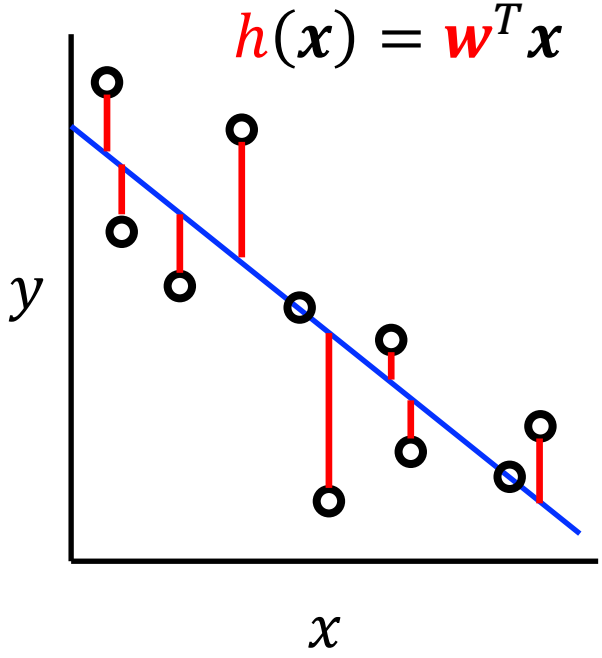
- In linear regression problem, we assume that:

$$y \sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{x}, \sigma^2)$$

$$y = \boldsymbol{w}^T\boldsymbol{x} + e$$

Gaussian noise assumption $\quad e \sim \mathcal{N}(\mathbf{0}, \sigma^2)$

where $\boldsymbol{y}$'s are independent.



- The linear regression should correctly model:

$$\mathbb{E}(y|\boldsymbol{x}; \boldsymbol{w}, \sigma^2) = \boldsymbol{w}^T\boldsymbol{x}$$

Regression function

- We have i.i.d. sampled data $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$.
- We can find the best parameter $\boldsymbol{w}^*$ for sample $\mathcal{D}$ through the MLE.

Linear Regression: Statistical View

- Assume that $y_i|\boldsymbol{w}, \boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{x}_i, \sigma^2)$, then $\theta = \{\boldsymbol{w}, \sigma\}$.
  - For each point $(\boldsymbol{x}_i, y_i)$:

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

- The log-likelihood for linear regression on the whole dataset $\mathcal{D}_n$:

$$\log p(\mathcal{D}_n; \boldsymbol{w}, \sigma) = \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i; \boldsymbol{w}, \sigma)$$

$$= \frac{n}{2}\log\frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$$

Linear Regression: Statistical View

- The Maximum Likelihood Estimation of the optimal parameter $\boldsymbol{w}^*$:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{n}{2} \log \frac{1}{2\pi\sigma^2} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

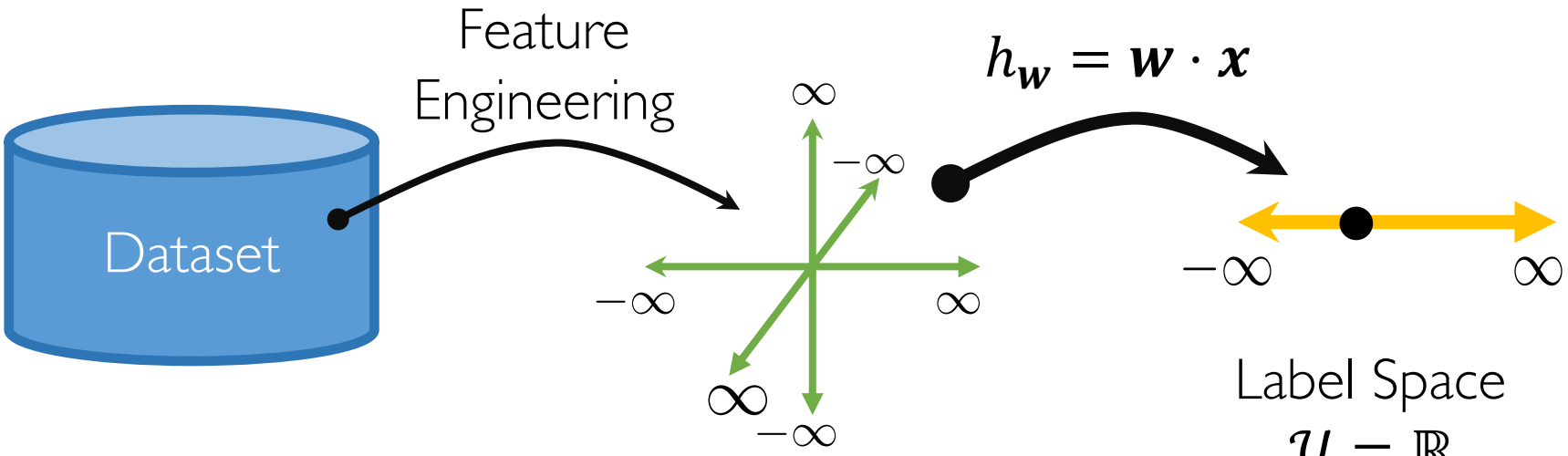
$$= \underset{\boldsymbol{w}}{\operatorname{argmax}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

How about estimating σ?
How about other noise model?

- Note: Maximum Likelihood Estimation of $\boldsymbol{w}$ under the Gaussian noise assumption is equivalent to linear regression with the squared loss

Linear Regression



Optimization

Gradient Descent

$$\nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Analytic Solution

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

Loss Function

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

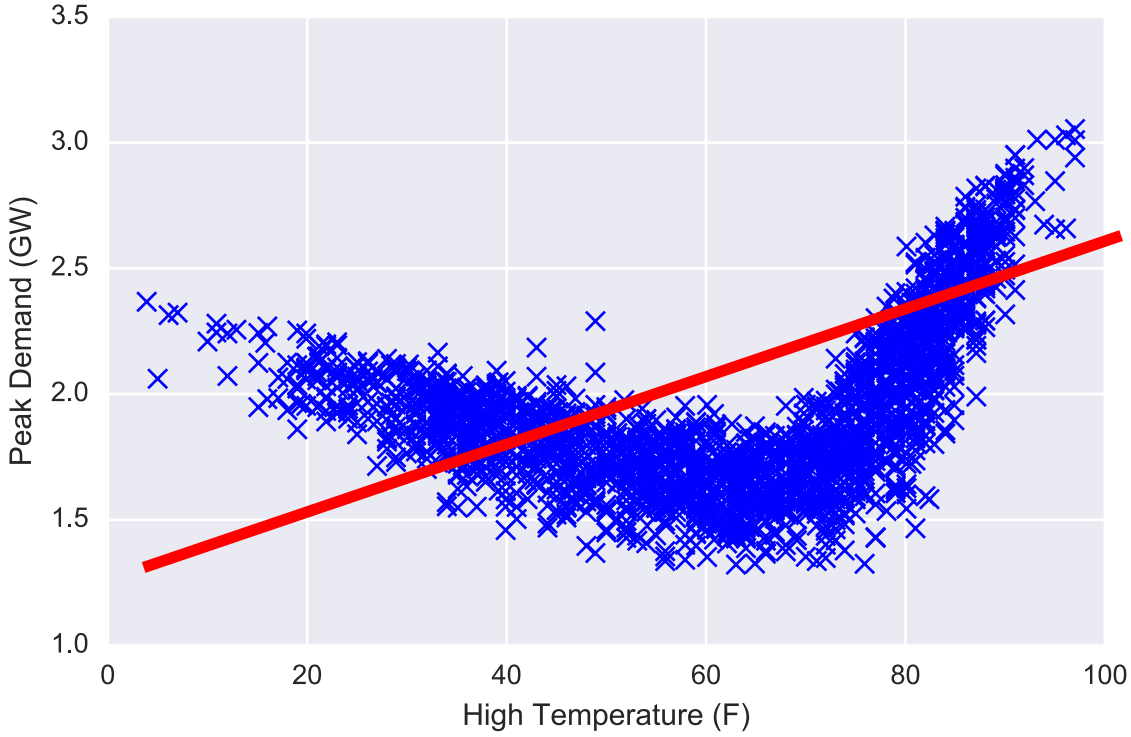
How to handle nonlinearity?

Outline

- Linear Regression
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- **Nonlinearization**
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- Linear Classification
  - Logistic Regression
  - Softmax Regression

An Example: Electricity Prediction

- **Problem:** Predict the peak power consumption in all months.
- **Exploratory Data Analysis (EDA):** *Peak demand* vs. *temperature* plot



Can we use linear regression again?

Underfit!

Basis Function (基函数)

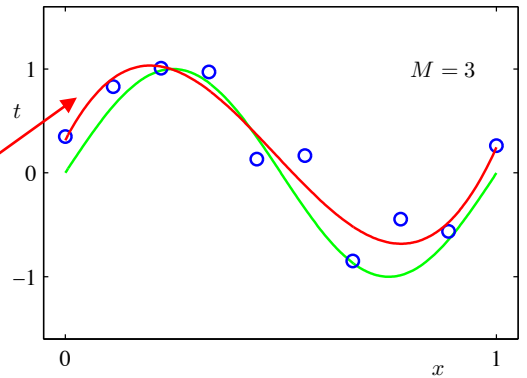
- Feature map: $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X} \xrightarrow{\Phi} \boldsymbol{z} = (z_1, \ldots, z_{\tilde{d}}) \in \mathcal{Z}$

$$\boldsymbol{z} = \Phi(\boldsymbol{x})$$

- Each $z_j = \phi_j(\boldsymbol{x})$ depends on some nonlinear transform ϕ_j.
- $\{\phi_j\}_{0 \le j \le \tilde{d}}$ is called basis functions.
- Polynomial basis functions

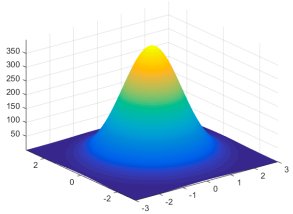
1-D vector: $\boldsymbol{z}' = (1, x_1, x_1^2, x_1^3)$

2-D vector: $\boldsymbol{z}' = (1, x_2, x_1 x_2, x_1^2)$



- Radial basis functions (RBF)

$$\phi_j(\boldsymbol{x}) = \left\{ \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_j||_2^2}{2\sigma^2} \right) : j = 1, \ldots, \tilde{d} \right\}$$

