Learning (I)

Mingsheng Long

Tsinghua University

Spring 2023

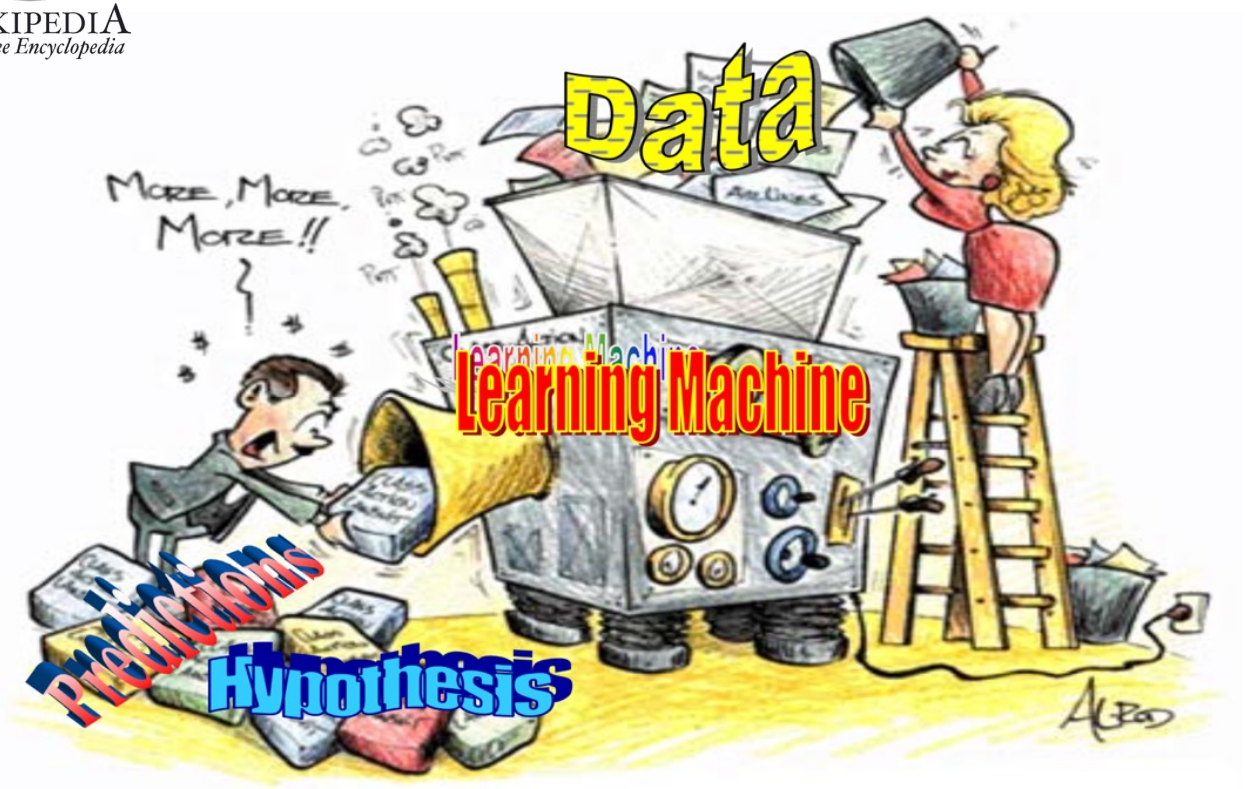
Outline

- **Machine Learning**
  - Framework
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- Linear Regression (LR)
  - Optimization
  - Nonlinearization
  - Regularization

What is Machine Learning

WIKIPEDIA
The Free Encyclopedia

Machine learning is a field of study that gives computers the ability to learn from data without being explicitly programmed.



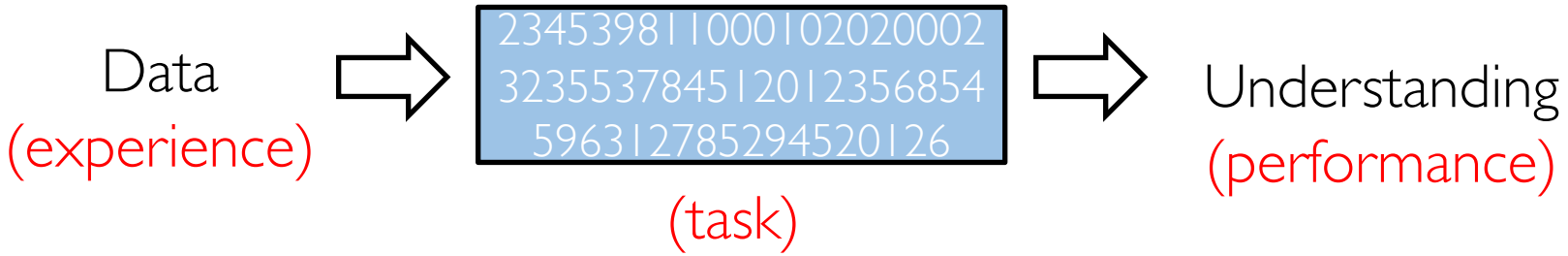
Arthur Samuel

Samuel, Arthur. Some Studies in Machine Learning Using the Game of Checkers. IBM J. R&D 3(3), 1959.

Machine Learning

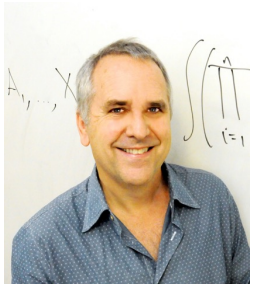
- How to build computers that:
  - (automatically) improve their performance (**P**)
  - at some task (**T**)
  - with experience (**E**)

Tom Mitchell

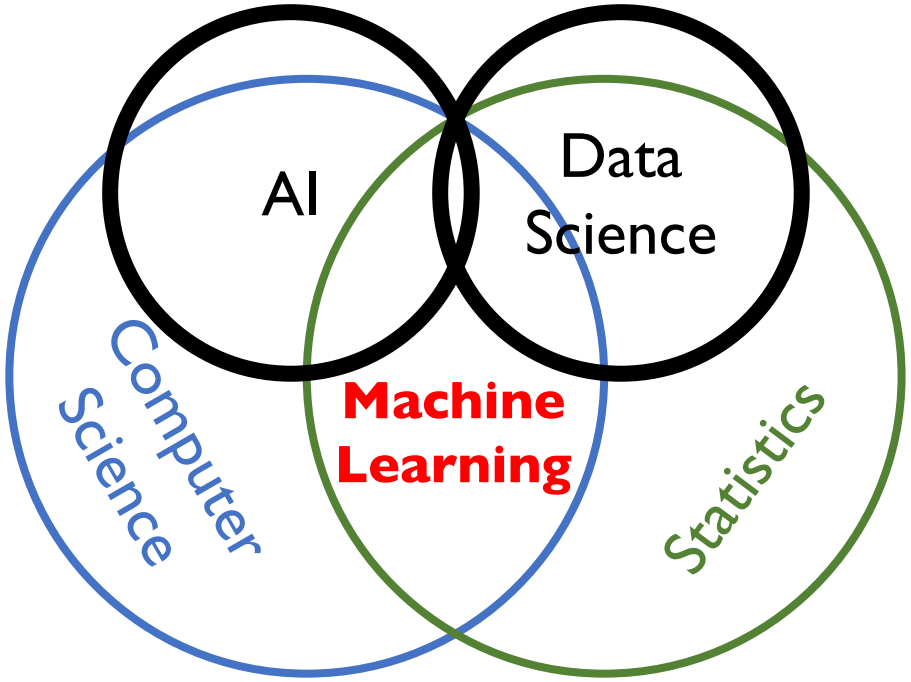


Mitchell, Tom M. Machine learning. 1997. Burr Ridge, IL: McGraw Hill 45.37 (1997): 870-877.

Machine Learning



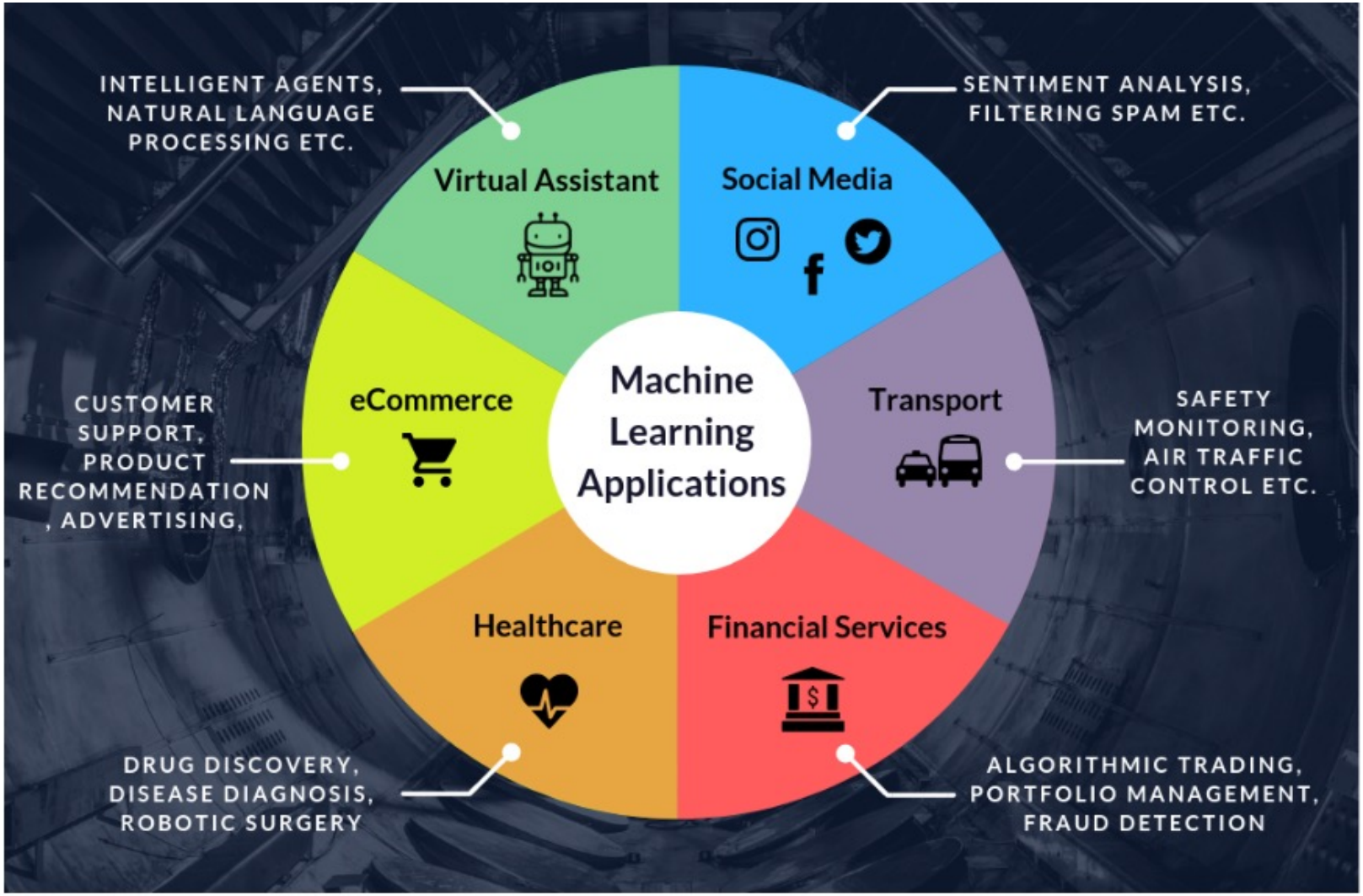
Michael Jordan



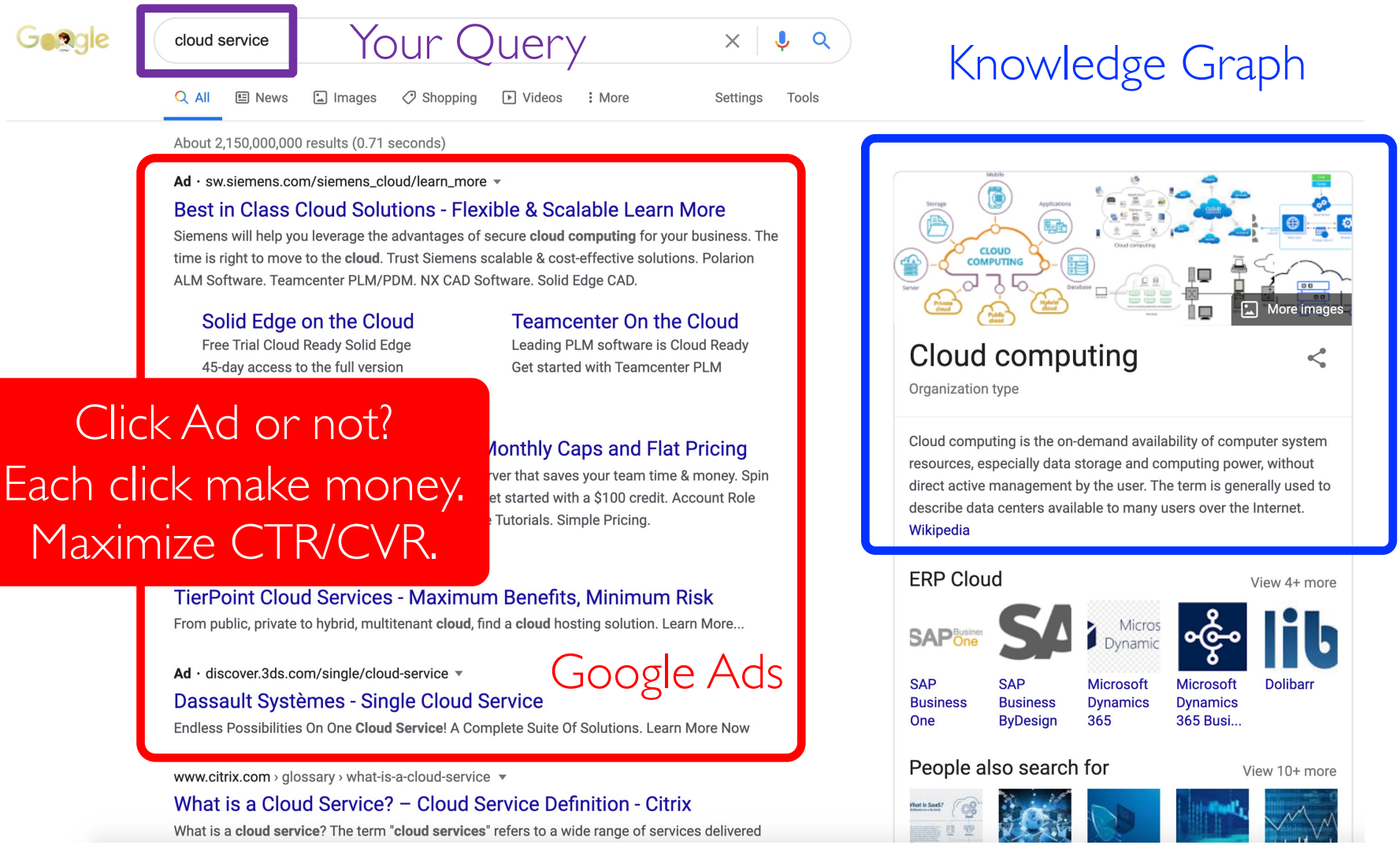
At the intersection of computer science and statistics, and at the core of artificial intelligence and data science.