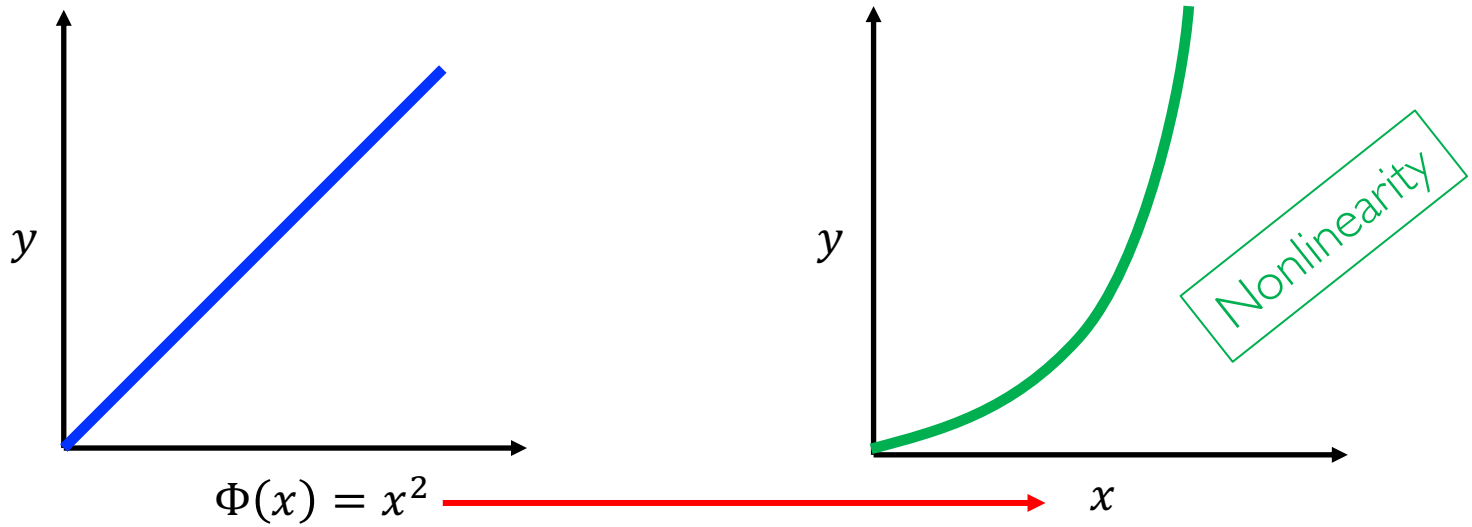


Linear Regression: Basis Function

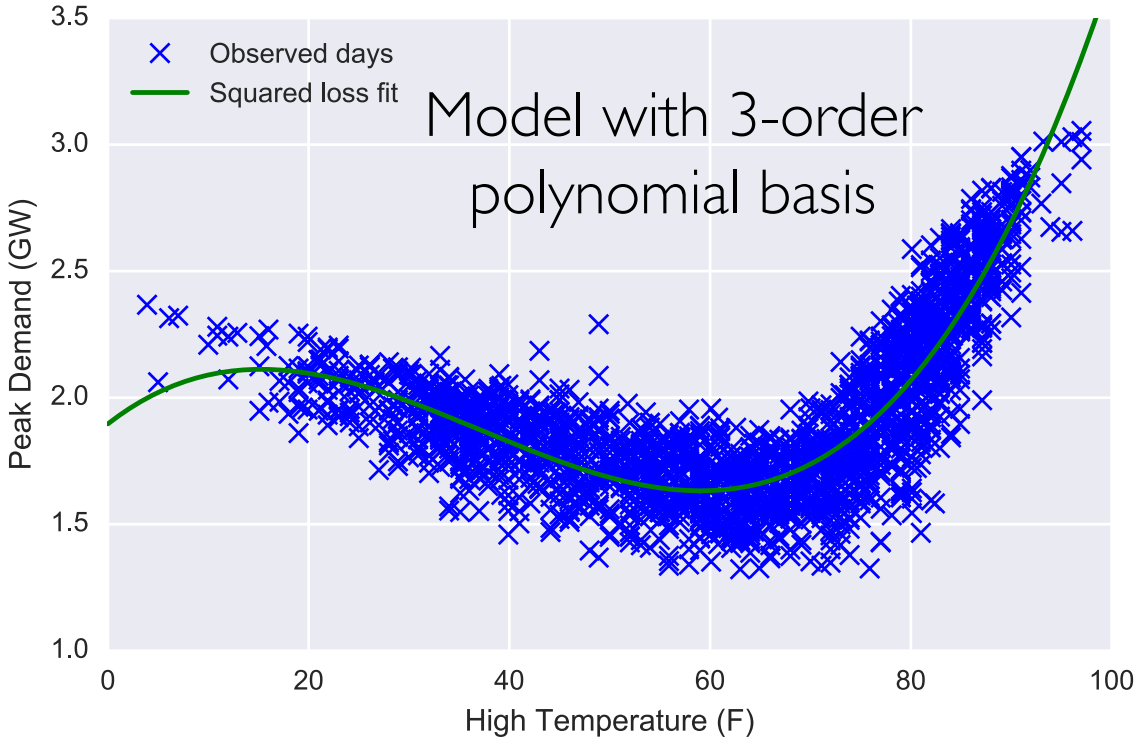
- The final hypothesis is linear in the feature space $\mathcal{Z}$.
- The final hypothesis is nonlinear in the input space $\mathcal{X}$:

$$h(\boldsymbol{x}) = \tilde{\boldsymbol{w}} \cdot \boldsymbol{z} = \tilde{\boldsymbol{w}} \cdot \Phi(\boldsymbol{x}) = \sum_{j=1}^{\tilde{d}} \tilde{w}_j \phi_j(\boldsymbol{x})$$

- Algorithm works because of linearity in the parameters (still simple!)



Linear Regression: Basis Function

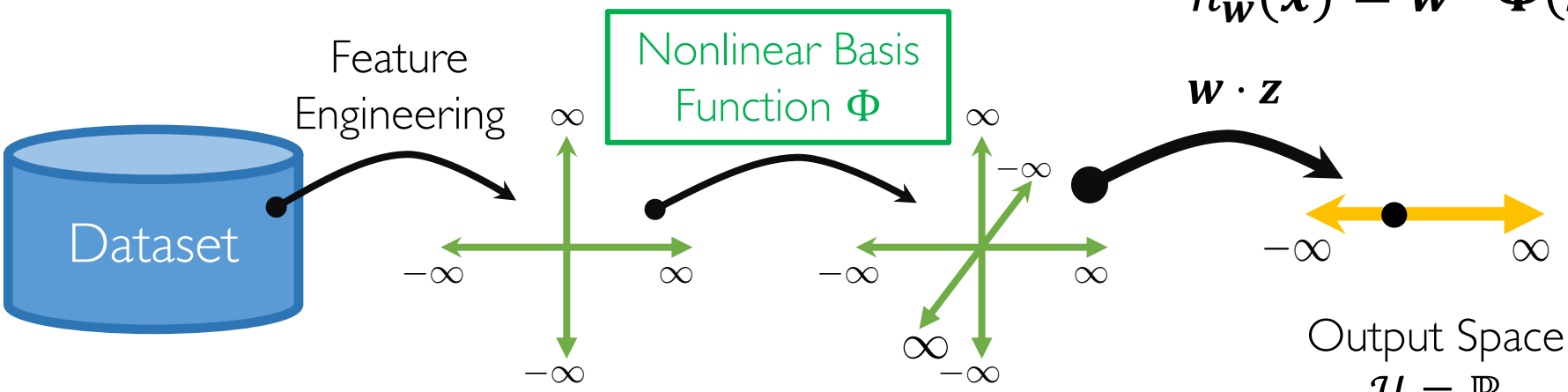


- With polynomial basis functions, we could fit nonlinear dataset.
  - Linear regression with polynomial basis functions.
- Higher-order polynomial has stronger ability to fit complex data.

Linear Regression

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



Gradient Descent

$$\nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{w} - \boldsymbol{y})$$

Analytic Solution

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

Loss Function

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

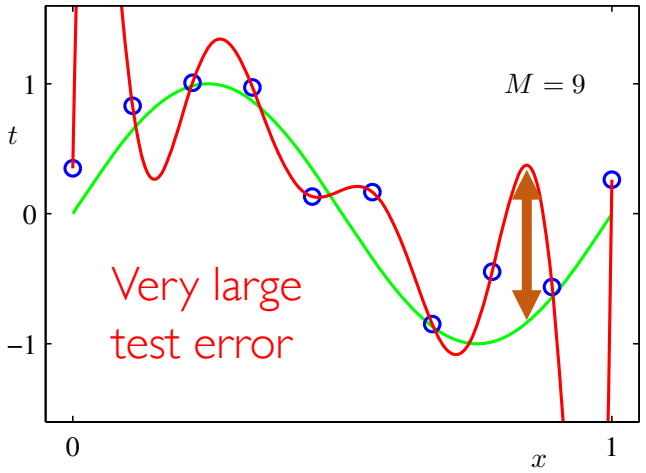
How to deal with overfitting?

Outline

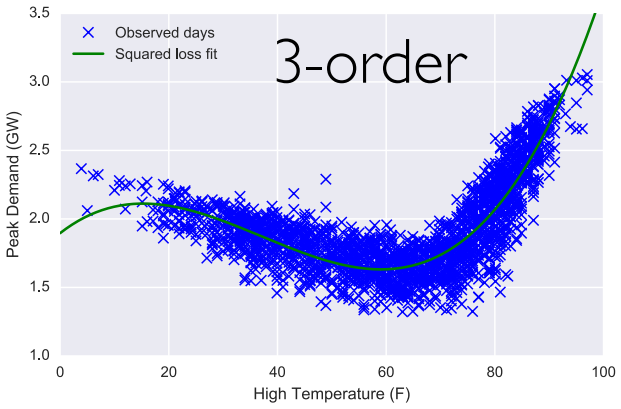
- Linear Regression
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- **Regularization**
  - **L1 and L2 Norm**
  - Statistical View: Maximum a Posteriori Estimation
- Linear Classification
  - Logistic Regression
  - Softmax Regression

Overfitting

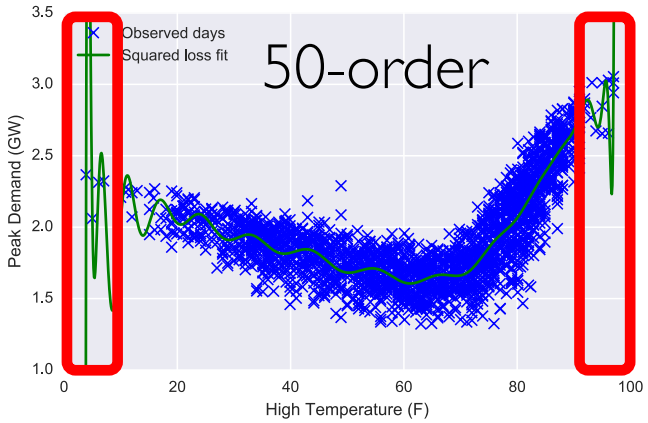
- Higher-order polynomial fits data better.
- But may fit noise!
- Also may cause very large coefficients.



Good fit



Over fit



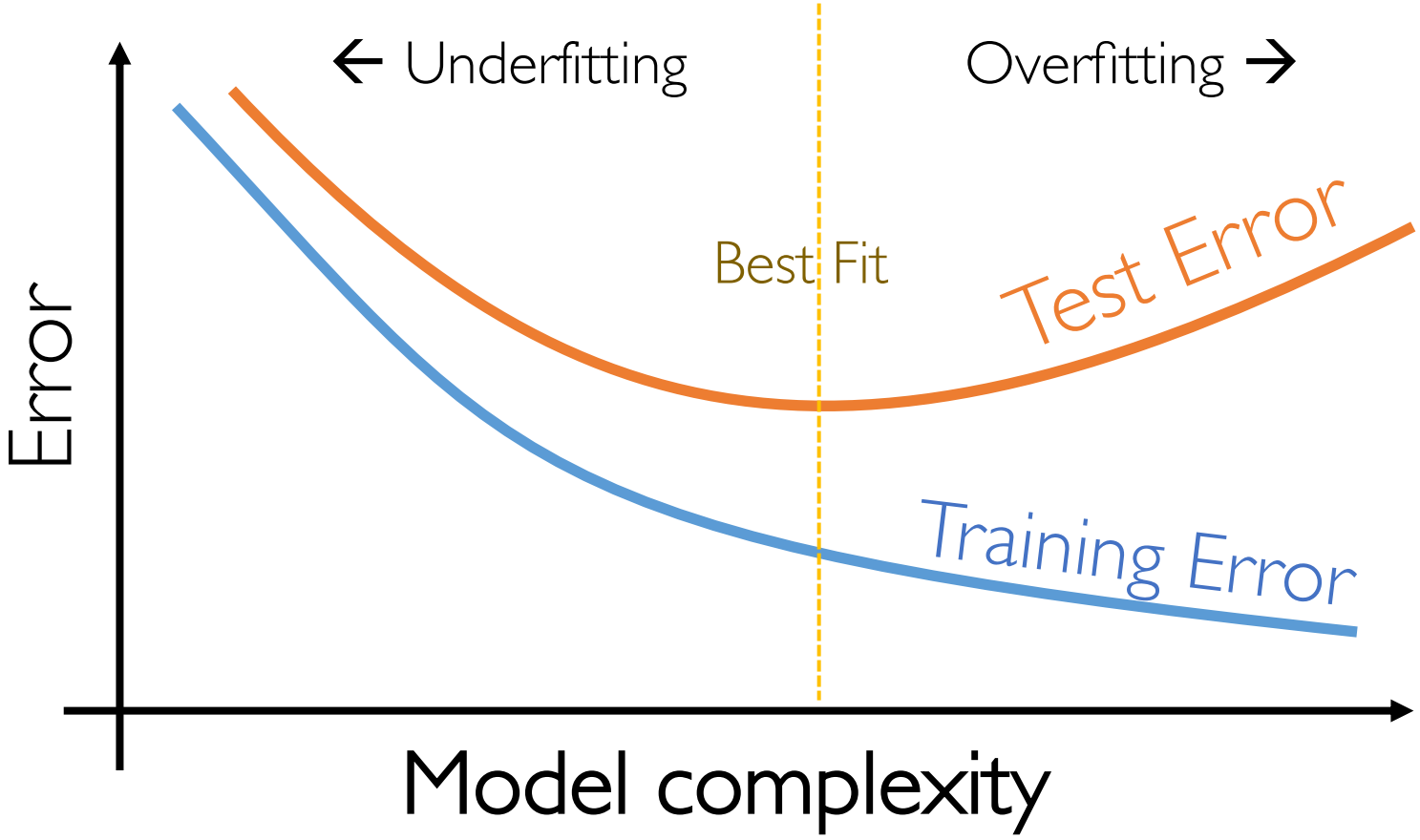
$w_0^\star$	0.35
$w_1^\star$	232.37
$w_2^\star$	-5321.83
$w_3^\star$	48568.31
$w_4^\star$	-231639.30
$w_5^\star$	640042.26
$w_6^\star$	-1061800.52
$w_7^\star$	1042400.18
$w_8^\star$	-557682.99
$w_9^\star$	125201.43

Solution: Control norm of $\boldsymbol{w}$!

Why?

When the size of dataset is fixed:

Training vs Test Error



How to control the model complexity? Solution: Control norm of $\boldsymbol{w}$!

Linear Regression: L2-Regularization

- The L2-regularized linear regression (ridge regression) imposes L2-norm regularization trading off with a hyperparameter $\lambda \geq 0$:

$$\hat{\boldsymbol{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

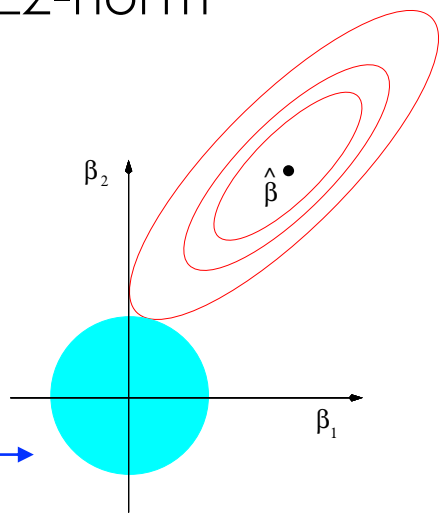
  - Where $\|\boldsymbol{w}\|_2^2 = w_1^2 + \cdots + w_d^2$ is the square of L2-norm
  - Which is equivalent to:

$$\hat{\boldsymbol{w}} = \arg\min_{\|\boldsymbol{w}\|_2^2 \leq r} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$

  for some $r > 0$.

  - $r \to 0 \iff \lambda \to +\infty$

L2 unit ball with radius r



Linear Regression: L1-Regularization

- The L1-regularized linear regression (lasso regression) imposes L1-norm regularization trading off with a hyperparameter $\lambda \geq 0$:

$$\hat{\boldsymbol{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|_1$$

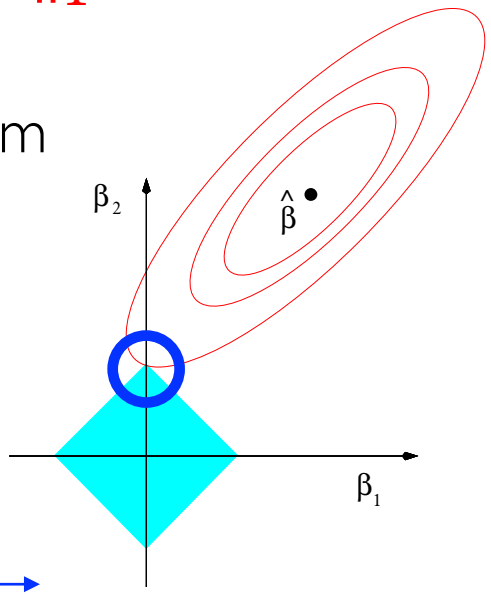
  - Where $\|\boldsymbol{w}\|_1 = |w_1| + \cdots + |w_d|$ is the L1-norm
  - Which is equivalent to:

$$\hat{\boldsymbol{w}} = \arg\min_{\|\boldsymbol{w}\|_1 \leq r} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$

  with some $r > 0$.

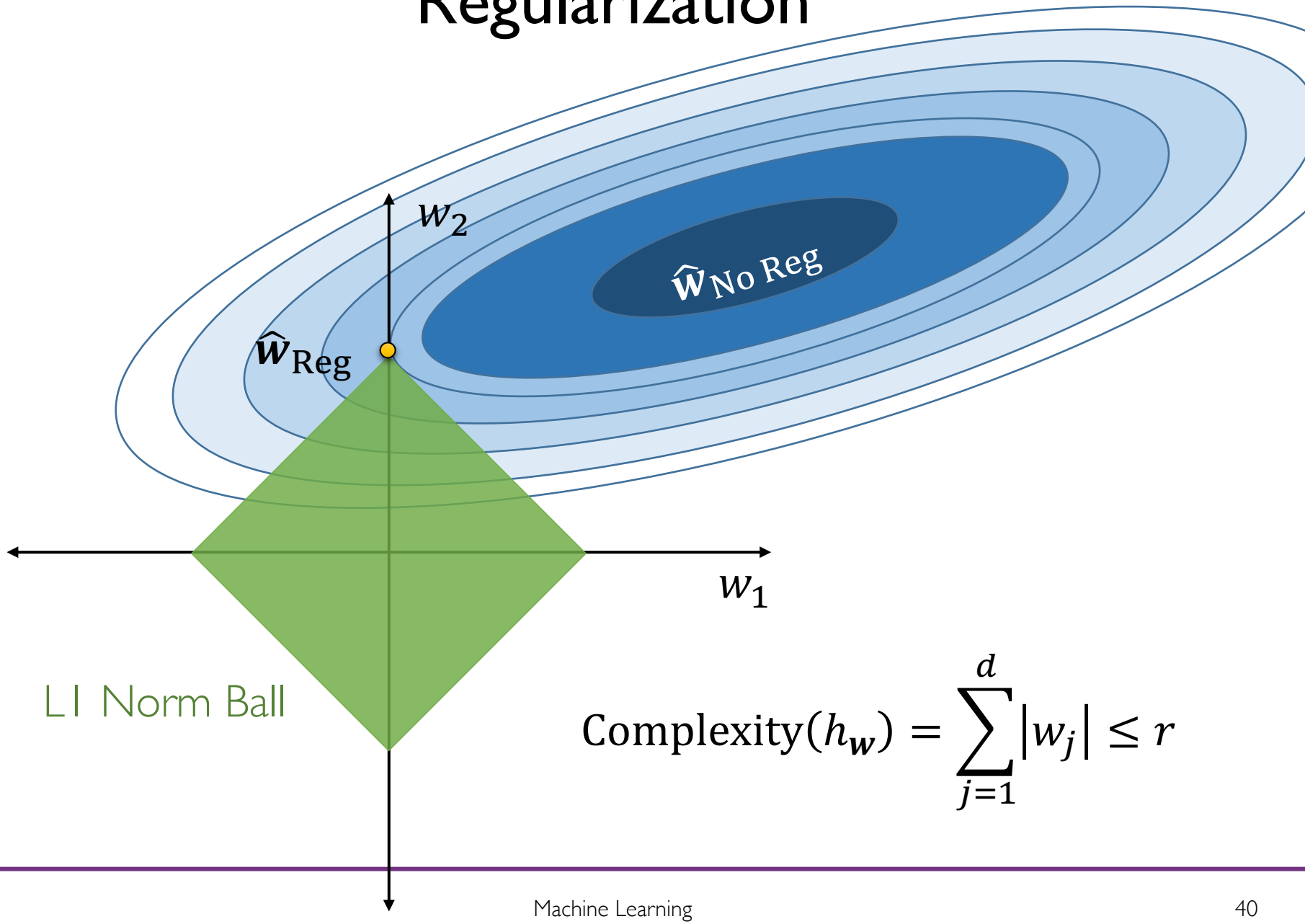
  - $r \to 0 \Leftrightarrow \lambda \to +\infty$

L1 unit ball with radius r



Sparse Solution!