M. I. Jordan1, T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. **Science**. 2015.

Machine Learning Applications



Advertising



COVID-19

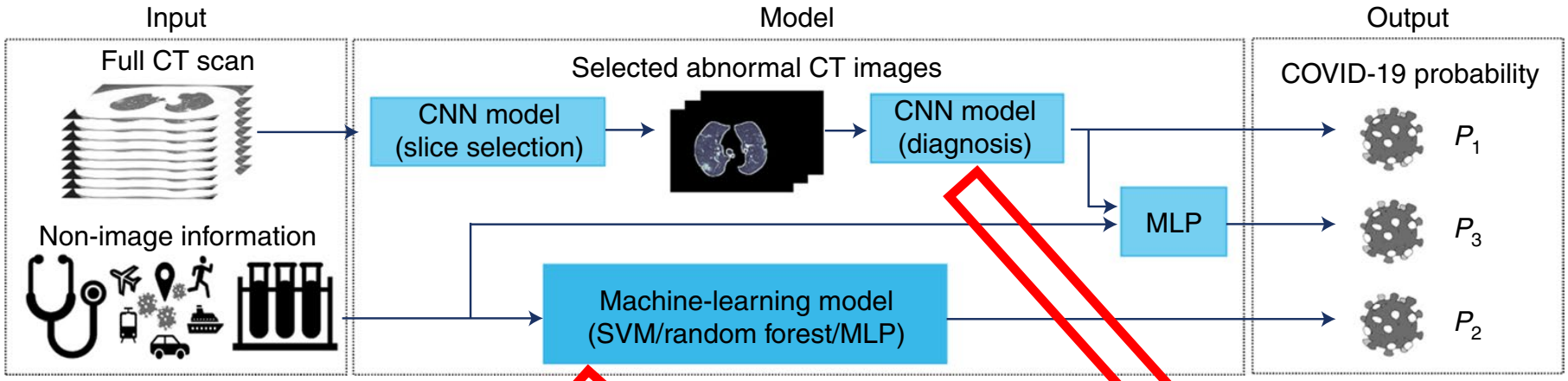
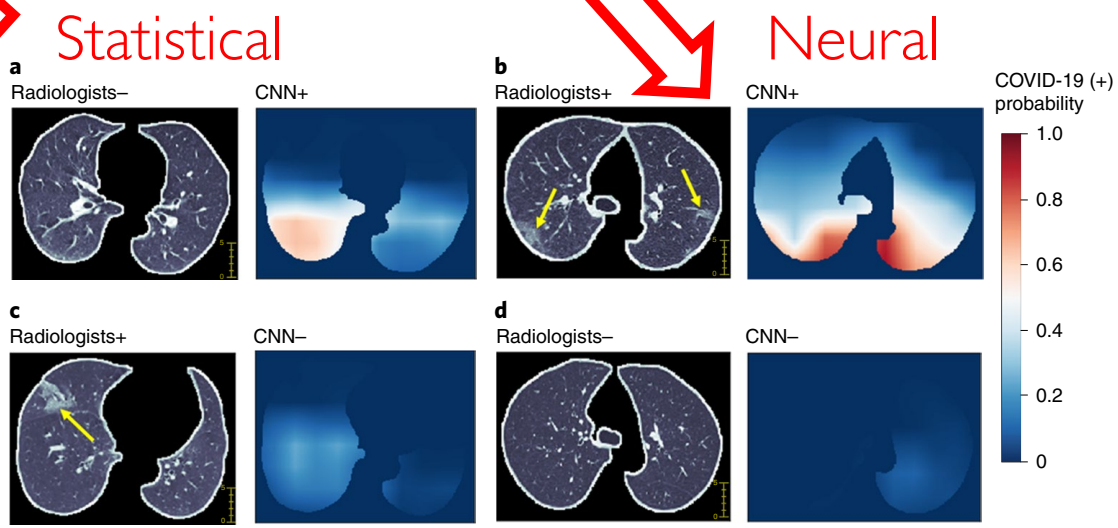


Table 1 | Characteristics of patient's clinical information

	COVID-19 positive ($n=419$)	COVID-19 negative ($n=486$)	*P* value
Sex			
Men	208 (49.6)	280 (57.6)	0.36
[a,b]**Age (years)**	43.0±16.4 (32, 54)	38.6±16.3 (27, 48)	0.00086
[b]**Temperature (°C)**	37.6±0.9	37.5±1.0	0.60
Exposure history	319 (76.1)	254 (52.3)	$<1\times10^{-4}$
Clinical symptoms			
Fever	314 (75.0)	318 (65.4)	0.0089
Cough	210 (50.1)	128 (26.3)	$<1\times10^{-4}$
Cough with sputum	83 (20.0)	177 (36.4)	$<1\times10^{-4}$



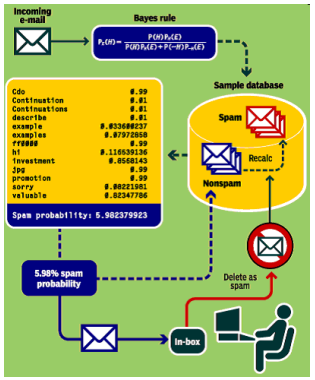
Artificial intelligence – enabled rapid diagnosis of patients with COVID-19. **Nature Medicine**. 2020.

Outline

- **Machine Learning**
  - **Framework**
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- Linear Regression (LR)
  - Optimization
  - Nonlinearization
  - Regularization

An Example: Spam Filtering

- Task: Spam Filtering (binary classification)
- Email information:



From: mingsheng@tsinghua.edu.cn
Date: 5:20 p.m. September 14, 2020
Subject: Class announcement

Hello students,
Welcome to Machine Learning!
Here's what...

From: a5p7kn@hotmail.com
Date: 2:30 a.m. September 9, 2019
Subject: URGENT

Dear Sir or maDam:
my friend left sum of 10m dollars...

Label: Spam or Not-spam

- Can we program that directly or train a model to do it automatically?

An Example: Spam Filtering

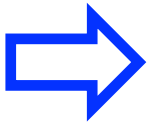
- Feature extraction: by hand or learn features automatically

From: a5p72n@hotmail.com
Date: 2:30 a.m. September 9, 2019
Subject: URGENT

Dear Sir or maDam:
my friend left sum of 10m dollars...

From: mingsheng@tsinghua.edu.cn
Date: 5:20 p.m. September 14, 2020
Subject: Class announcement

Hello students,
Welcome to Machine Learning!
Here's what...



Category	Spam
FracOfAlpha	0.5
Sent at midnight	1
Subject len	6
Contains 'dollar'	1

label

feature vector

Category	Not-spam
FracOfAlpha	1.0
Sent at midnight	0
Subject len	18
Contains 'dollar'	0

label

feature vector

Components of Learning

Formalization:

- Input: $\boldsymbol{x} \in \mathbb{R}^d$ (Email Features)
- Output: $y = \{0,1\}$ (Email Classes)
  - 0 for Not-spam and 1 for Spam
- Target function: $f: \mathcal{X} \rightarrow \mathcal{Y}$ (Ideal Spam Filter → unknown)
- Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ (Dataset of Spam/Not-Spam emails)

Training ↓

Supervised Learning

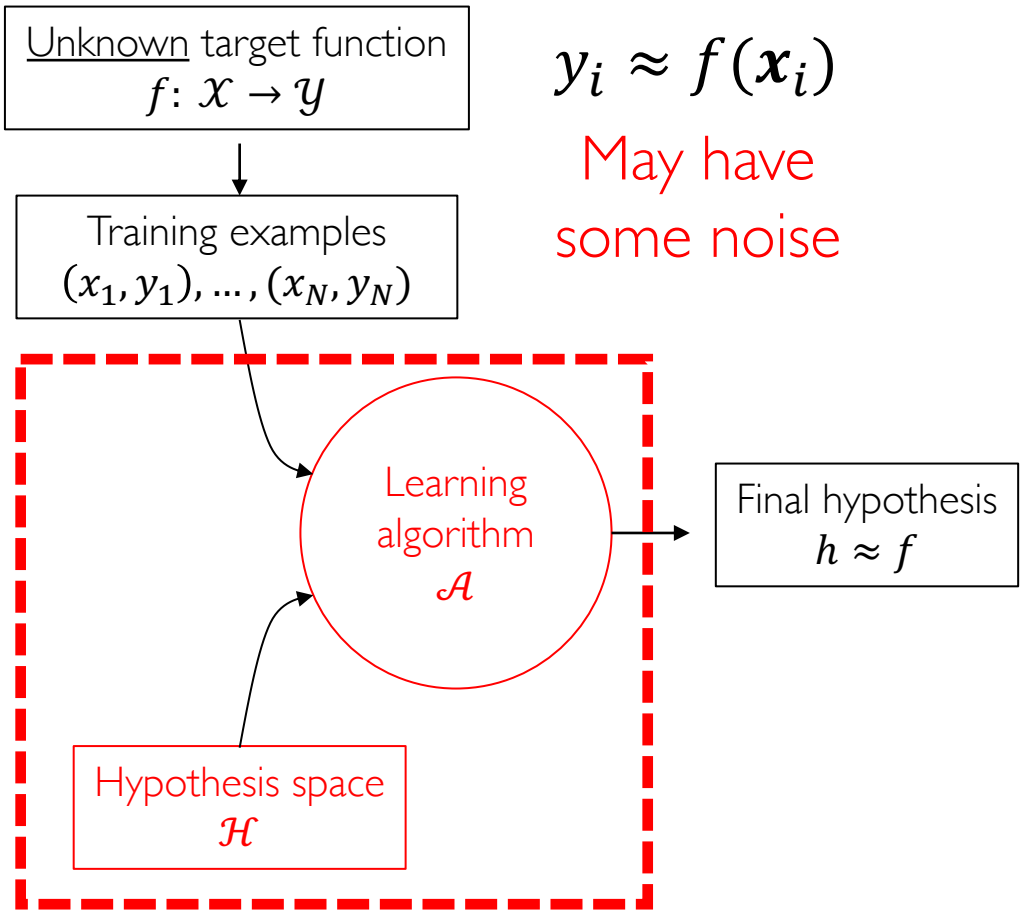
- Hypothesis (假设): $h: \mathcal{X} \rightarrow \mathcal{Y}$ (Formula to be used for prediction)

Components of Learning

- The two core components of the learning problem:
  - The hypothesis space

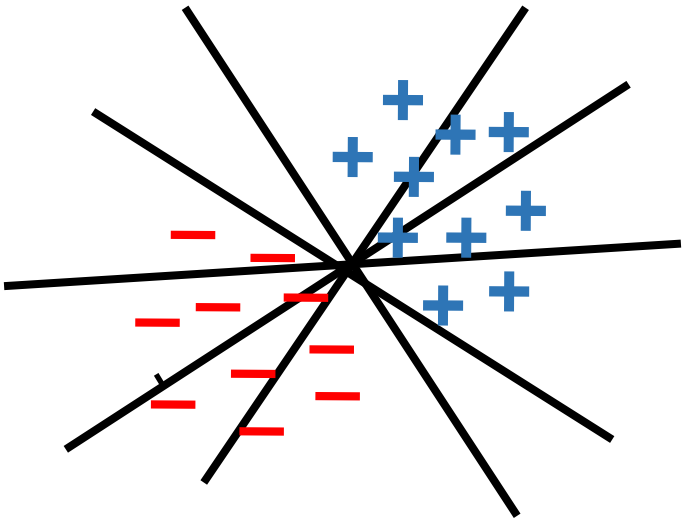
    $\mathcal{H} = \{h\} \quad h \in \mathcal{H}$

  - The learning algorithm $\mathcal{A}$
    - To find the best h
- They are together referred to as the learning model



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:

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

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

- 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?

Machine Learning Scenarios

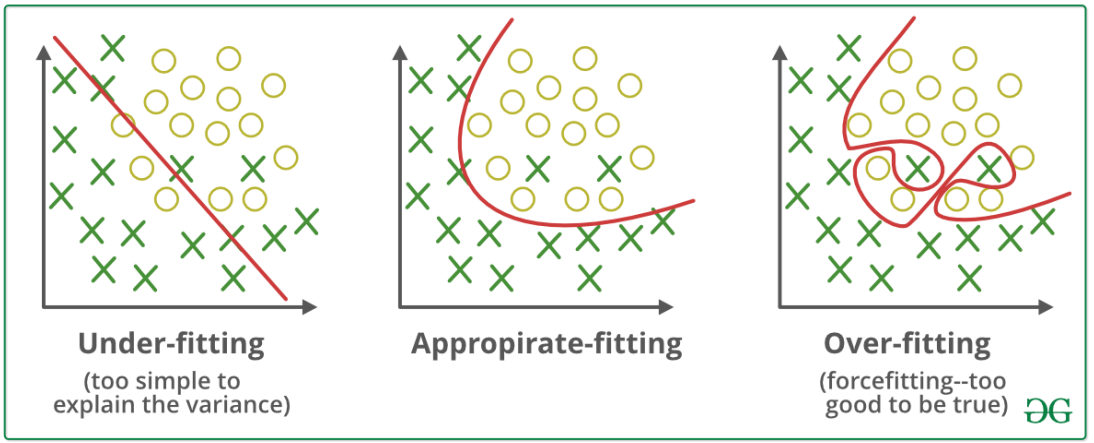
	Supervised Learning	Unsupervised Learning
Discreet Output	Classification (分类)	Clustering (聚类)
Continuous Output	Regression (回归)	Embedding (嵌入)

Supervised Learning

Known y at training

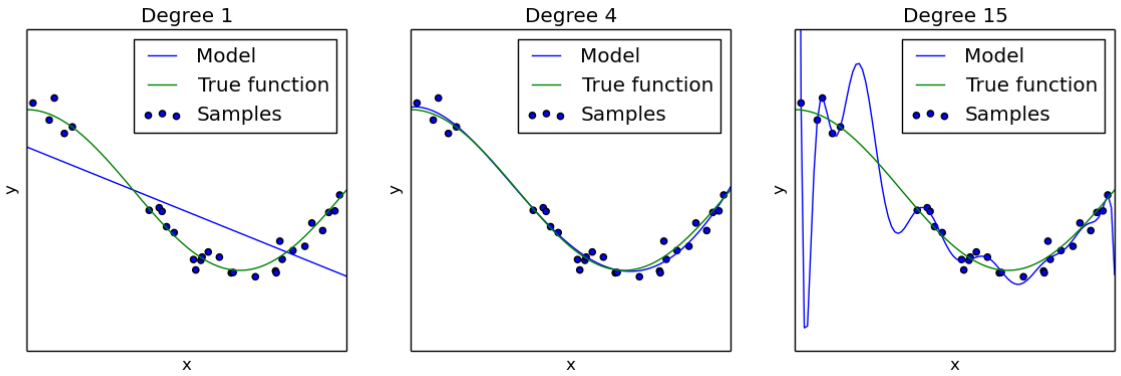
Learner: $h(\boldsymbol{x}) = y$

- Classification:



Most successful in practice

- Regression:

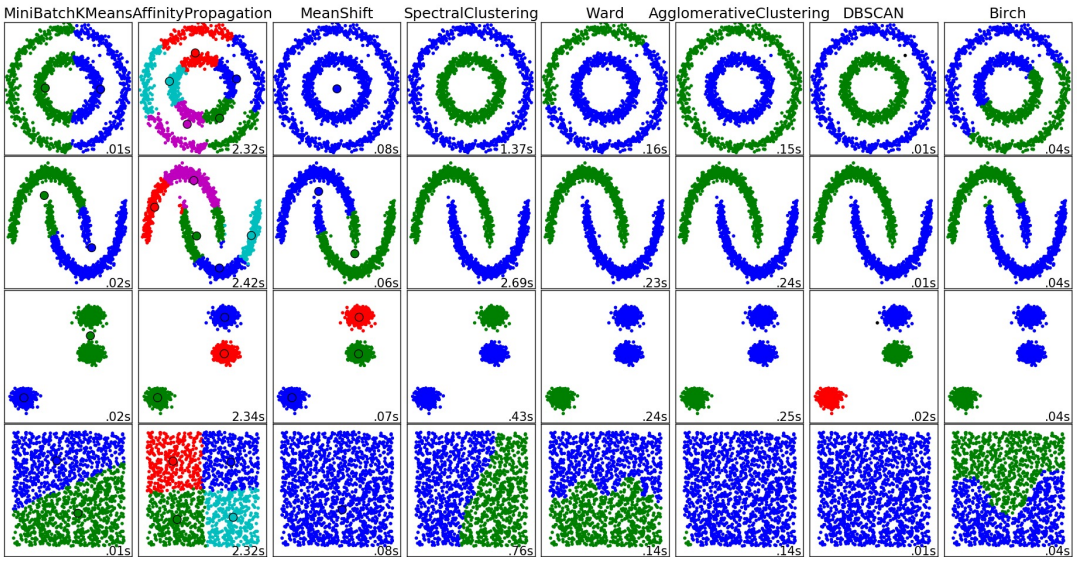


Unsupervised Learning

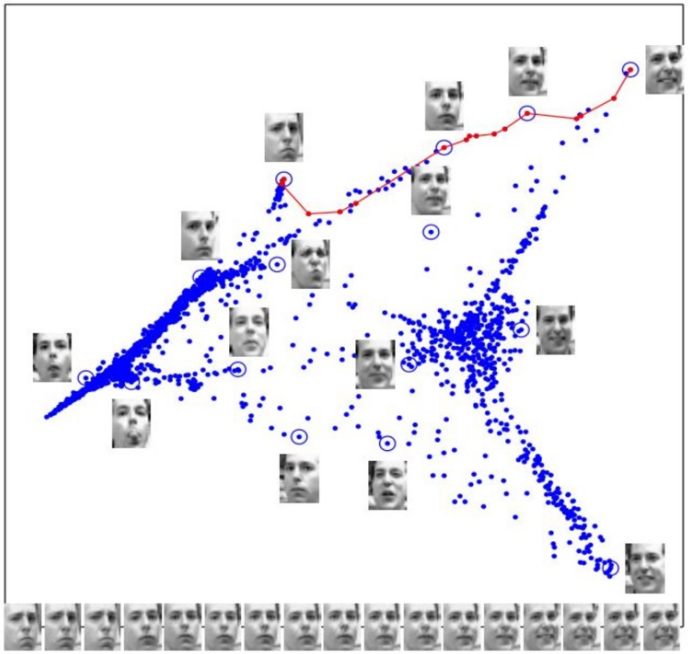
Learner: $h(\boldsymbol{x}) = y$

Unknown y at training

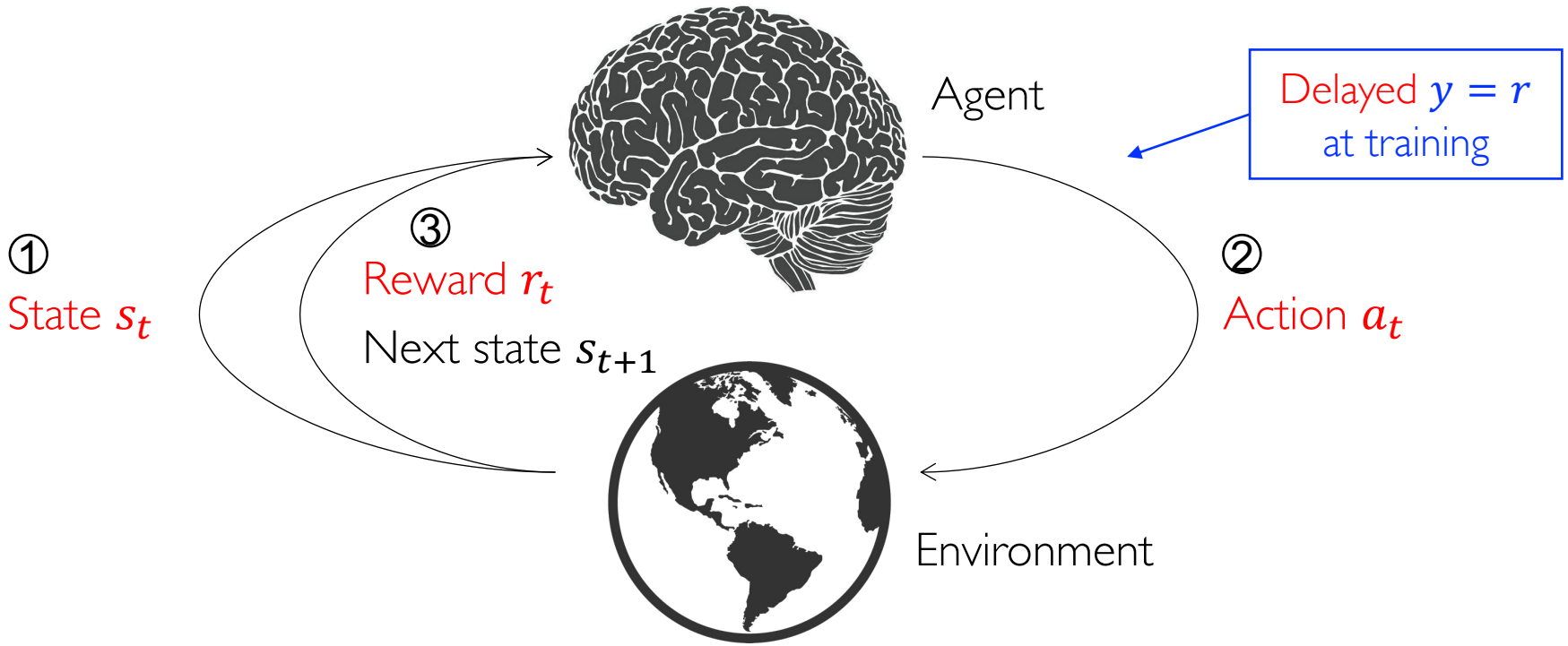
- Clustering:



- Embedding



Reinforcement Learning



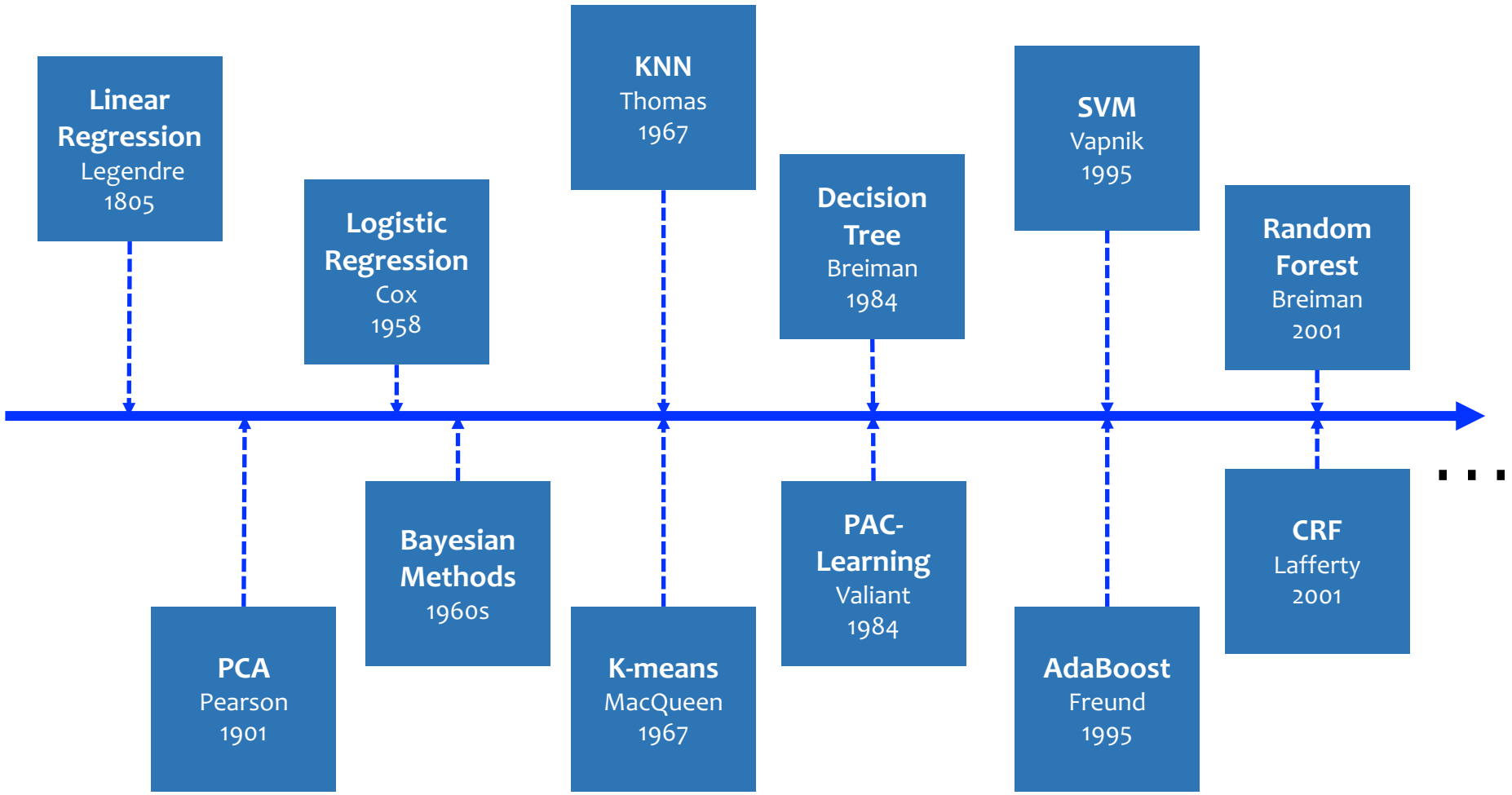
At each step t the agent:

- Executes action a_t
- Receives state s_t
- Receives scalar reward r_t

At each step t the environment:

- Receives action a_t
- Emits state s_{t+1}
- Emits scalar reward r_{t+1}

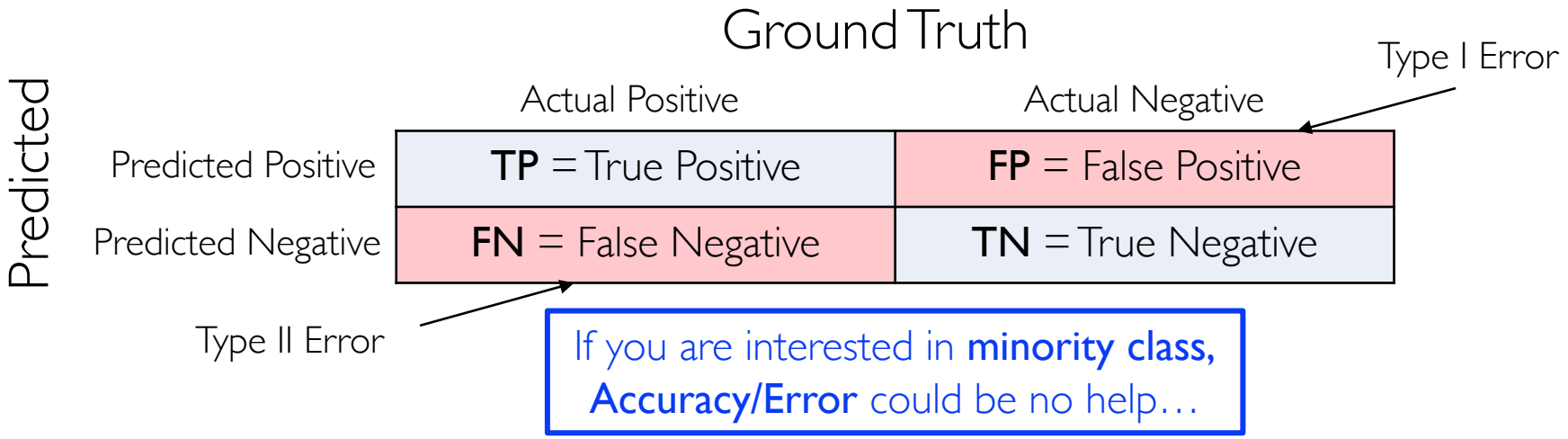
Milestones of Machine Learning



Outline

- **Machine Learning**
  - Framework
  - **Evaluation Metrics**
  - Model Selection
- K-Nearest Neighbors (KNN)
- Linear Regression (LR)
  - Optimization
  - Nonlinearization
  - Regularization

Classification: Precision, Recall, Accuracy

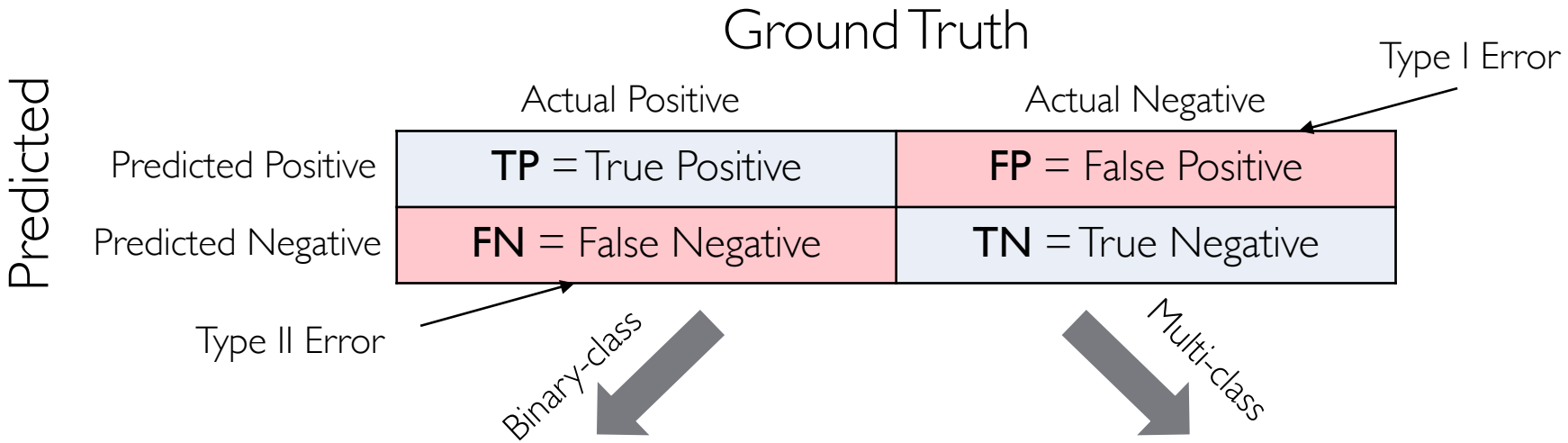


- Precision = $\frac{TP}{TP+FP}$: How many selected items are relevant
- Recall = $\frac{TP}{TP+FN}$: How many relevant items are selected
- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$: How many items are hit
- Error = 1 - Accuracy: How many items are missed

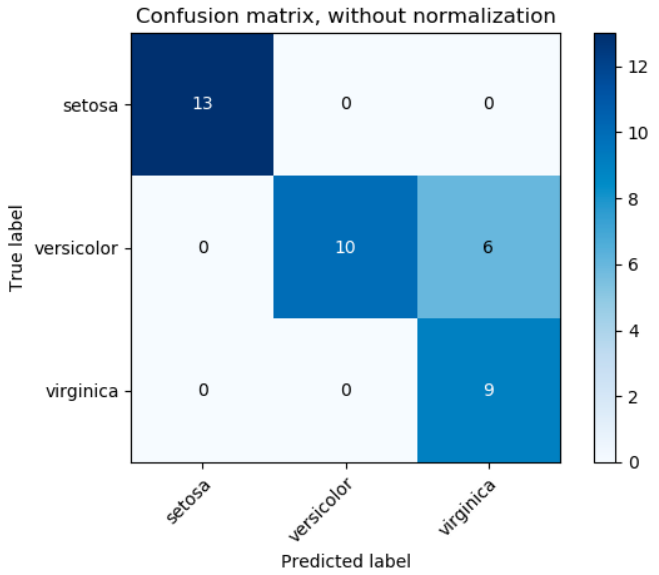
F_1-score:

$$\frac{1}{F_1} = \frac{1}{2} \cdot \left(\frac{1}{P} + \frac{1}{R}\right)$$

Classification: Confusion Matrix




```
>>> from sklearn.metrics import
confusion_matrix
>>> y_true = [0, 1, 1, 0, 1]
>>> y_pred = [0, 0, 1, 1, 1]
>>> confusion_matrix(y_true, y_pred)
array([[1, 1],
       [1, 2]])
```



https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

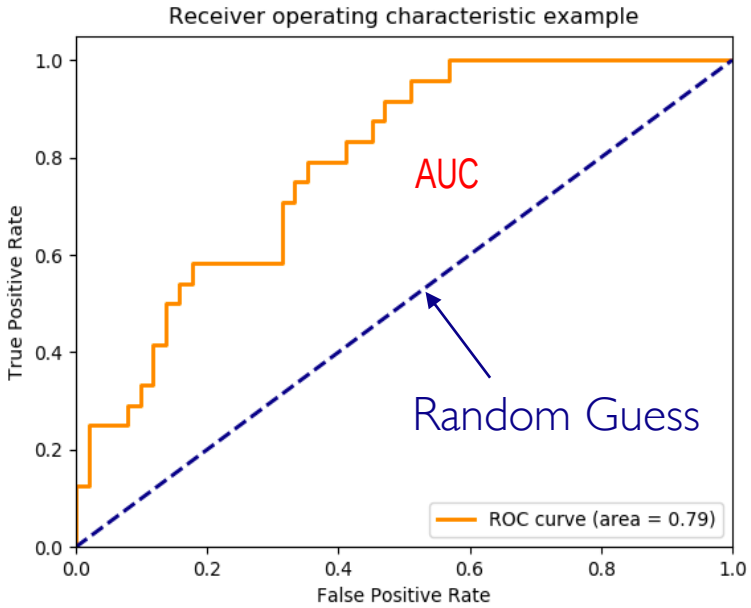
Classification: ROC and AUC

- Receiver Operating Characteristic
  - Take True Negatives into account, which P-R curve does not cover
  - $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$
  - Sensitivity = Recall = TPR = $P(\hat{Y} = 1 | Y = 1)$
  - Specificity = 1 – FPR = $P(\hat{Y} = 0 | Y = 0)$

Irrelevant to $P(Y)$, these metrics perform stable and are robust to imbalanced data

- Area Under the Curve
  - Similar definition to mAP
    - Check it yourself.
  - The larger, the better

Given a random pair of samples (x_+, x_-), their scores yielded by classifier f are (y_+, y_-), then the probability $P(y_+ > y_-)$ equals to AUC.