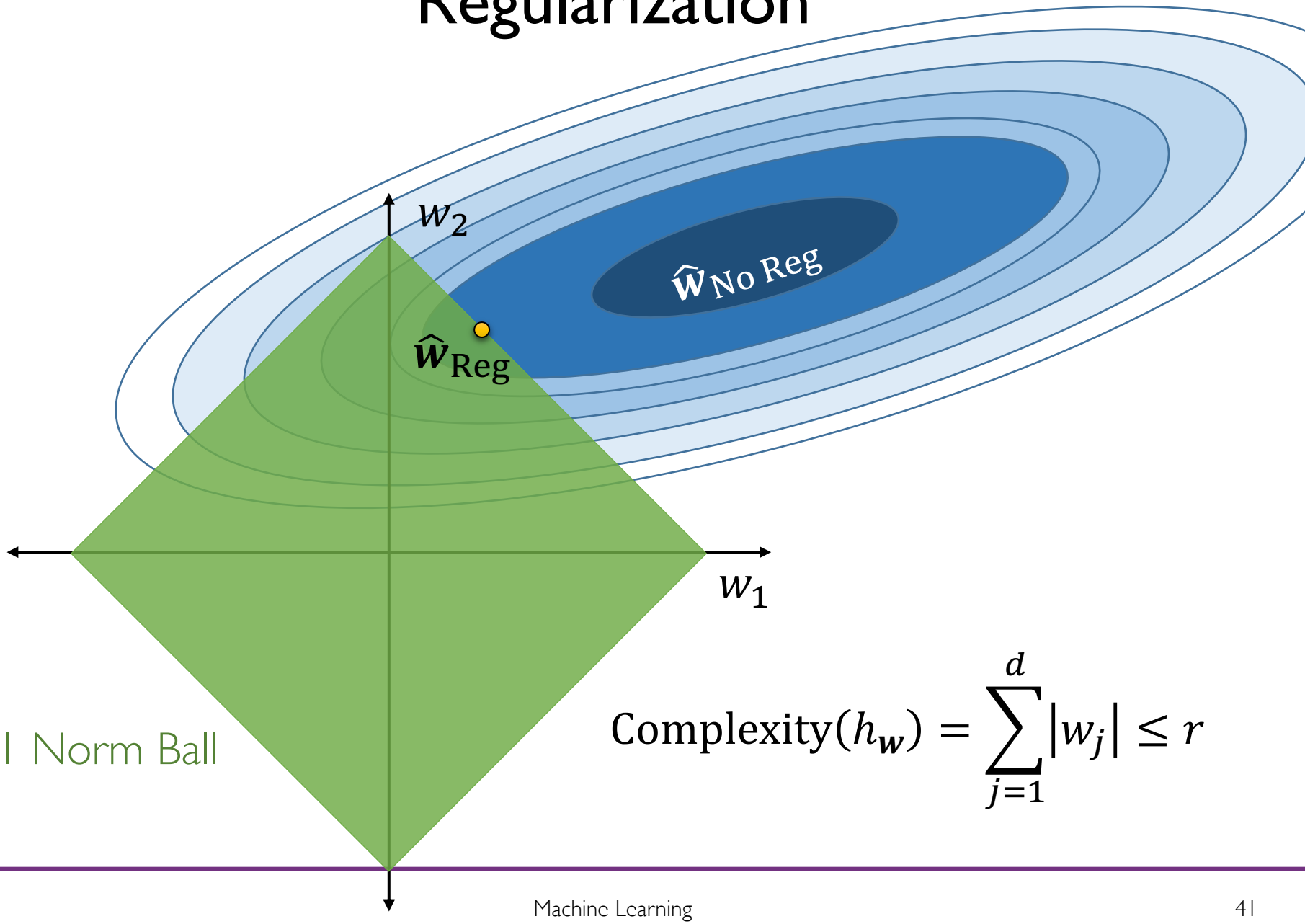
Regularization



L1 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} |w_j| \leq r$$

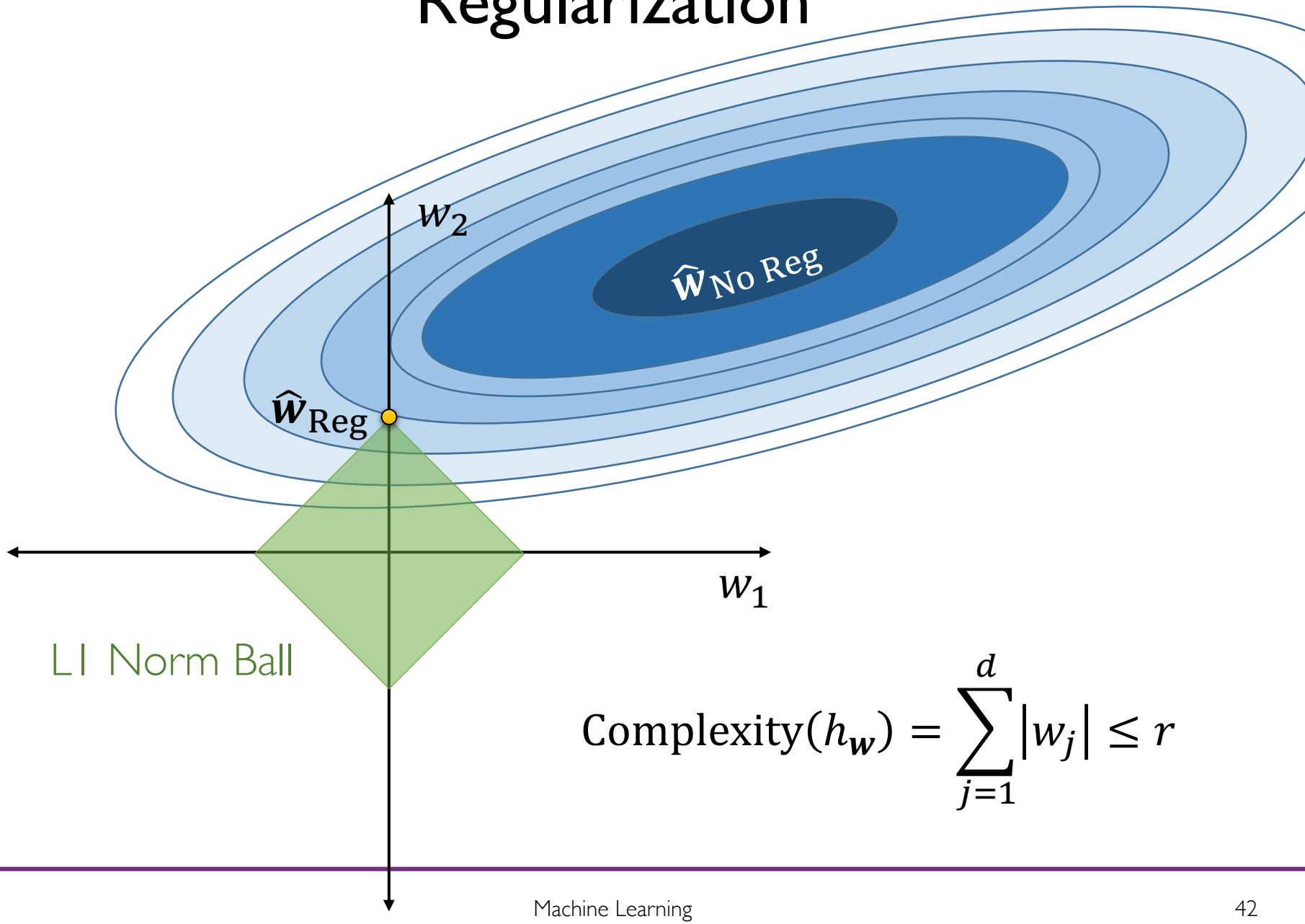
Regularization



L1 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} |w_j| \leq r$$

Regularization

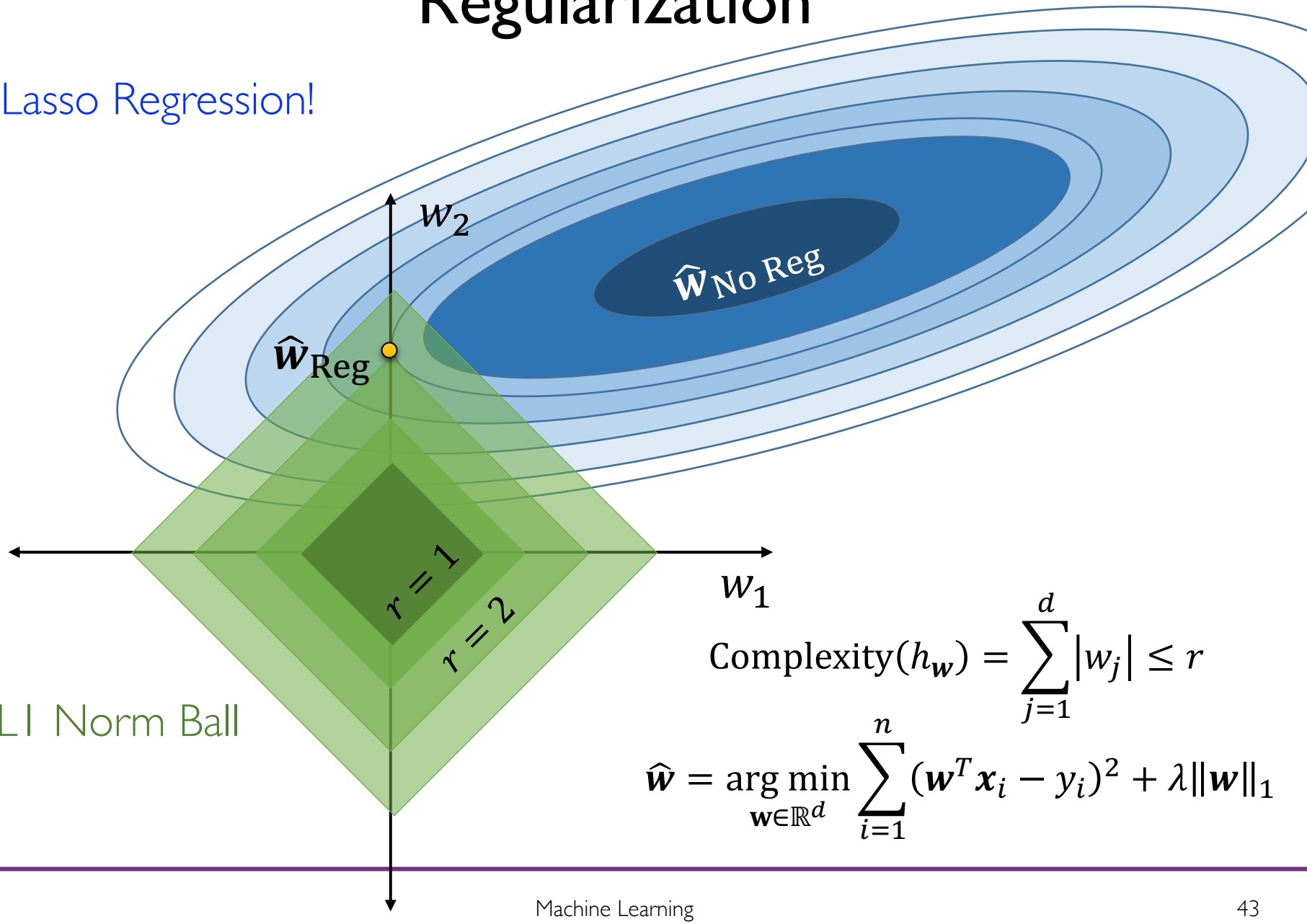


L1 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} |w_j| \leq r$$

Regularization

Lasso Regression!

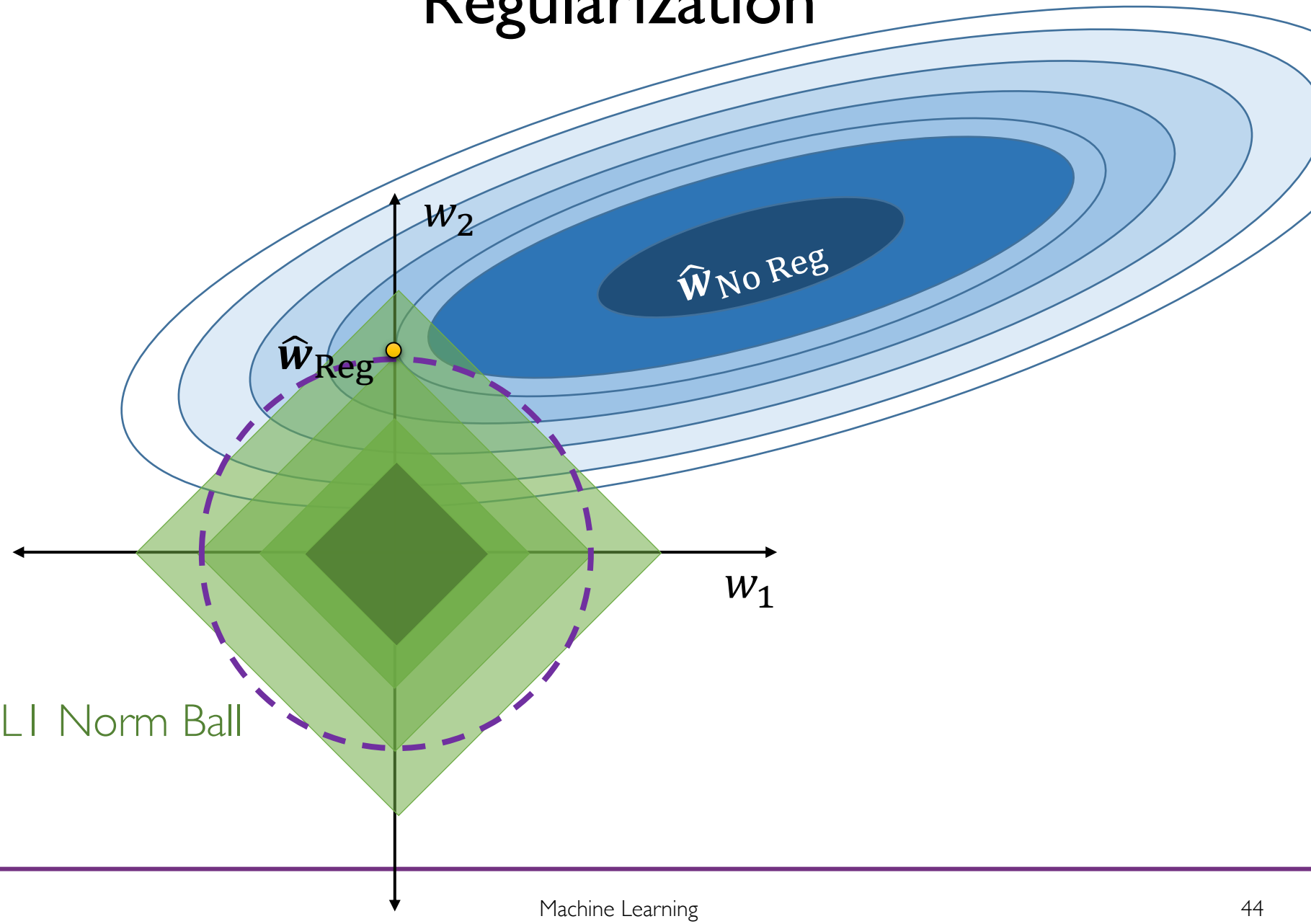


L1 Norm Ball

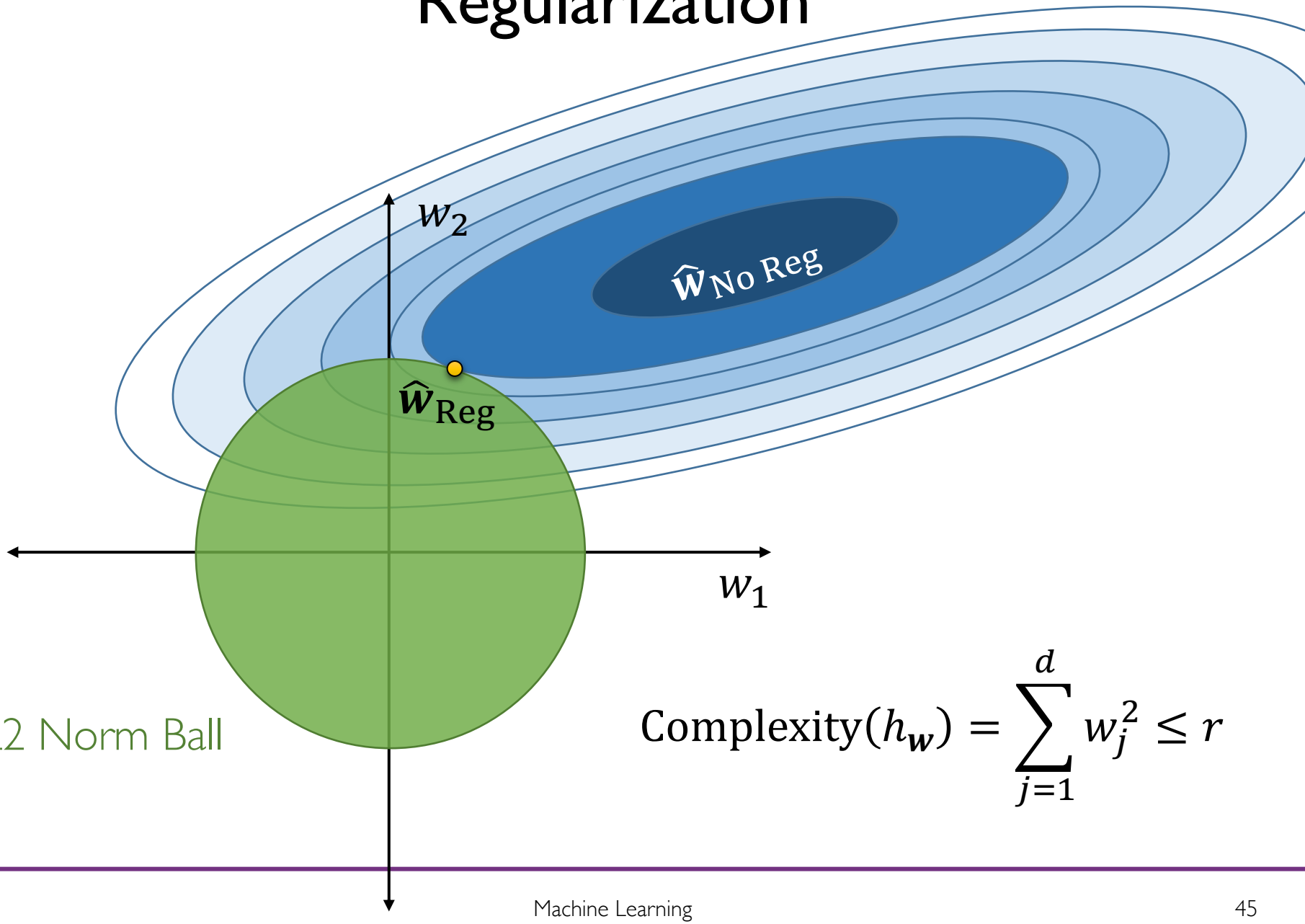
$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} |w_j| \leq r$$

$$\widehat{\boldsymbol{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|_1$$

Regularization



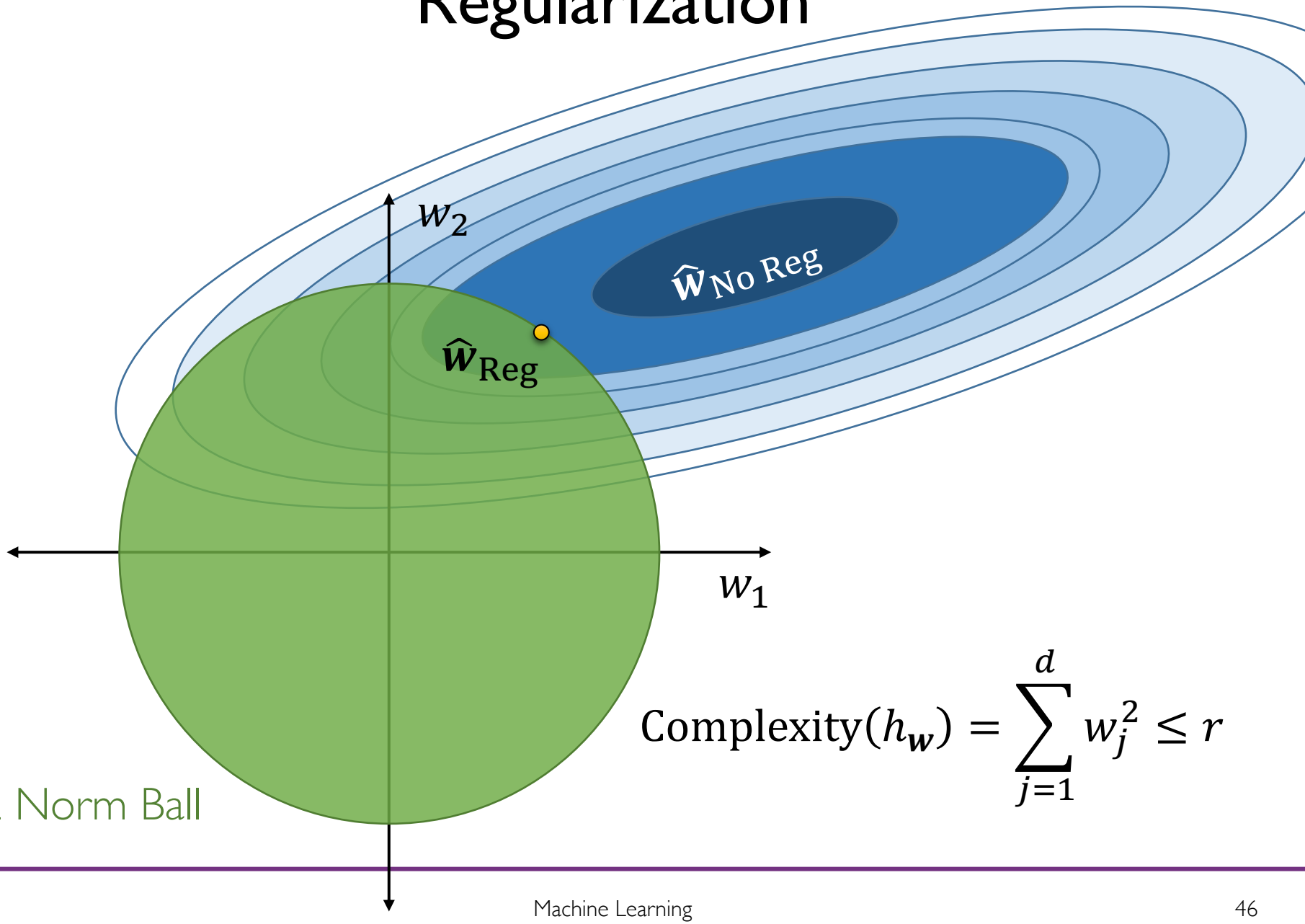
Regularization



L2 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} w_j^2 \leq r$$

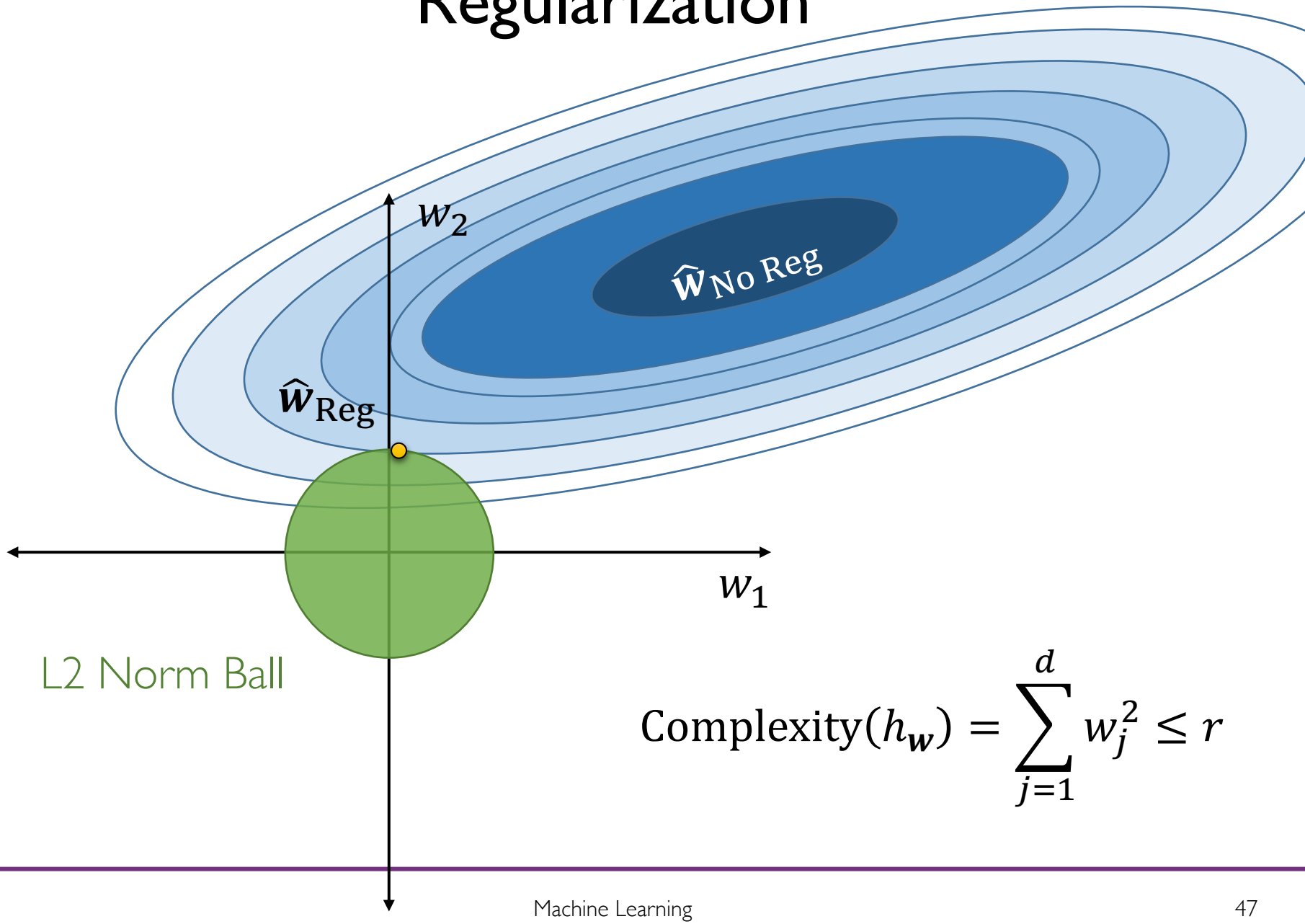
Regularization



$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} w_j^2 \leq r$$

L2 Norm Ball

Regularization

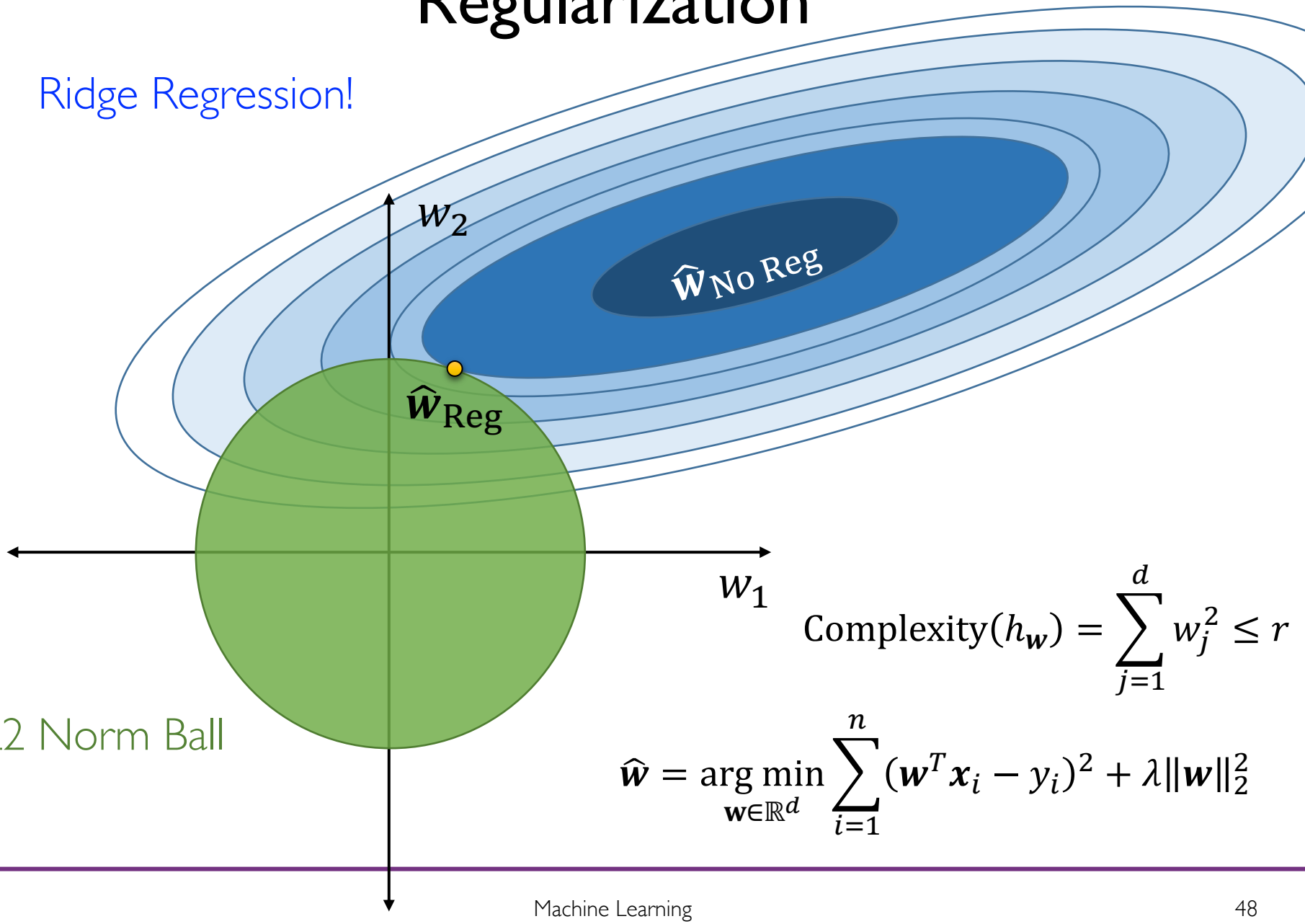


L2 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} w_j^2 \leq r$$

Regularization

Ridge Regression!