Regression: Error

Sensitive to outliers

Mean

Median

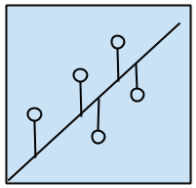
Robust to outliers

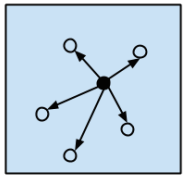
Method	Formula	Traits
Squared Error	$(y_i - \hat{y}_i)^2$	More sensitive to outliers, usually use RMSE
Absolute Error	$\lvert y_i - \hat{y}_i \rvert$	Corresponding to the expected value of L_1 loss
Squared Logarithmic Error	$(\log(1 + y_i) - \log(1 + \hat{y}_i))^2$	Useful when targets having exponential growth, e.g. population counts
Absolute Percentage Error	$\left\lvert \frac{y_i - \hat{y}_i}{y_i} \right\rvert$	Samples with $\left\lvert \frac{y_i - \hat{y}_i}{y_i} \right\rvert > 1$ can be regarded as outliers; fails when $y_i = 0$

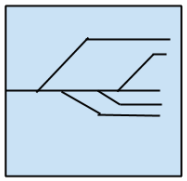
Outline

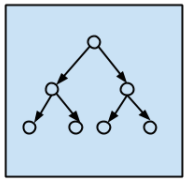
- **Machine Learning**
  - Framework
  - Evaluation Metrics
  - **Model Selection**
- K-Nearest Neighbors (KNN)
- Linear Regression (LR)
  - Optimization
  - Nonlinearization
  - Regularization

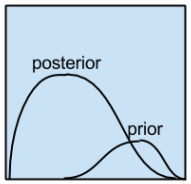
Why Model Selection


Regression Algorithms

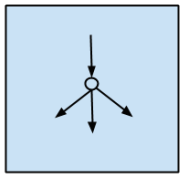

Instance-based Algorithms

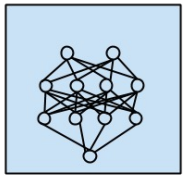

Regularization Algorithms

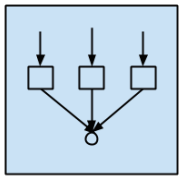

Decision Tree Algorithms



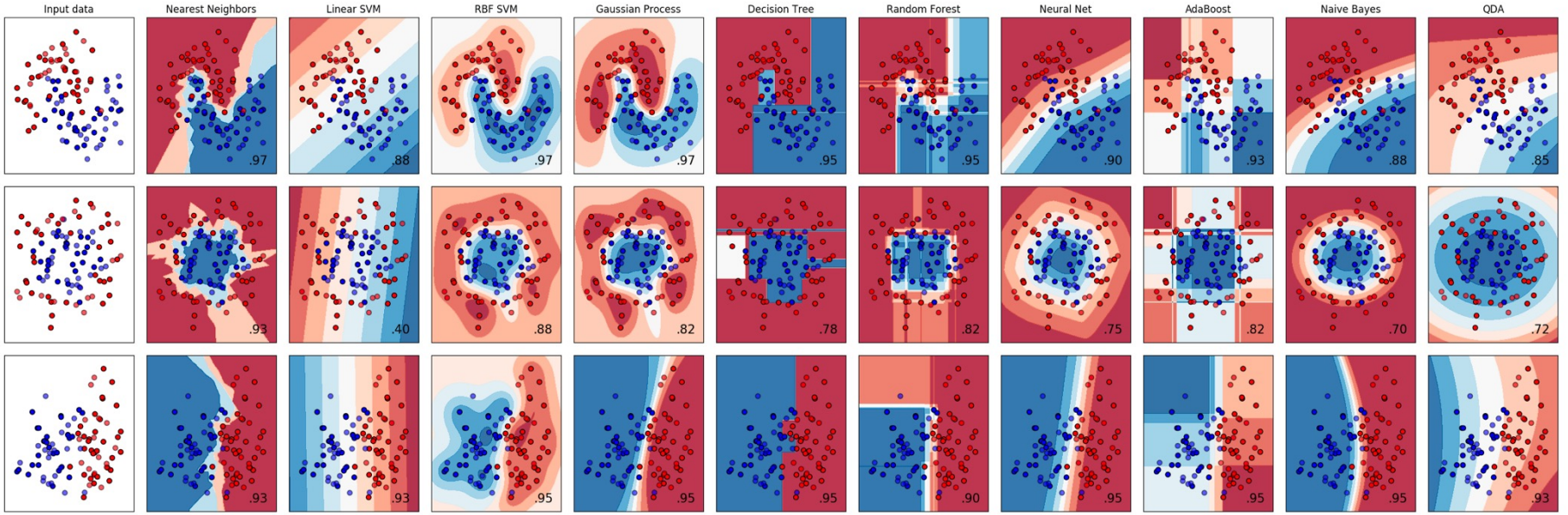
Bayesian Algorithms


Artificial Neural Network Algorithms


Deep Learning Algorithms

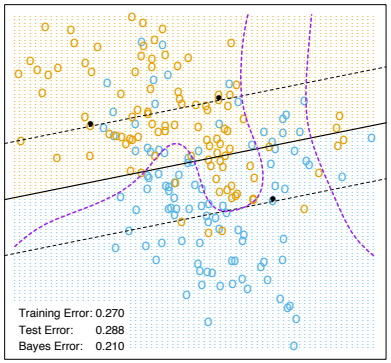

Ensemble Algorithms

All models are wrong, but some are useful.
-G. E. P. Box

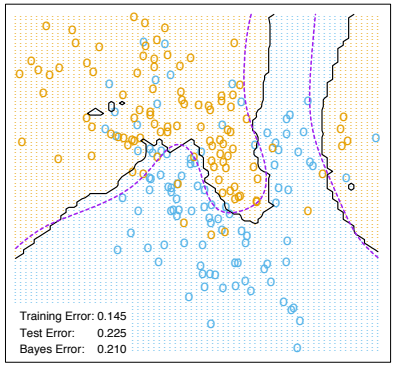


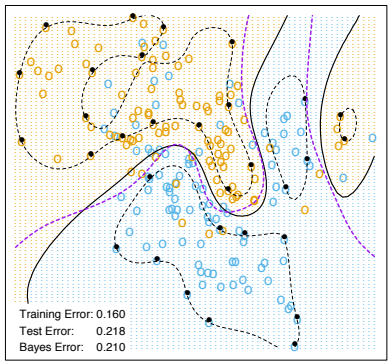
Model Selection

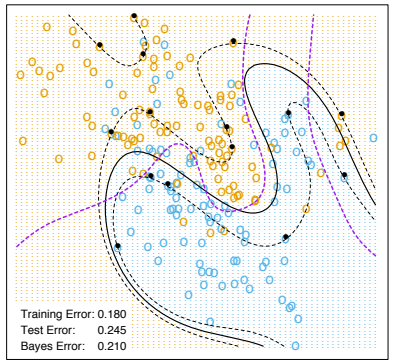
- There are too many candidate learning algorithms or models.
  - Each of which can be fit well to data and used for prediction.



$C = 10000$







Different Algorithms

Linear Classifier K-nearest-neighbor

Different Hyperparameters

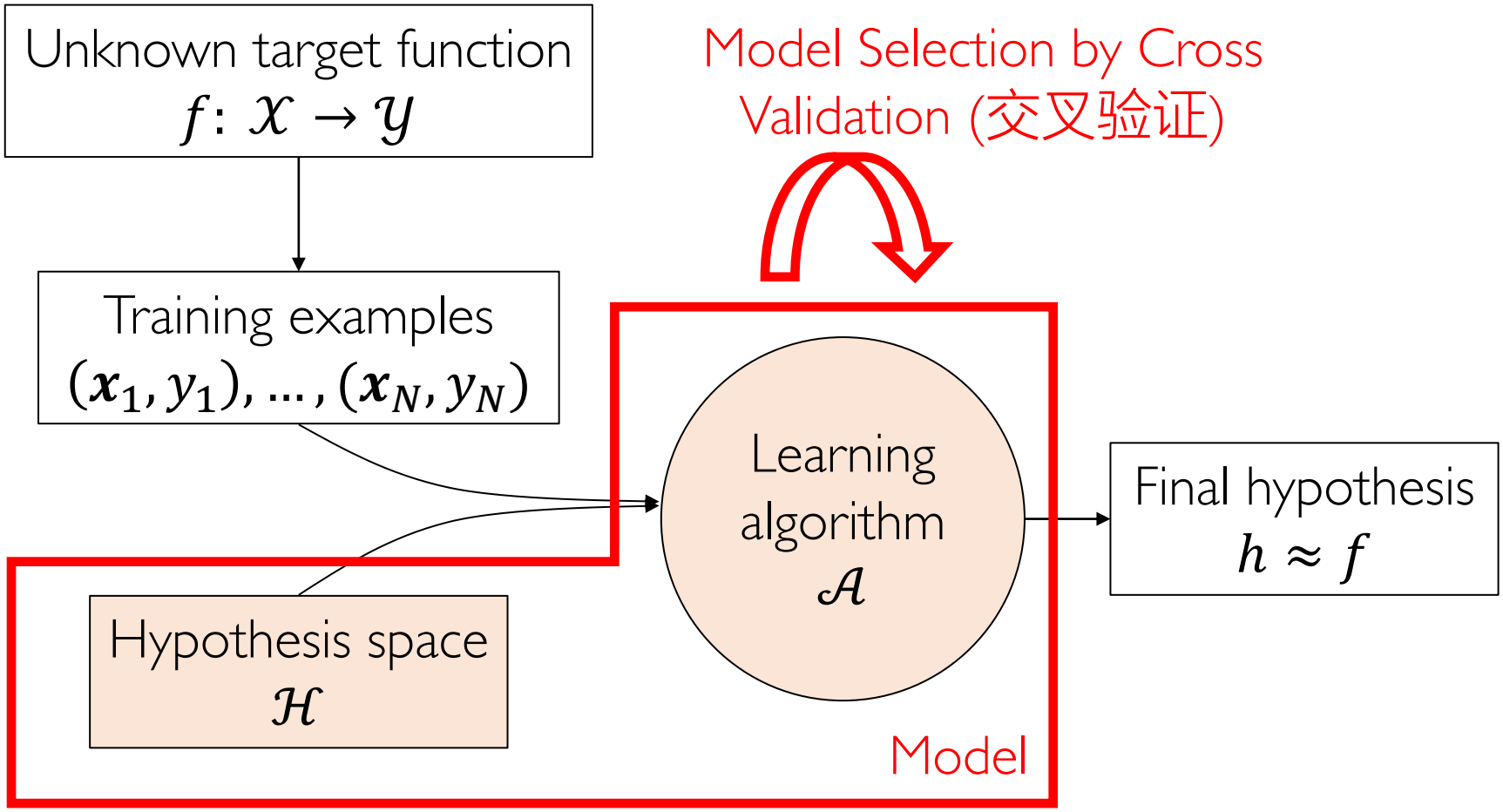
Kernel SVM (same algorithm)

  - How can we decide which is best?
  - Which part of features should we use?

Tuner?
(调参侠)

- We need model selection.

Components of Learning



Training and Test Data

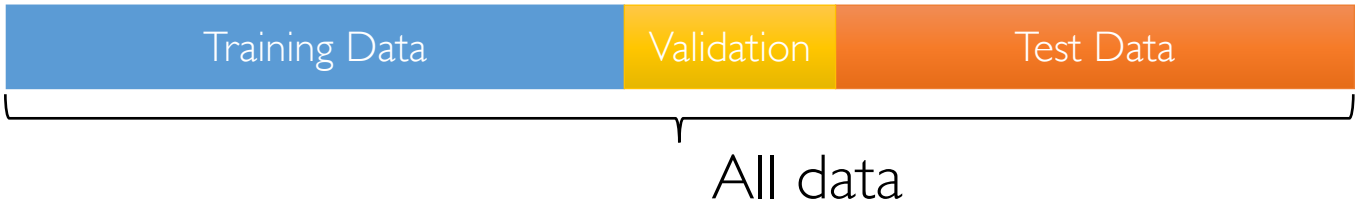
- How can we decide which algorithm or model is best?

- Train the parameters θ of each model h_θ from the training data.
- Evaluate on the test data, and pick the best performer.
- Problems:
  - Over-estimates the test performance ("lucky" model)
  - This is the most common but wrong practice of machine learning.

The learning algorithms should never ever have access to test data!

Training and Test Data

- Reserve some data for validation.



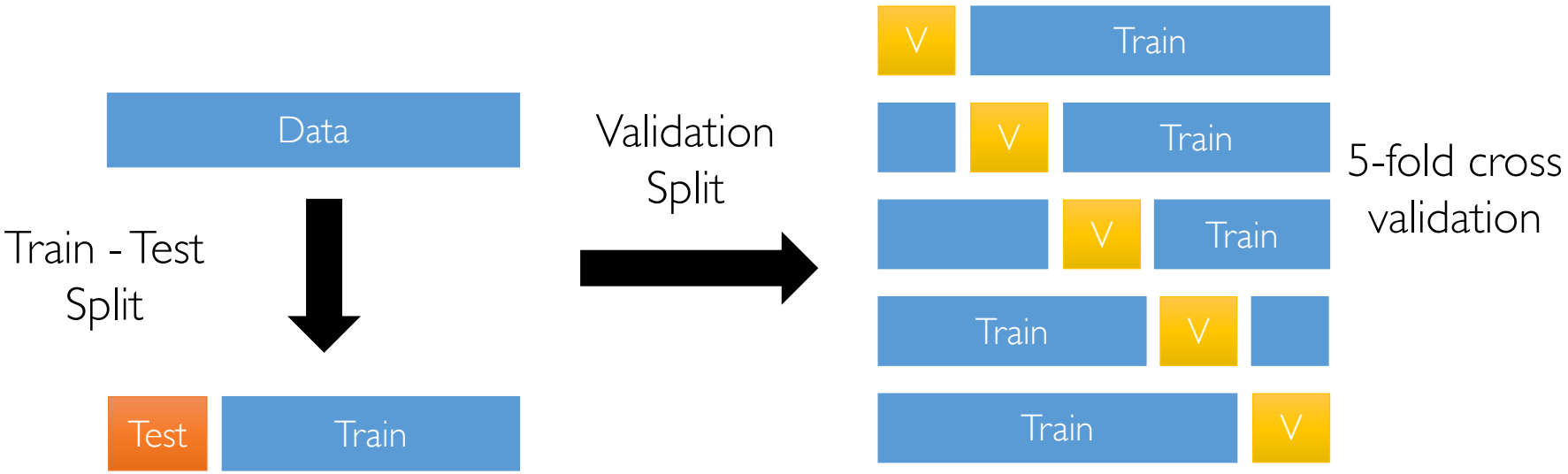
- Train the parameters θ of each model h_θ from the training data.
- Evaluate on the validation data, and pick the best performer.
- Reserve test data to benchmark the chosen model.
- Problems:
  - Wasteful of training data (learning cannot use validation data).
  - May bias the selection towards overly simple models.

The learning algorithms should never ever have access to test data!