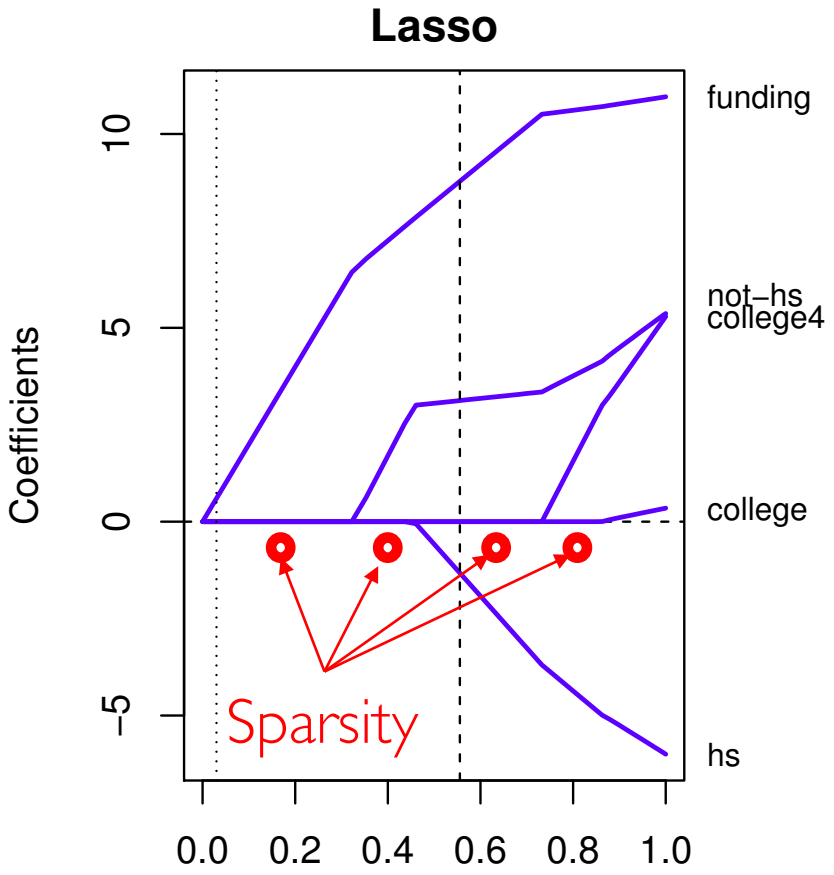


L2 Norm Ball

$$\text{Complexity}(h_{\boldsymbol{w}}) = \sum_{j=1}^{d} w_j^2 \leq r$$

$$\widehat{\boldsymbol{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

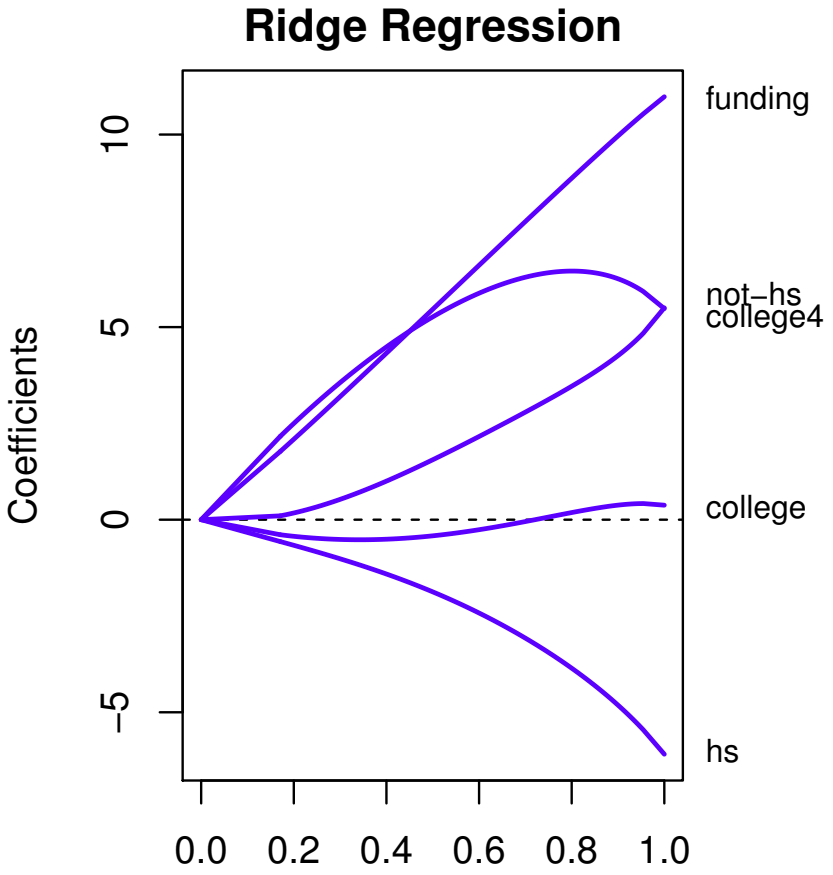
Regularization Path: L1 vs. L2



$\|\hat{\boldsymbol{w}}_r\|_1/\|\hat{\boldsymbol{w}}\|_1$

Feature selection

$$\hat{\boldsymbol{w}}_r = \arg\min_{\|\boldsymbol{w}\|_1 \le r} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$



$\|\hat{\boldsymbol{w}}_r\|_2/\|\hat{\boldsymbol{w}}\|_2$

Smooth optimization

Outline

- Linear Regression
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- **Regularization**
  - L1 and L2 Norm
  - **Statistical View: Maximum a Posteriori Estimation**
- Linear Classification
  - Logistic Regression
  - Softmax Regression

Bayes Rule in Parametric Models

- Bayes rule:

Observed Data Parameter

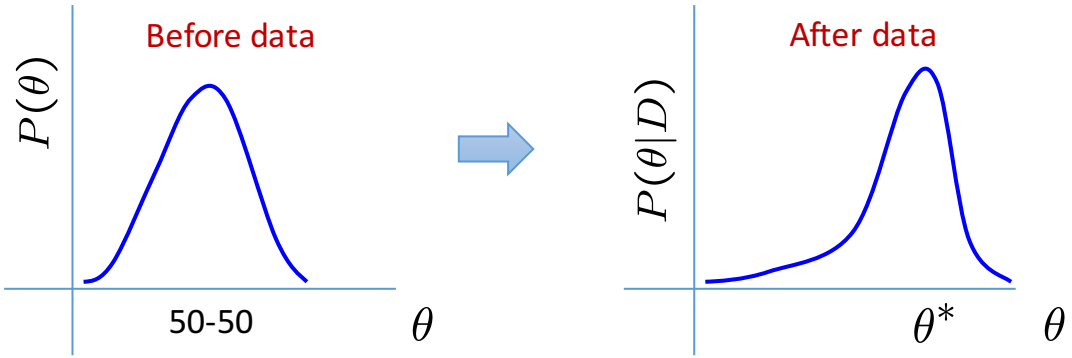
$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

- Or equivalently:

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

Posterior Likelihood Prior

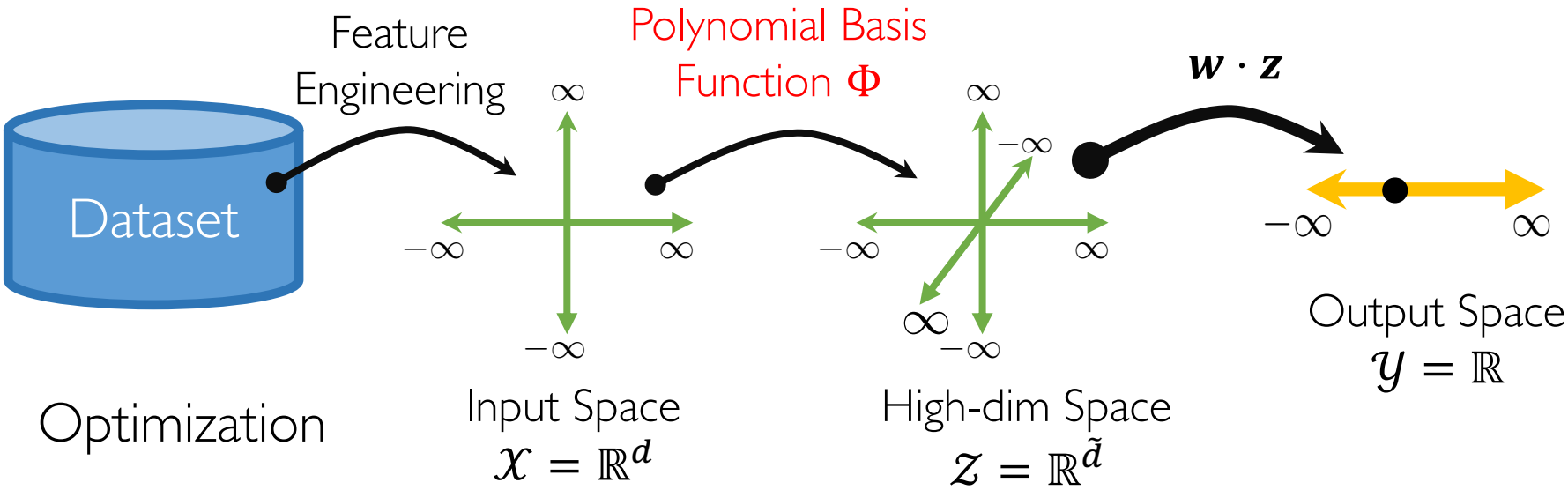
Thomas Bayes



Linear Regression

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



Gradient Descent

$$\nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w}) = 2\boldsymbol{Z}^T(\boldsymbol{Z}\boldsymbol{w} - \boldsymbol{y}) + 2\lambda \boldsymbol{w}$$

Analytic Solution

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

Loss Function

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda \boldsymbol{\Omega}(\boldsymbol{w})$$

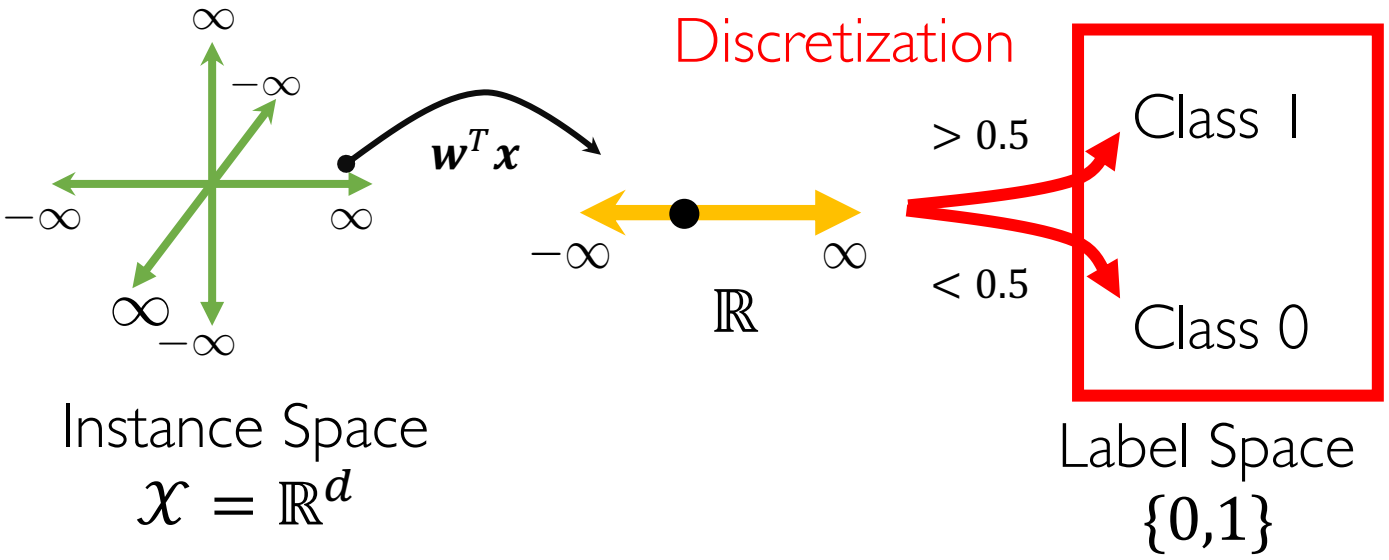
Well Done! A simple but complete model!