Cross Validation

- What if we get an unfortunate split?
- K-fold Cross Validation:
  - Split the data set $\mathcal{D}$ into K subsets $\mathcal{D}^{(i)}$ (called folds)
  - For $i = 1, \ldots, K$, train $h^{(i)}$ using all data but the i-*th* fold ($\mathcal{D} \backslash \mathcal{D}^{(i)}$).
  - Report cross-validation error by averaging all val errors $\hat{\epsilon}_{\mathcal{D}^{(i)}}(h^{(i)})$

Standard in practice



Model Selection: Whole Procedure

- Combined Algorithm Selection & Hyperparameter (**CASH**) optimization problem
  - A set of algorithms $\mathcal{A} = \{A^{(1)}, \ldots, A^{(n)}\}$
  - Denote the hyperparameter space of algorithm $A^{(i)}$ as $\boldsymbol{\Lambda}^{(i)}$
  - Denote the error of $A^{(i)}$ as $\mathcal{L}\left(A_{\lambda}^{(i)}, D_{\mathbf{train}}, D_{\mathbf{valid}}\right)$ using $\lambda \in \boldsymbol{\Lambda}^{(i)}$ trained on $D_{\mathbf{train}}$ and evaluated on $D_{\mathbf{valid}}$
  - The problem is to find optimal algorithm and its hyperparameter:

$$A_{\lambda^*}^{*} = \underset{A^{(i)} \in \mathcal{A}, \lambda \in \boldsymbol{\Lambda}^{(i)}}{\operatorname{argmin}} \mathcal{L}\left(A_{\lambda}^{(i)}, D_{\mathbf{train}}, D_{\mathbf{valid}}\right)$$

AutoML

Model Selection Too Slow

- The procedure to select both algorithms and hyperparameters

$$A^*_{\lambda^*} = \underset{A^{(i)} \in \mathcal{A}, \lambda \in \mathbf{\Lambda}^{(i)}}{\operatorname{argmin}} \mathcal{L}\left(A^{(i)}_{\lambda}, D_{\text{train}}, D_{\text{valid}}\right)$$

- The overall complexity (with K-fold cross validation)

Tuner!
(调参侠)

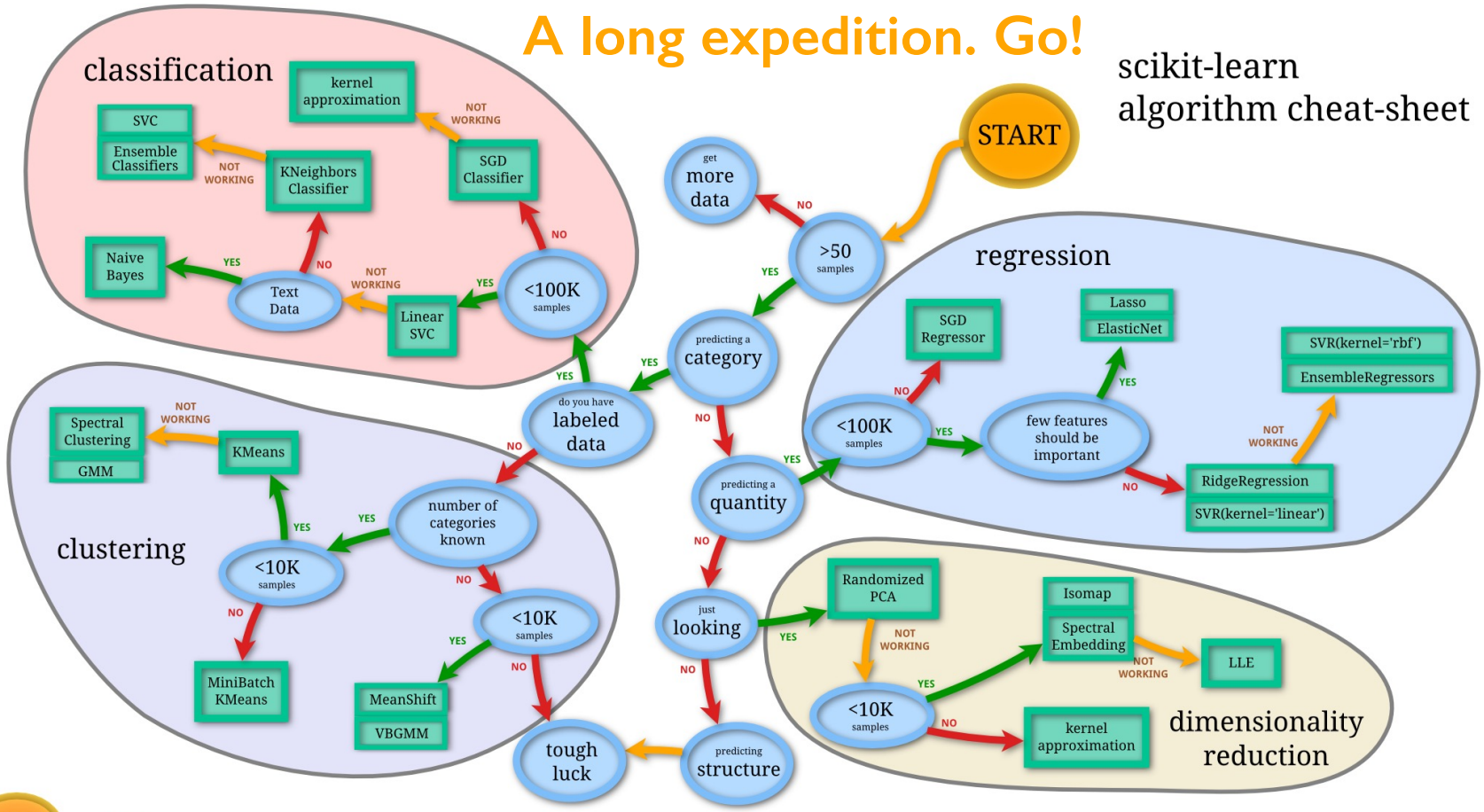
$$\textbf{Complexity} = \sum_{A^{(i)} \in \mathcal{A}} \left|\mathbf{\Lambda}^{(i)}\right| \cdot K \cdot O(A^{(i)})$$

How many algorithms to consider?

How many hyperparameter choices of each algorithm?

Complexity of the learning algorithm

How to Do ML Efficiently



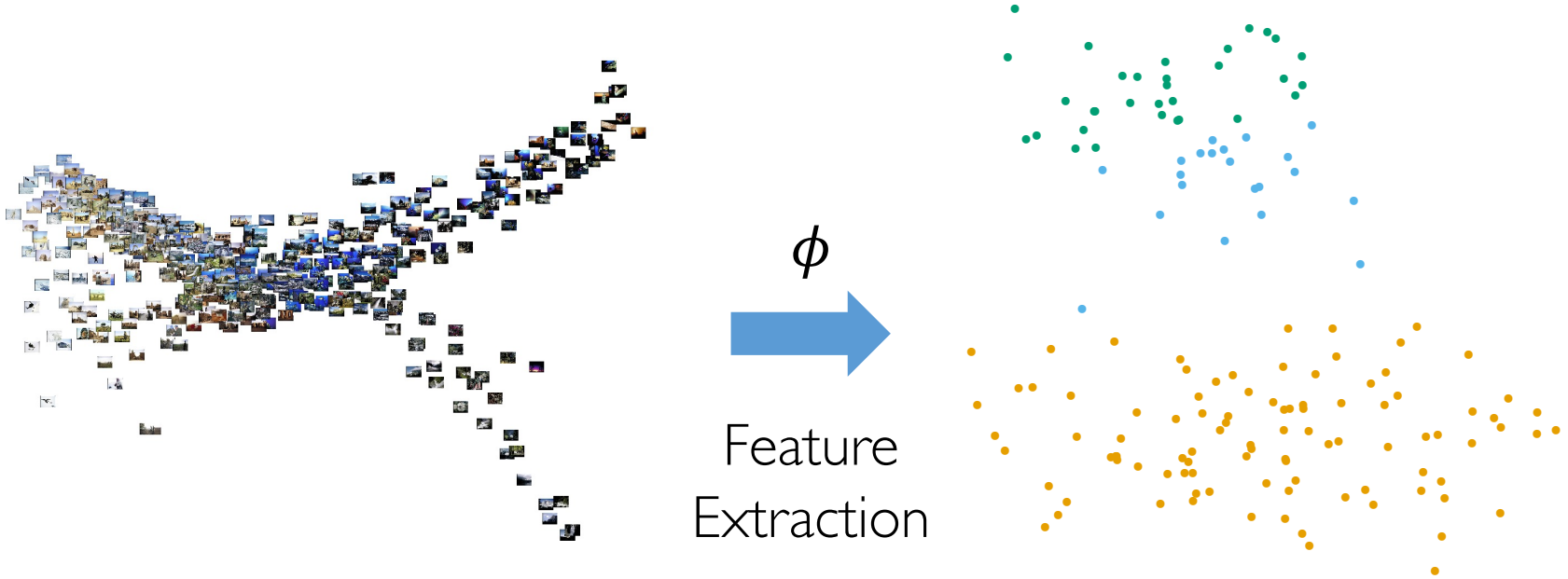
You need to know them all

https://scikit-learn.org/

Outline

- Machine Learning
  - Framework
  - Evaluation Metrics
  - Model Selection
- **K-Nearest Neighbors (KNN)**
- Linear Regression (LR)
  - Optimization
  - Nonlinearization
  - Regularization

Machine Learning: Geometric View

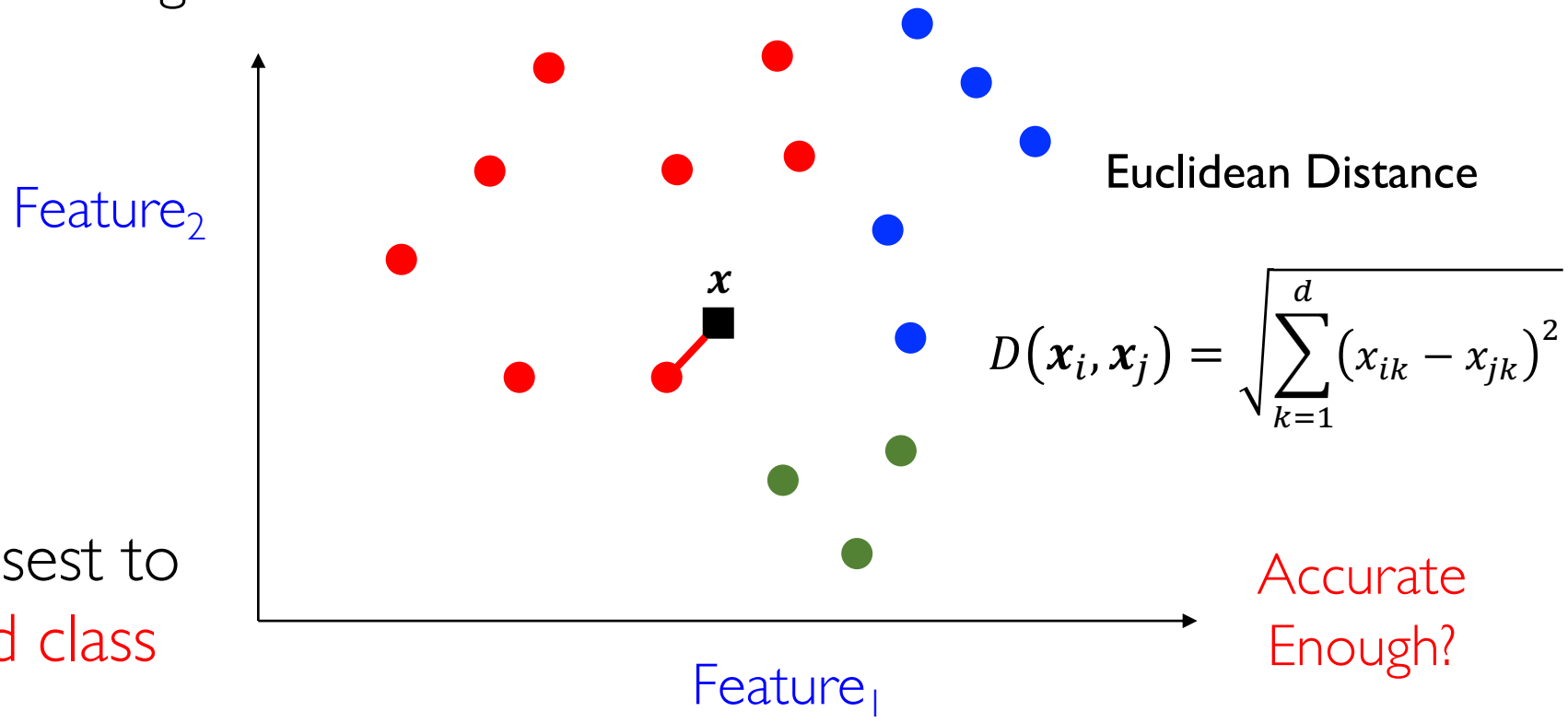


- We can view examples as points in an d-dimensional space where d is the number of features.
- **Assumption:** Closer points in feature space have similar semantics.

Birds of a feather flock together. (物以类聚)

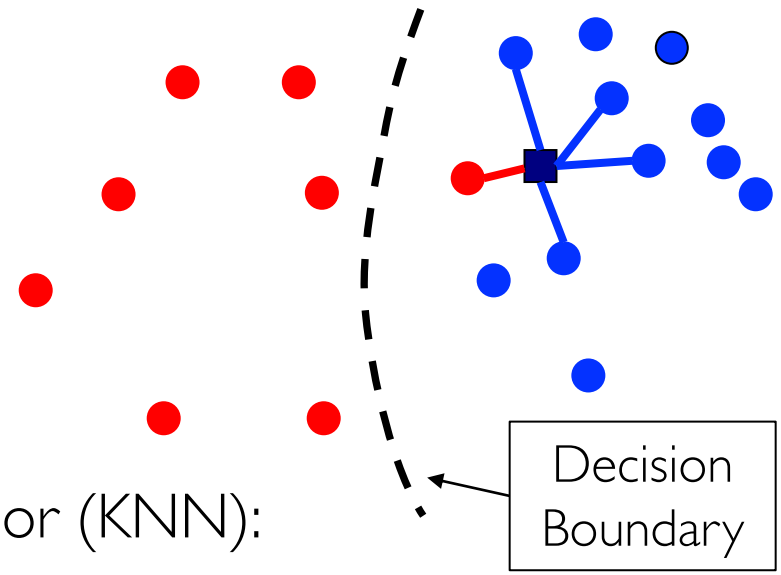
Nearest-Neighbor (1NN)

- To classify a new example $\boldsymbol{x}$:
  - Label $\boldsymbol{x}$ with the label of the closest example to $\boldsymbol{x}$ in the training set.



K-Nearest-Neighbors (KNN)

- $\boldsymbol{x}$ is close to a red point.
- But most of the next closest points are blue.

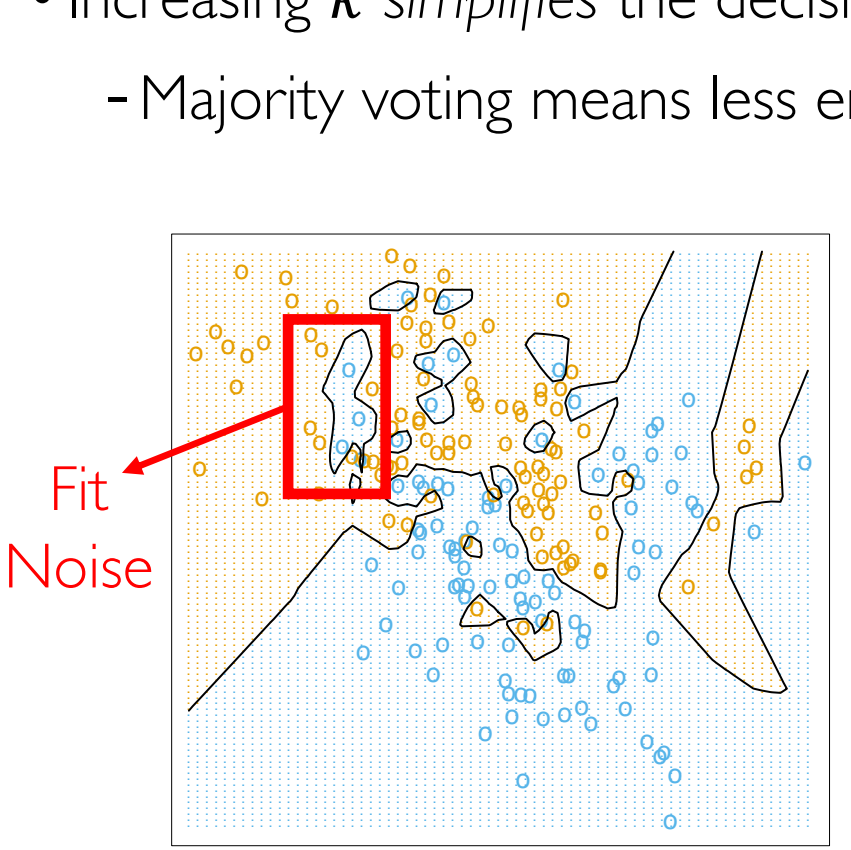


Model-free

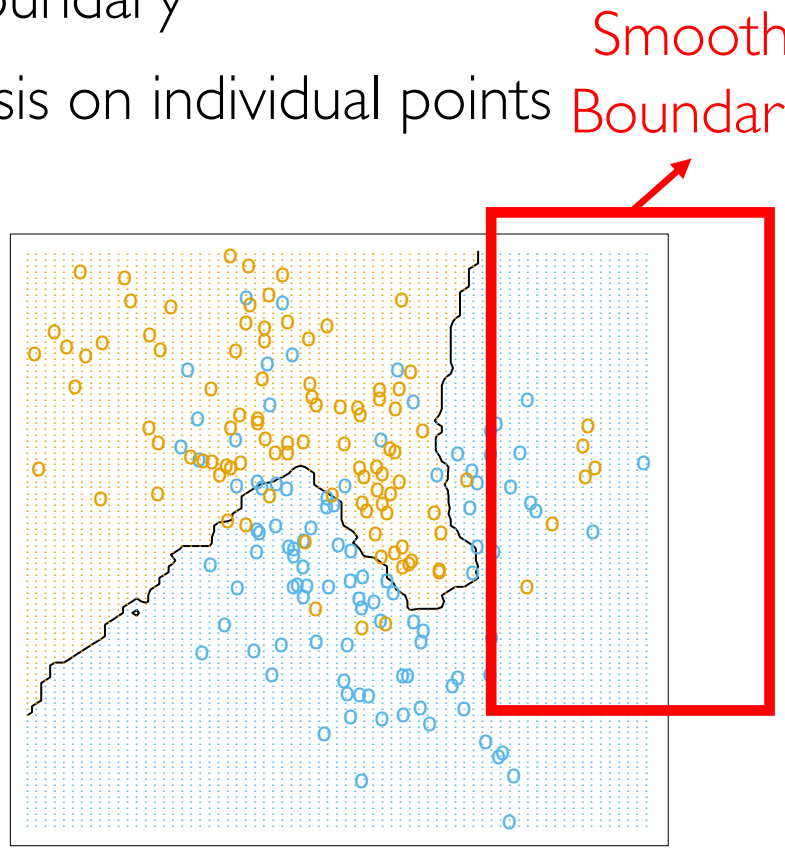
- K-Nearest-Neighbor (KNN):
  - Find $\boldsymbol{k}$ nearest neighbors of $\boldsymbol{x}$.
  - Label $\boldsymbol{x}$ with the majority label within the $\boldsymbol{k}$ nearest neighbors
  - Consider: How can we handle ties for even values of $\boldsymbol{k}$?

The Effect of K

- Increasing k *simplifies* the decision boundary
  - Majority voting means less emphasis on individual points

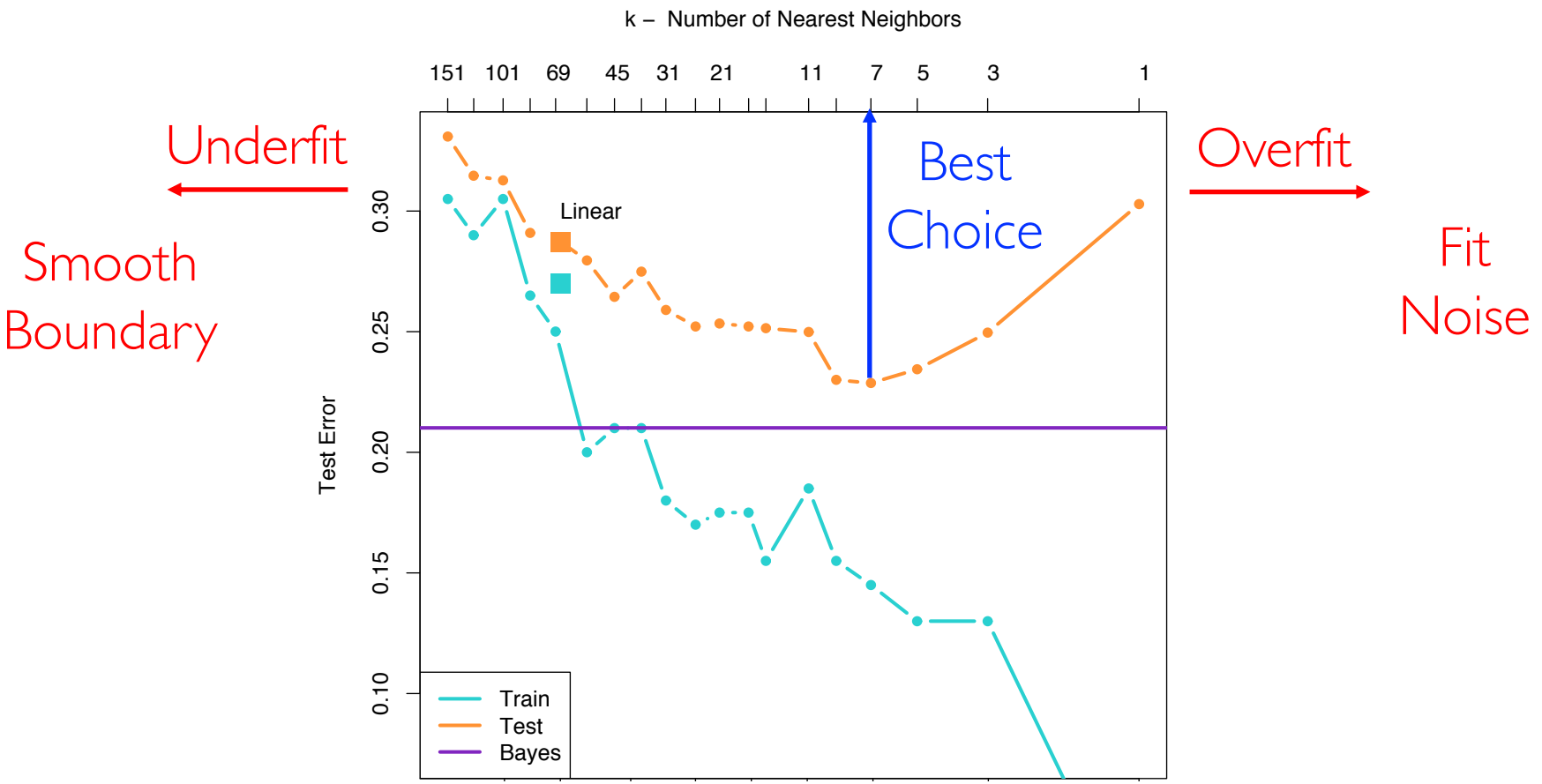


1-Nearest-Neighbor



15-Nearest-Neighbors

The Effect of K

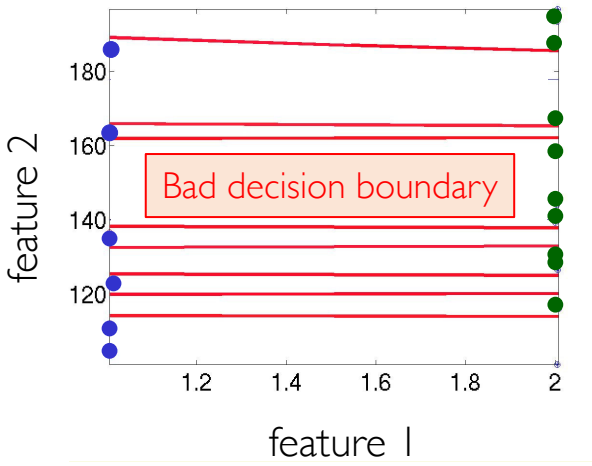


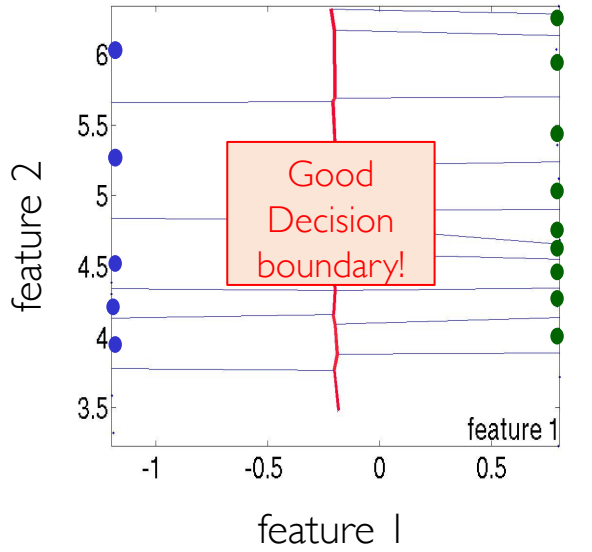
- Choose best k on the validation set. (Often set k as an odd number).

Feature Normalization

- We could normalize each feature to be of mean 0 and variance 1.
- **Z-score normalization:**
  - For each feature dimension j, compute based on its samples:
  - Mean $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$
  - Variance $\sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_j)^2}$
  - Normalize the feature into a new one:

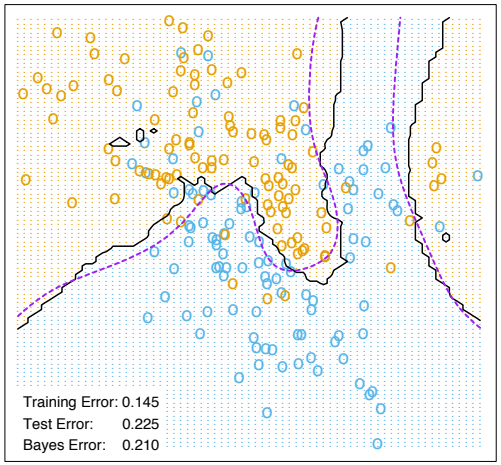
$$\hat{x}_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\sigma_j}$$





KNN Summary

- **When to use:**
  - Few attributes per instance (expensive computation)
  - Lots of training data (curse of dimensionality)
- **Advantages:**
  - Agnostically learn complex target functions
  - Do not lose information (store original data)
  - Data number can be very large (big pro)
  - Class number can be very large (biggest pro)
    - All other ML algorithms may fail here



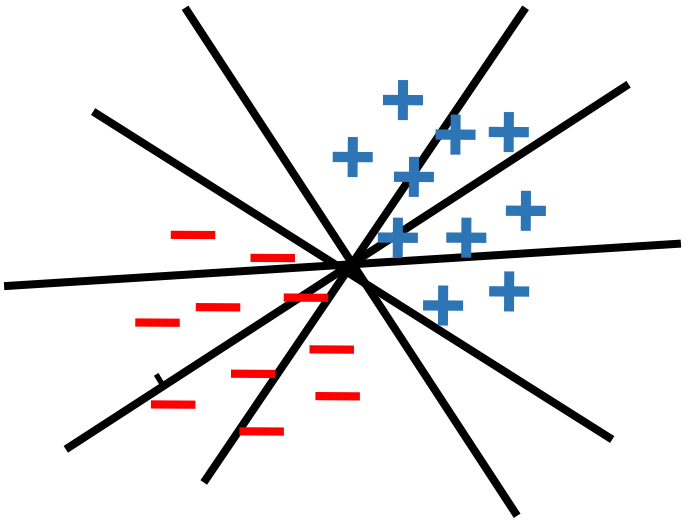
- **Disadvantages:**
  - Slow at inference time (acceleration a must)
  - Fooled easily by irrelevant attributes (feature engineering crucial)

Outline

- Machine Learning
  - Framework
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- **Linear Regression (LR)**
  - Optimization
  - Nonlinearization
  - Regularization

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:

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

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

- 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?

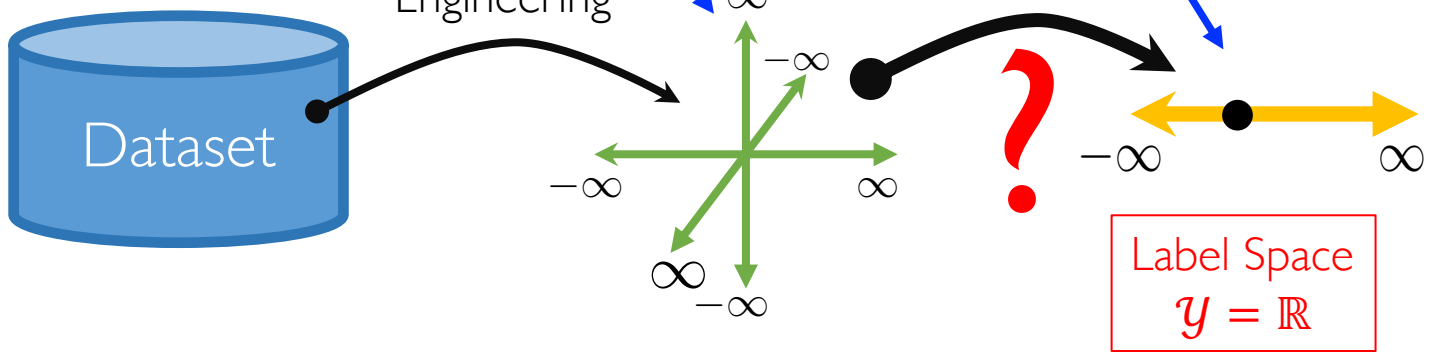
An Example: Electricity Prediction

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

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
	…	…

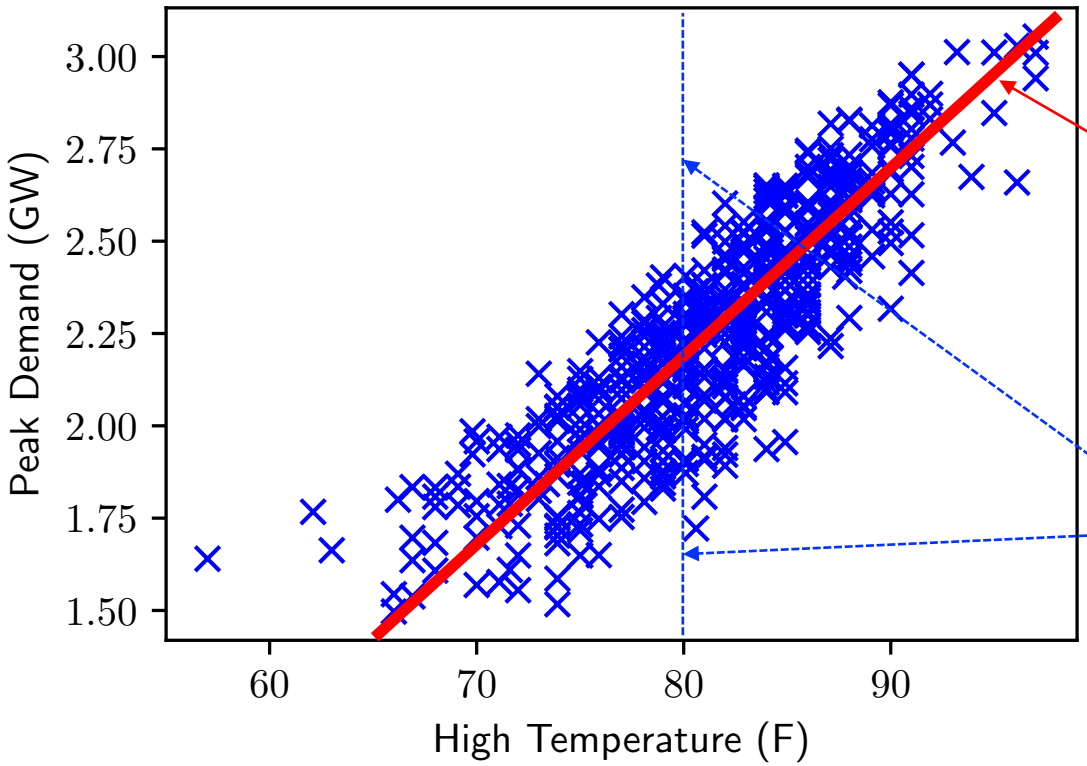


How to map from x to y?

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)

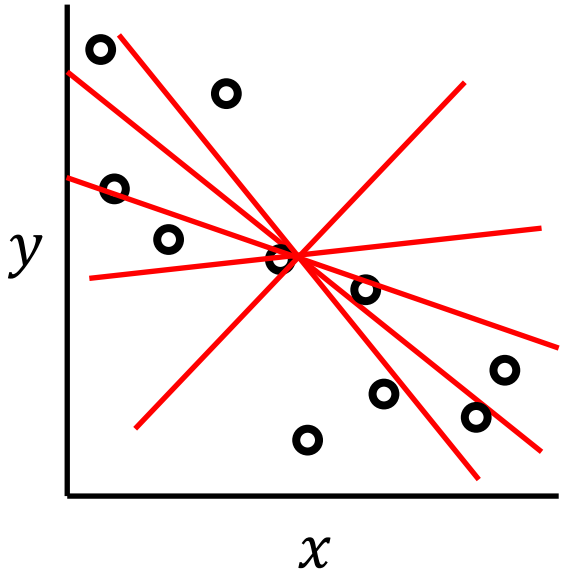


A linear model seems good

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 $\boldsymbol{w}$:

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

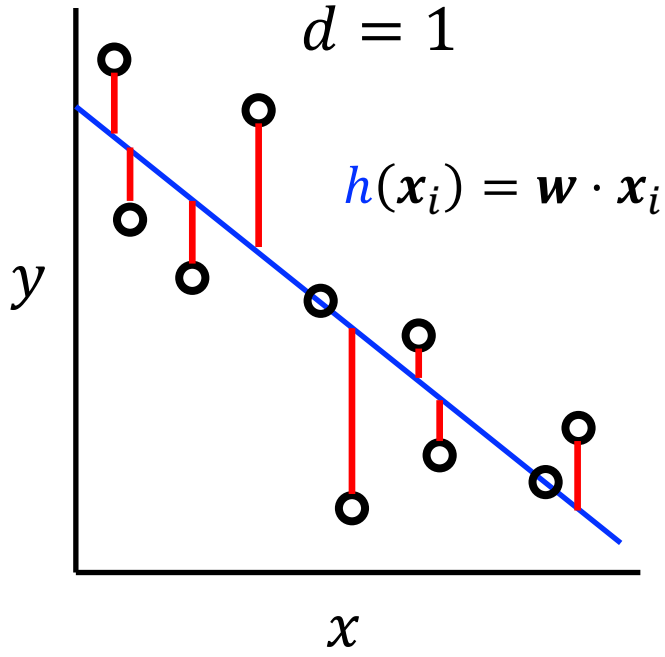
Linear Regression: Loss Function

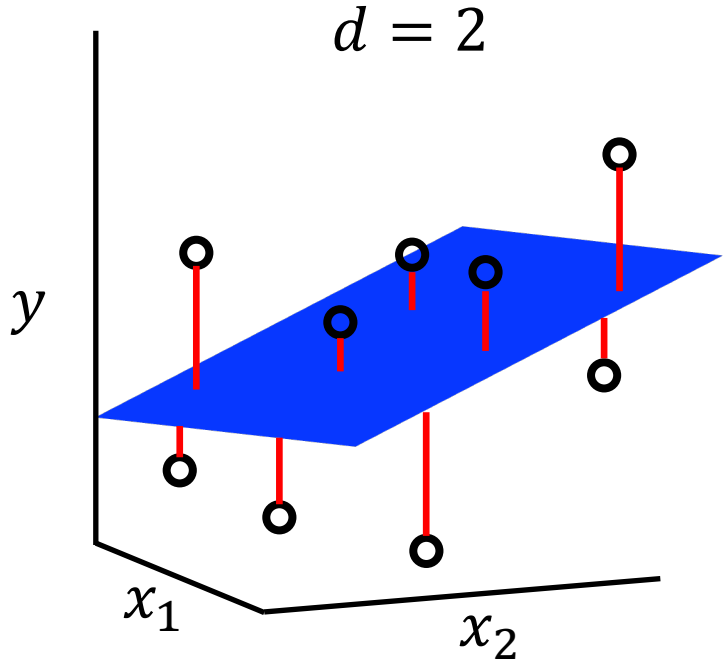
- Use the squared loss (L2 loss) to compute the error on training set:

$\hat{\epsilon}$ standards for *empirical*

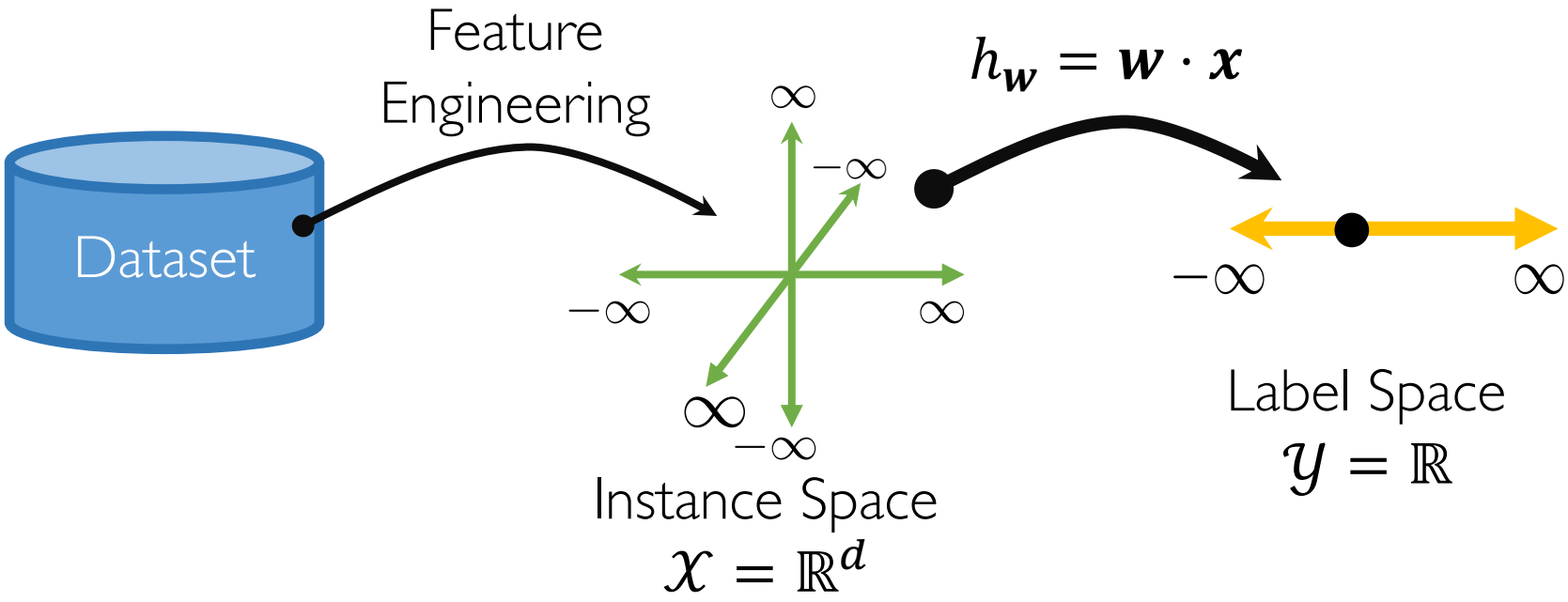
$$\hat{\epsilon}(h) = \sum_{i=1}^{n} (h(\boldsymbol{x}_i) - y_i)^2$$

Training Error





Linear Regression



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

Optimization

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

Outline

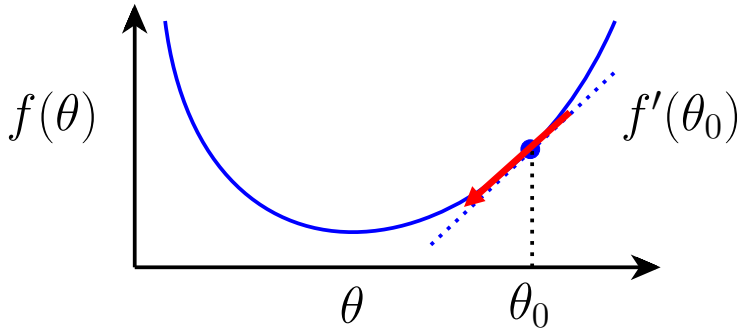
- Machine Learning
  - Framework
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- **Linear Regression (LR)**
  - **Optimization**
  - Nonlinearization
  - Regularization

Optimization

- Closed-form analytical solution 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

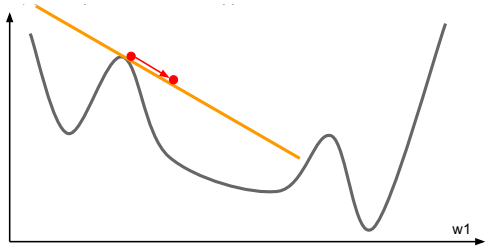
Gradient

- 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:

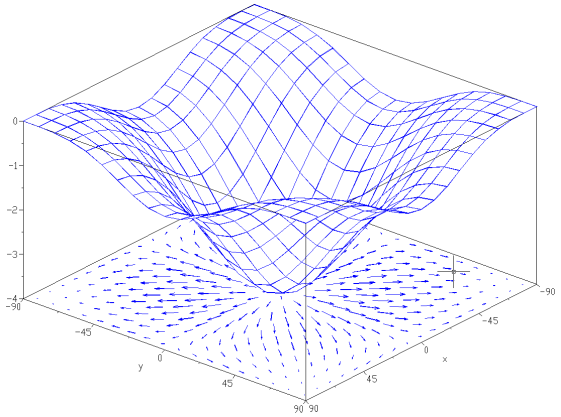
$$J(\boldsymbol{w}) = J(\boldsymbol{w}_0) + (\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{g} + \cdots$$

- 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.

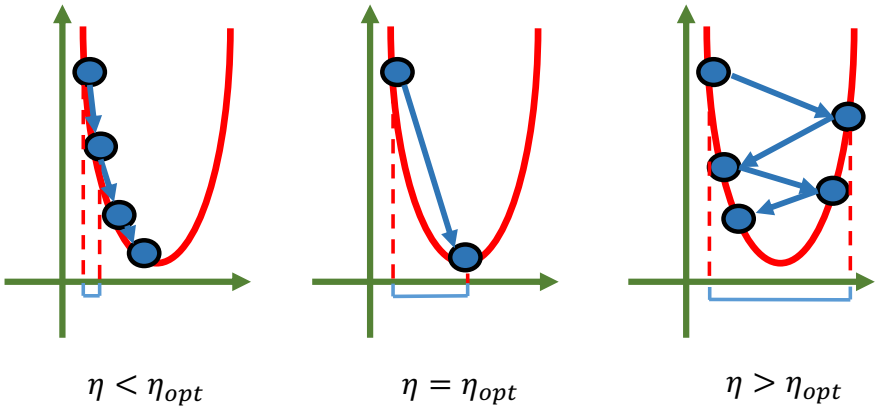
$$-\eta \boldsymbol{g}^T \boldsymbol{g} \leq 0$$



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

Gradient Descent (GD)

- General optimization algorithm
- For each iteration t ($\leq T$)
  - When $\boldsymbol{w}^t$ does not converge:
    - $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}} \hat{\epsilon}(\boldsymbol{w})$



- Learning rate η matters.

Squared Loss is convex.

- Theoretically, convergence rate is $O(1/\sqrt{T})$ under convex condition.
- 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:
    - $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - 2\eta \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w}^t - \boldsymbol{y})$

Complexity (for T iterations):
$O(dnT)$

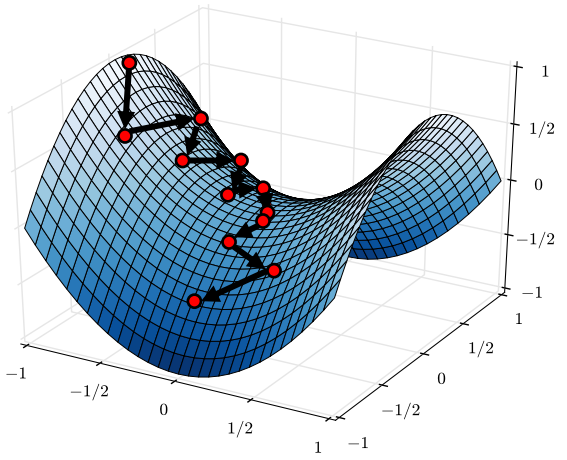
Stochastic Gradient Descent (SGD)

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

$$\Delta^t = \nabla_\theta J^t(\theta^t)$$

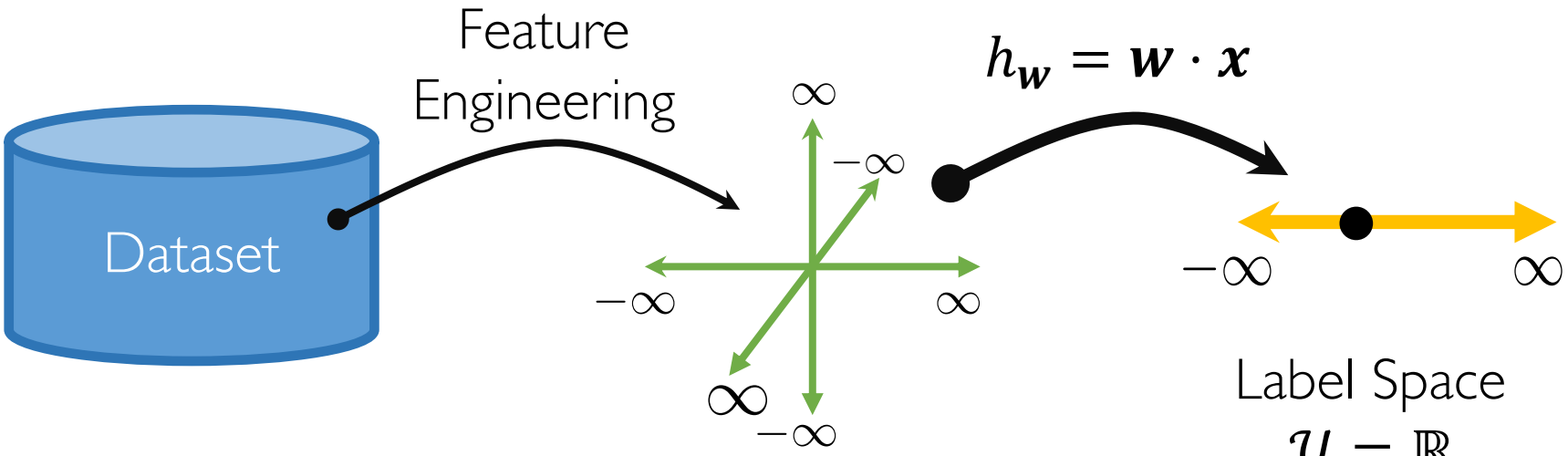
  - Update parameters with learning rate η:

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



- 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 $O(dmT)$. How about large dimension d?
- Theoretically, convergence rate is $O(1/\sqrt{T})$ under convex condition.

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

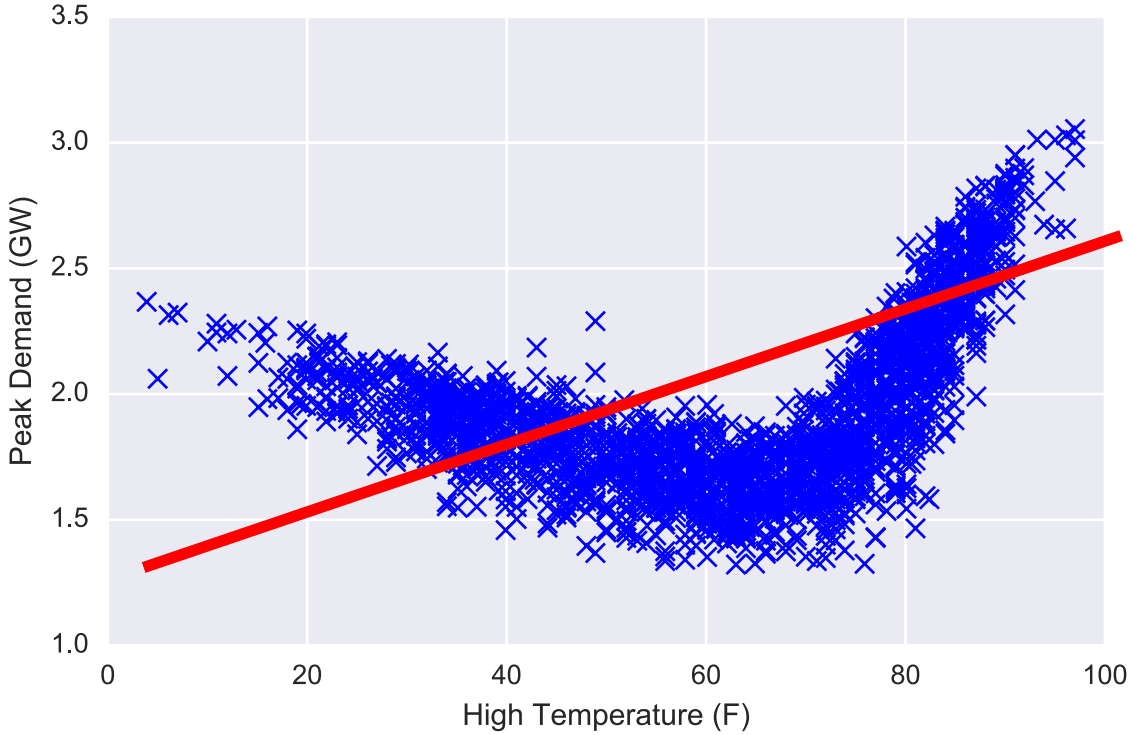
How to handle nonlinearity?