Outline

- Linear Regression
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- **Linear Classification**
  - **Logistic Regression**
  - Softmax Regression

Regression to Classification

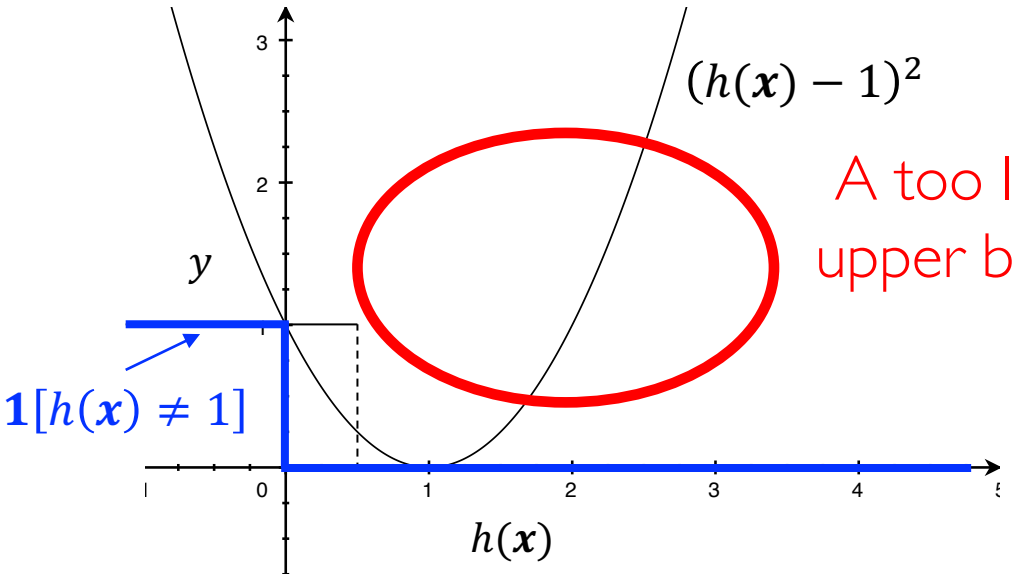
- How to design loss function over linear hypothesis for classification?
- Naïve idea: $\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda \Omega(\boldsymbol{w})$ where $y_i \in \{0,1\}$
  - Discretize the continuous output $h_{\boldsymbol{w}}(\boldsymbol{x}_i)$ to be $\{0,1\}$
- During test part:

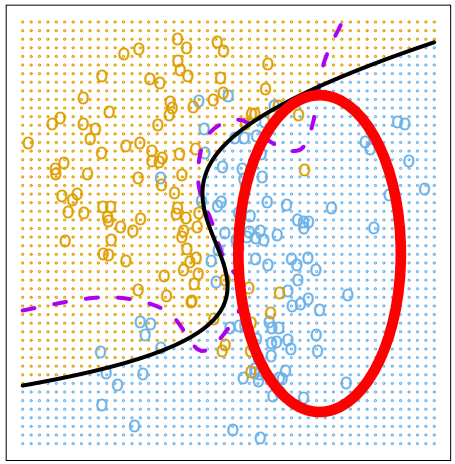


Regression to Classification

- It does not make sense for $h_{\boldsymbol{w}}$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0,1\}$.
- We use accuracy (01-loss) when validating the classification model:

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}[y \neq h(\boldsymbol{x})]$$





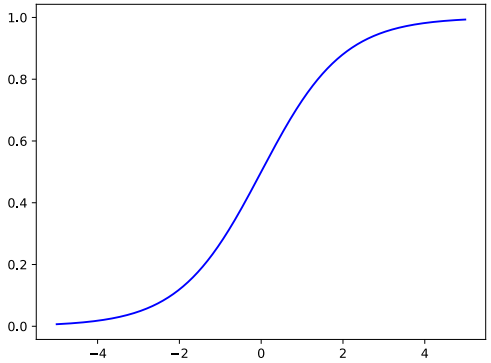
Outputs on all these points should exactly be $\mathbf{1}$ or $\mathbf{0}$?

- $h(\boldsymbol{x}) \in \mathcal{Y}$ is not a proper assumption this time.

Logistic Regression: Link Function

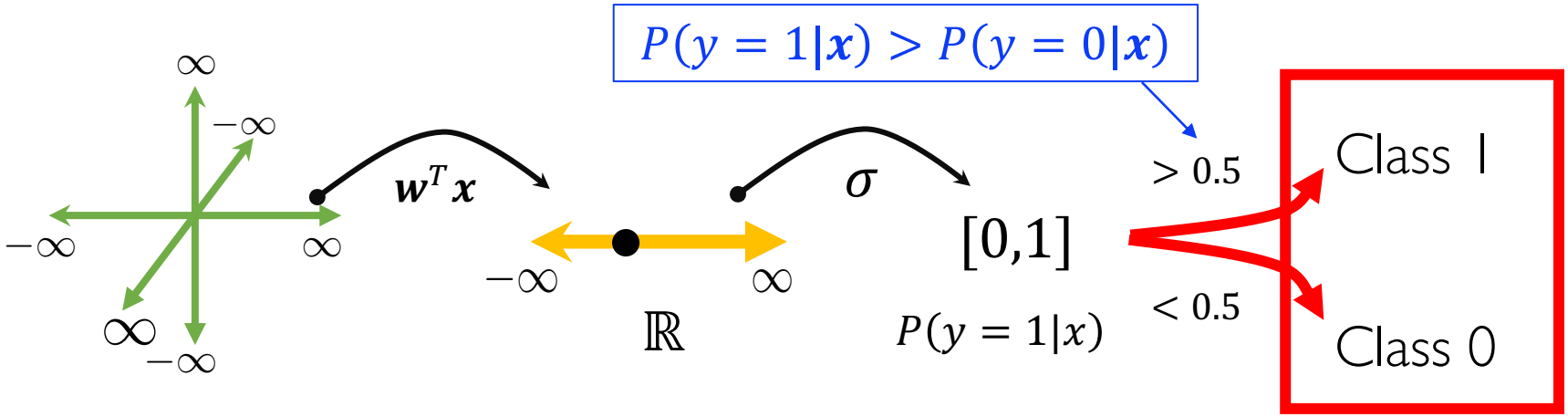
- To map the output of $h(\boldsymbol{x})$ into $[0,1]$, we use sigmoid function:

$$\sigma(t) = \frac{1}{1+e^{-t}}$$



  - Sigmoid maps $\mathbb{R}$ to $[0,1]$.
  - $t \to +\infty, \sigma(t) \to 1$.

- Set $\sigma(h(\boldsymbol{x})) = p(y=1|\boldsymbol{x})$ as the probability to label $\boldsymbol{x}$ as $y=1$.

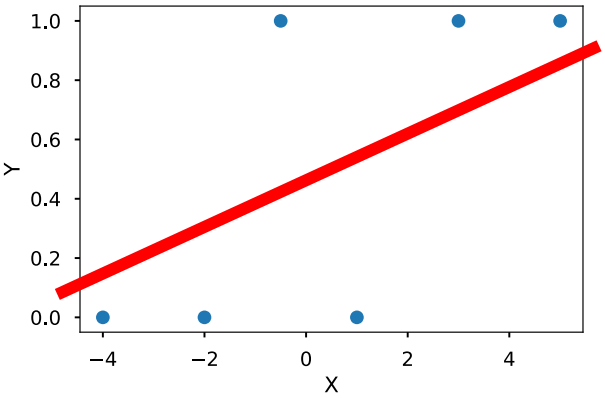


Logistic Regression: Loss Function

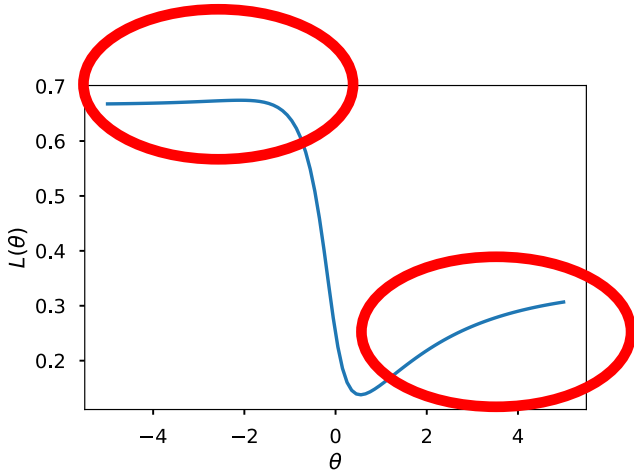
- How about using squared loss now as $h(\boldsymbol{x}) \in [0,1]$ and $y \in \{0,1\}$?

$$\ell(h(\boldsymbol{x}), y) = \sum_{i=1}^{n} (\sigma(\boldsymbol{w}^T \boldsymbol{x}_i) - y_i)^2 + \lambda \Omega(\boldsymbol{w})$$

- Tries to match continued probability with discrete 0/1 labels.
  - Non-convex
  - Small loss when prediction is overly far in wrong side
  - What is wrong?



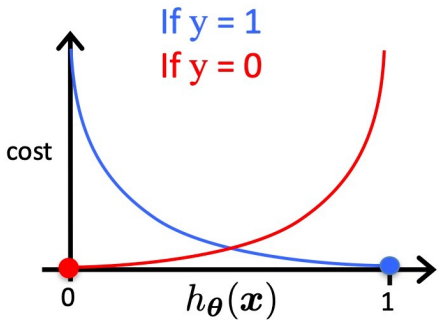
Toy data



Squared Loss Surface

Logistic Regression: Cross-Entropy Loss

$$\ell(h(\boldsymbol{x}_i), y_i) = \begin{cases} -\log[\sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] & y_i = 1 \\ -\log[1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] & y_i = 0 \end{cases}$$

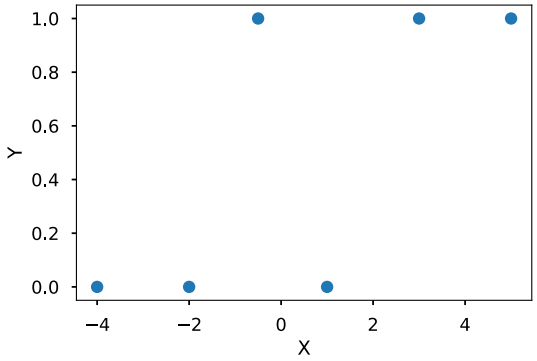


Check! If $y_i = 1$,

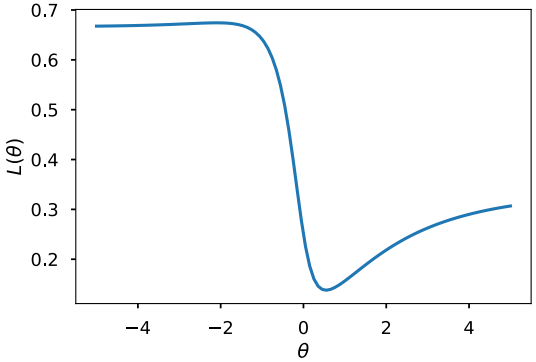
- $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \to 0$, loss goes to ∞
- $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \to 1$, loss goes to 0

$P(y_i = 1 | \boldsymbol{x}_i)$

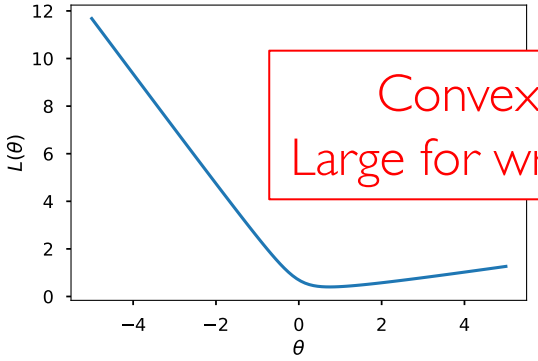
Toy data



Squared Loss Surface



Cross-Entropy Surface

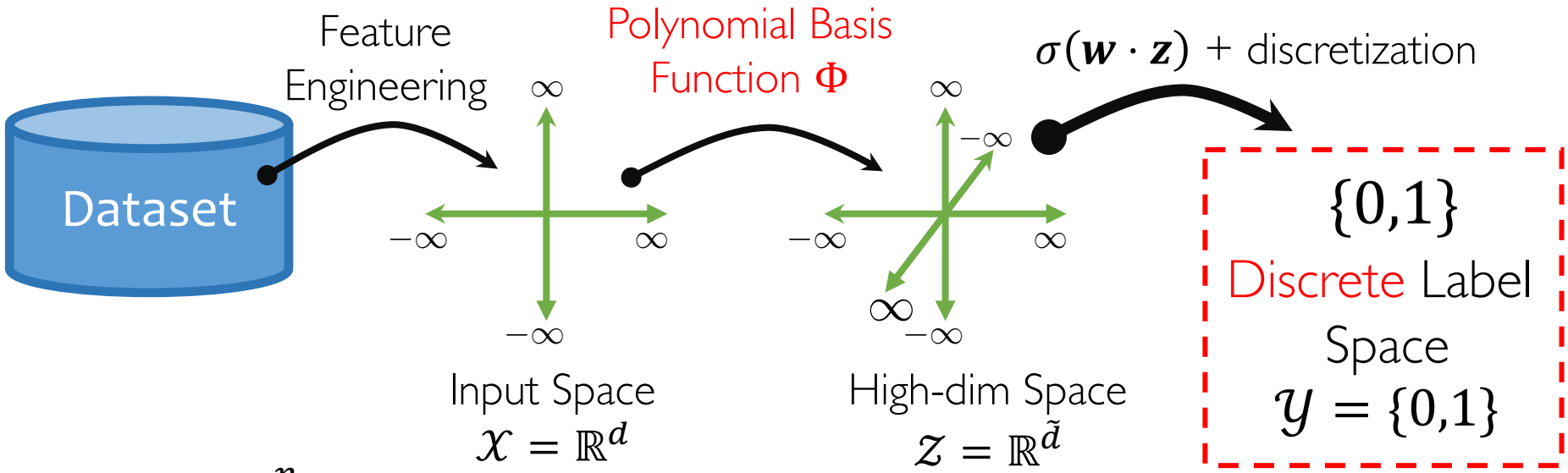


Logistic Regression

2-class Classification:

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



$$\hat{\epsilon}(w) = -\sum_{i=1}^{n} \{y_i \log \sigma(h_{\boldsymbol{w}}(\boldsymbol{x})) + (1 - y_i) \log[1 - \sigma(h_{\boldsymbol{w}}(\boldsymbol{x}))]\} + \lambda \Omega(\boldsymbol{w})$$

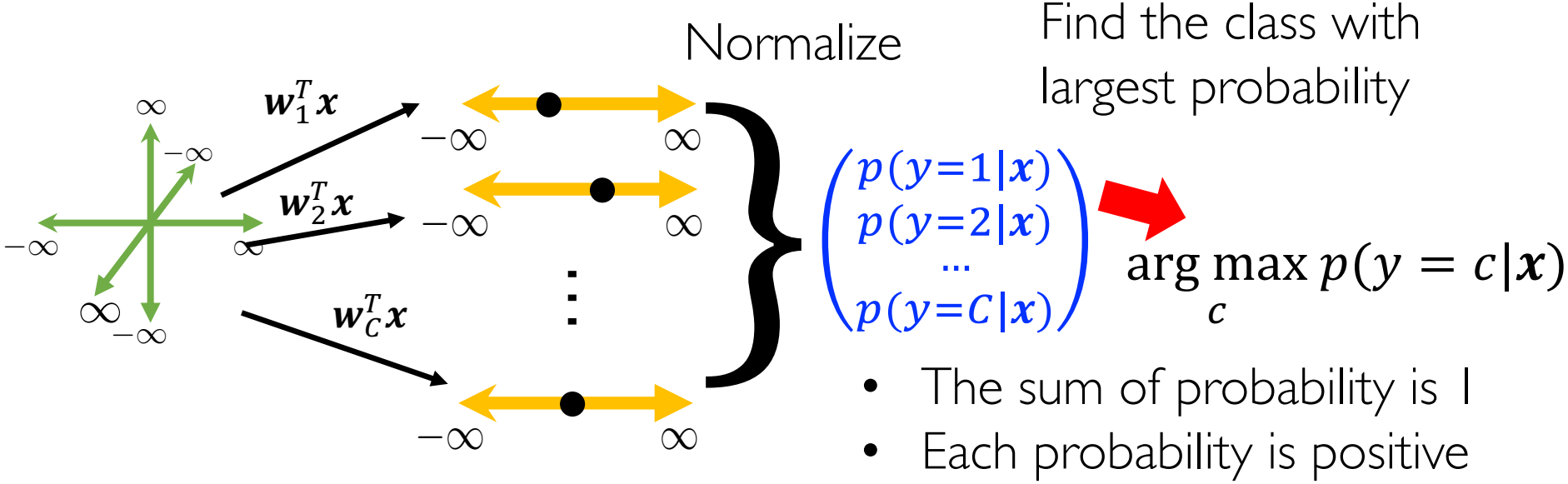
- How to solve this problem?
  - No analytic solution! Using GD, SGD, Newton method…
- How to deal with multiclass problems?

Outline

- Linear Regression
  - Optimization
  - Statistical View: Maximum Likelihood Estimation
- Nonlinearization
- Regularization
  - L1 and L2 Norm
  - Statistical View: Maximum a Posteriori Estimation
- **Linear Classification**
  - Logistic Regression
  - **Softmax Regression**

Multiclass Classification

- We keep on mapping the output of linear hypothesis into probability.
- There are multiple classes, so we use multiple linear predictors to predict the probability: $\boldsymbol{w_1, w_2, \ldots, w_C}$.
- How to change the outputs of multiple predictors into probability?



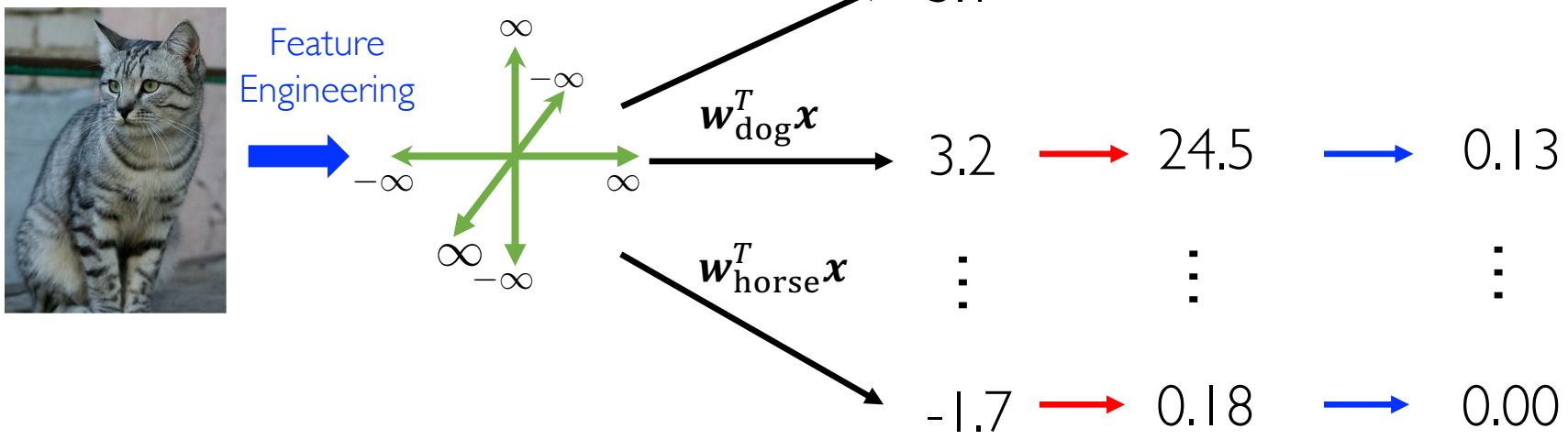
- The sum of probability is 1
- Each probability is positive

Softmax Regression: Softmax Function

- Softmax function normalizes multiple outputs in a probability vector:

$$p(y = c|\boldsymbol{x}; \boldsymbol{W}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x})}$$

Sum over all classes



Softmax Regression: Cross-Entropy Loss

Loss for each data point $(\boldsymbol{x}_i, y_i)$:

$$\ell(h(\boldsymbol{x}_i), y_i) = \begin{cases} -\log \frac{\exp(\boldsymbol{w}_1^T \boldsymbol{x}_i)}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x}_i)}, y_i = 1 \\ -\log \frac{\exp(\boldsymbol{w}_2^T \boldsymbol{x}_i)}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x}_i)}, y_i = 2 \\ \vdots \\ -\log \frac{\exp(\boldsymbol{w}_C^T \boldsymbol{x}_i)}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x}_i)}, y_i = C \end{cases}$$

Equivalent to maximum log-likelihood estimation

These two hypotheses have same outputs!

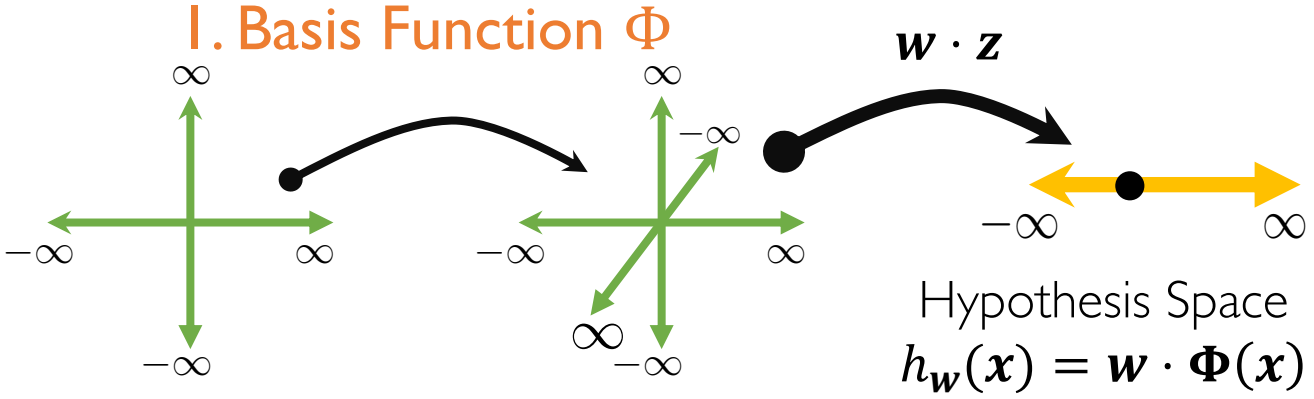
$$\begin{pmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \dots \\ \boldsymbol{w}_C \end{pmatrix} \quad \begin{pmatrix} \boldsymbol{w}_1 + \boldsymbol{V} \\ \boldsymbol{w}_2 + \boldsymbol{V} \\ \dots \\ \boldsymbol{w}_C + \boldsymbol{V} \end{pmatrix}$$

This loss is still convex. ✅

Add regularization! Or the parameter will go to infinity.

Minimize this loss: GD, SGD…

Linear Model: Summary



Objective function may not be linear to the parameters.

$$\min_{w} \sum_{i=1}^{n} \ell(h_w(x_i), y_i) + \lambda\Omega(w)$$

4. Optimization

- High-dimension
  - GD
- High-dimension big data
  - SGD
- Non-differentiable
  - Subgradient

2. Loss Function

- Regression
  - Squared Loss
- Classification
  - Cross-Entropy

3. Regularization

- General
  - L2 Regularizer
- Sparse Solution
  - L1 Regularizer