Outline

- Machine Learning
  - Framework
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- **Linear Regression (LR)**
  - Optimization
  - **Nonlinearization**
  - Regularization

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

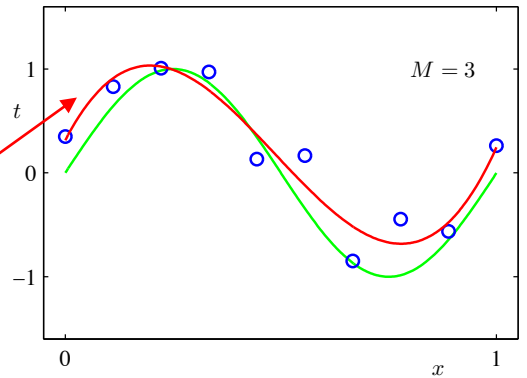
- Feature map: $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X} \xrightarrow{\Phi} \boldsymbol{z} = (z_1, \ldots, z_{\tilde{d}}) \in \mathcal{Z}$

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

- 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

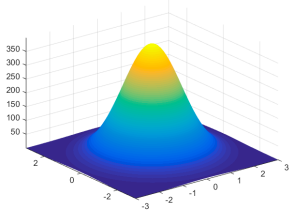
  1-D vector: $\boldsymbol{z}' = (1, x_1, x_1^2, x_1^3)$

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



- Radial basis functions (RBF)

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



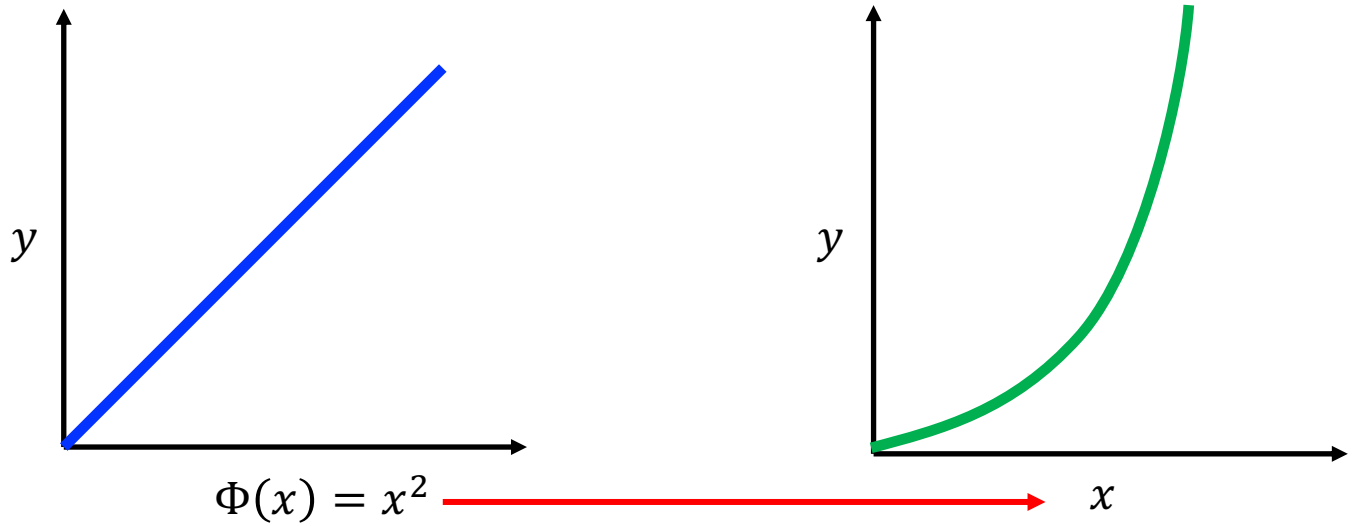
Linear Regression: Basis Function

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

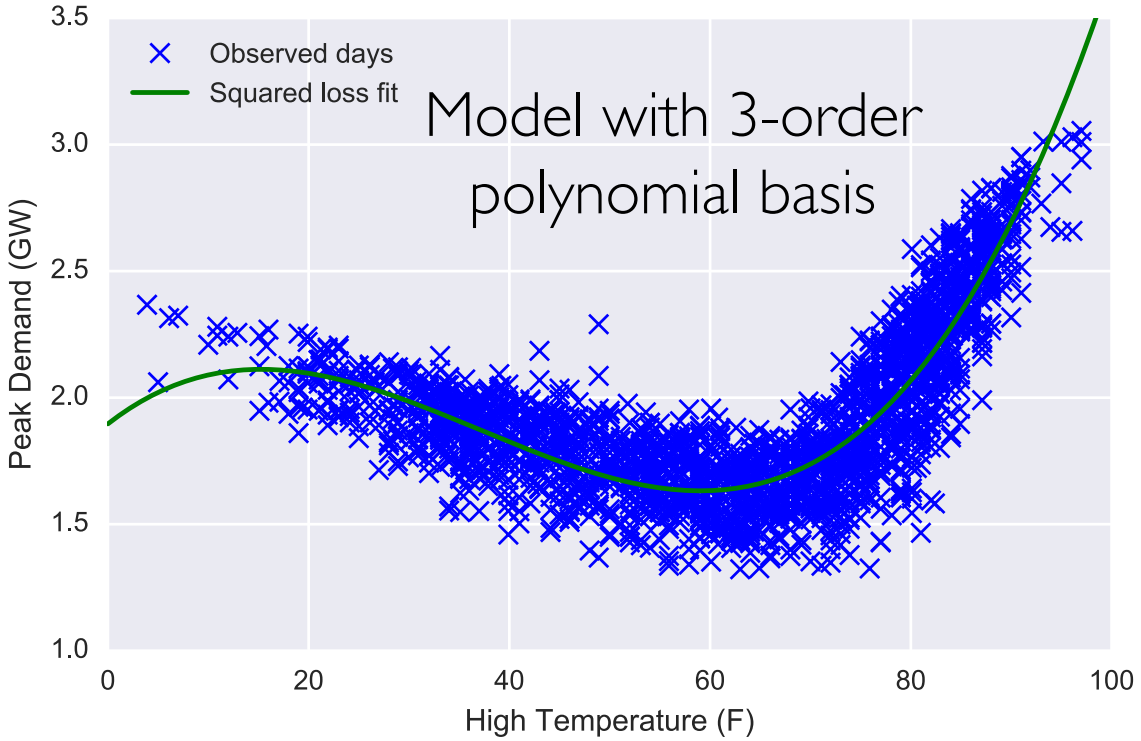
Nonlinearity

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

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



Linear Regression: Basis Function

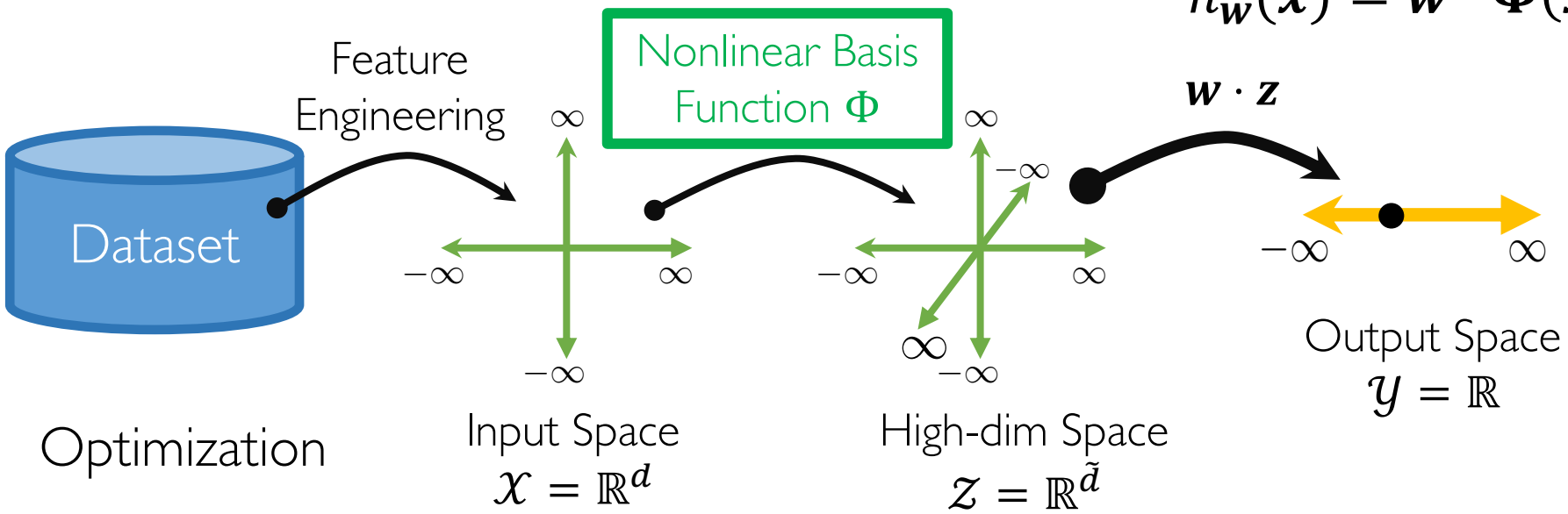


- With polynomial basis functions, we could fit nonlinear dataset.
  - Linear regression with polynomial basis functions.
- Higher-order polynomial has stronger ability to fit complex data.

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

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

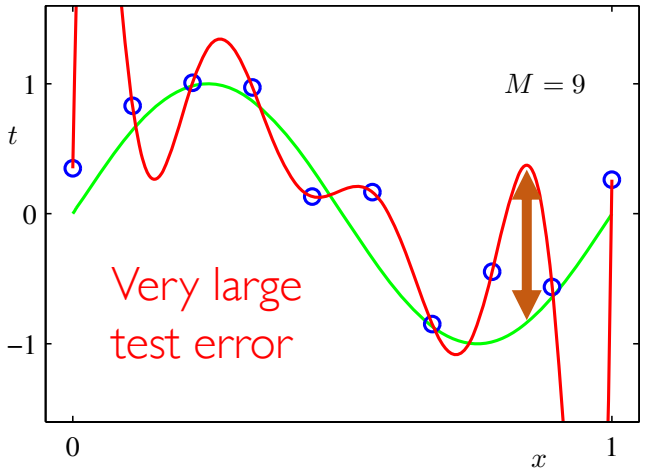
How to deal with overfitting?

Outline

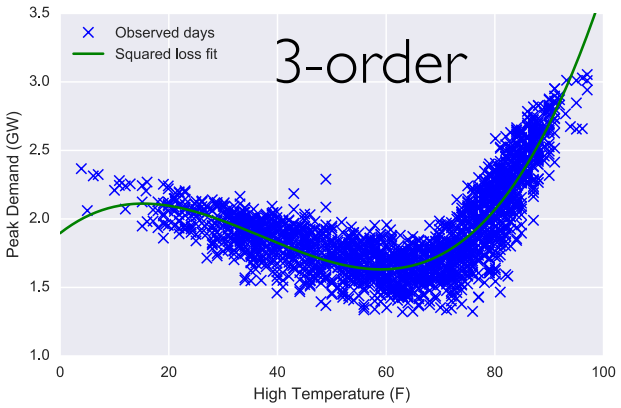
- Machine Learning
  - Framework
  - Evaluation Metrics
  - Model Selection
- K-Nearest Neighbors (KNN)
- **Linear Regression (LR)**
  - Optimization
  - Nonlinearization
  - **Regularization**

Overfitting

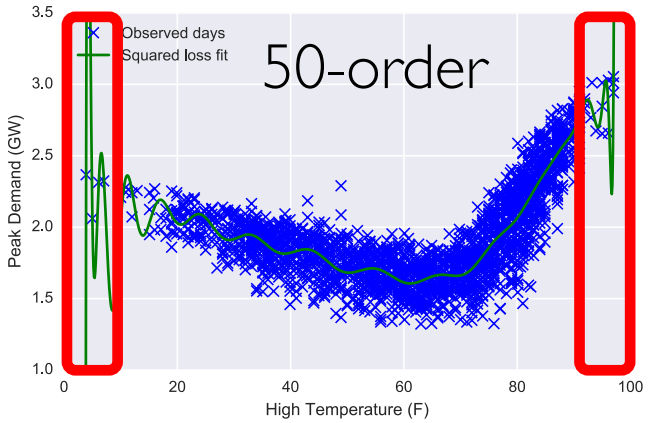
- Higher-order polynomial fits data better.
- But may fit noise.
- Also may cause very large coefficients.



Good fit



Over fit



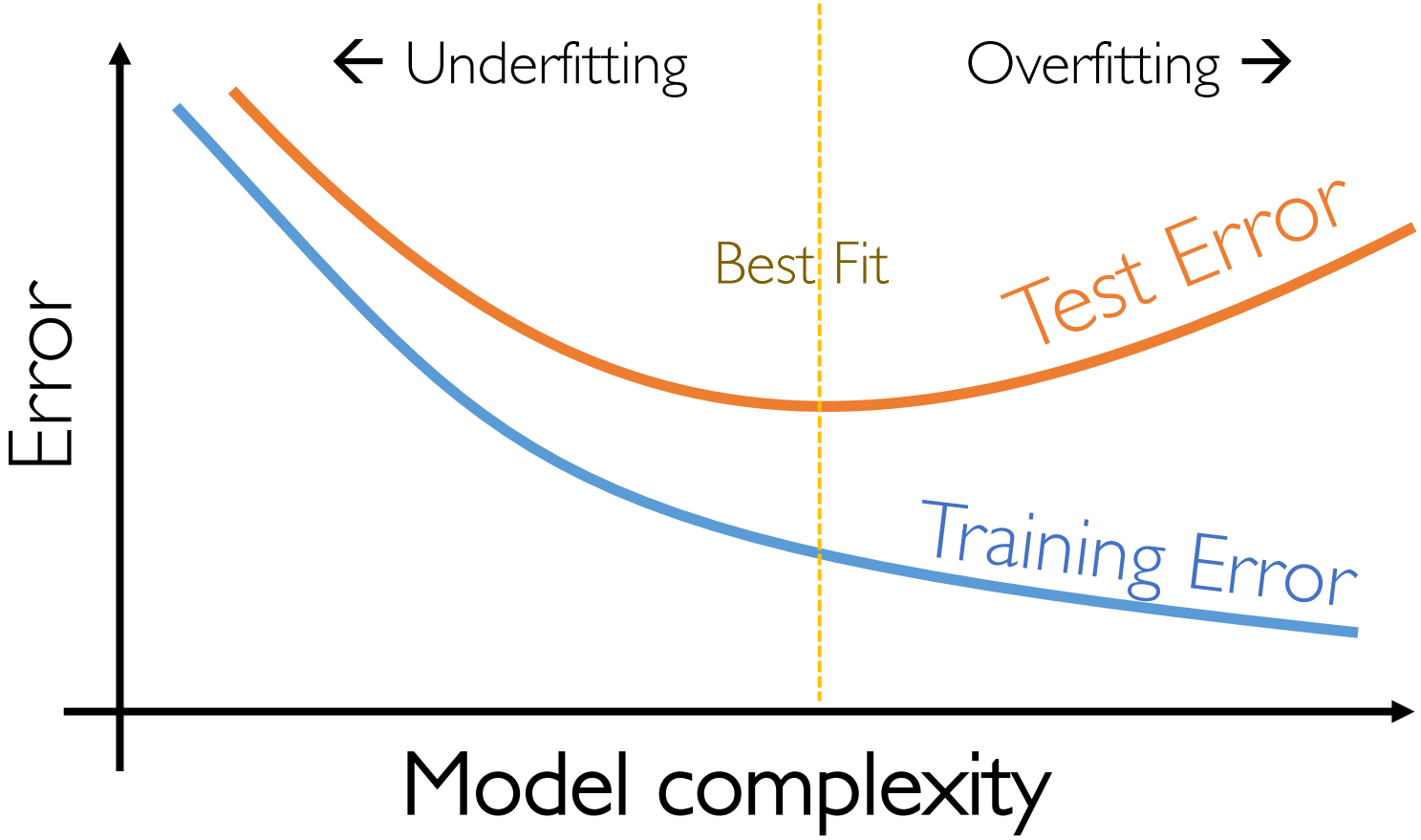
$w_0^\star$	0.35
$w_1^\star$	232.37
$w_2^\star$	-5321.83
$w_3^\star$	48568.31
$w_4^\star$	-231639.30
$w_5^\star$	640042.26
$w_6^\star$	-1061800.52
$w_7^\star$	1042400.18
$w_8^\star$	-557682.99
$w_9^\star$	125201.43

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 traded off with a hyperparameter $\lambda \geq 0$:

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

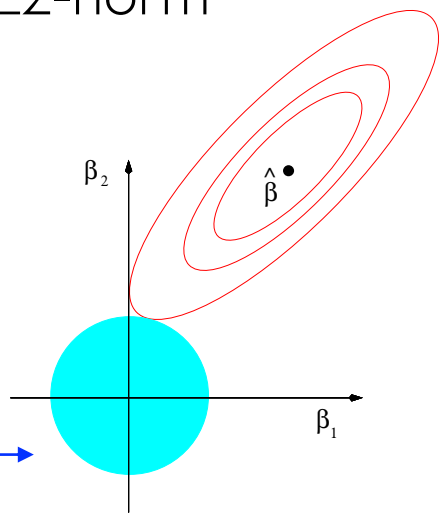
  - Where $\|\boldsymbol{w}\|_2^2 = w_1^2 + \cdots + w_d^2$ is the square of L2-norm
  - Which is equivalent to:

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

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 traded off with a hyperparameter $\lambda \geq 0$

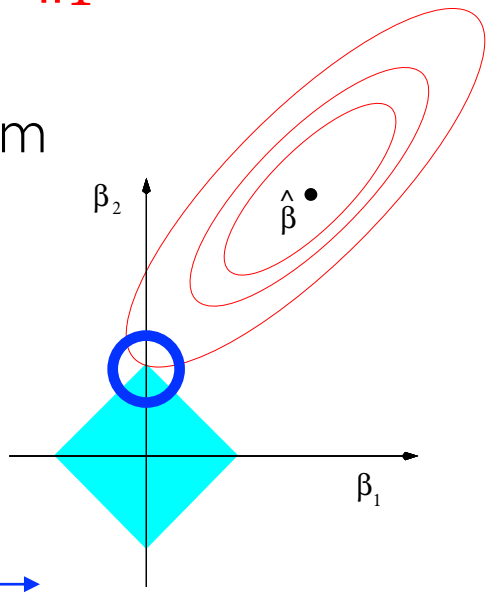
$$\hat{\boldsymbol{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \left(\mathbf{w}^{\mathrm{T}} \boldsymbol{x}_i - y_i\right)^2 + \lambda \|\boldsymbol{w}\|_1$$

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

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

with some $r > 0$.

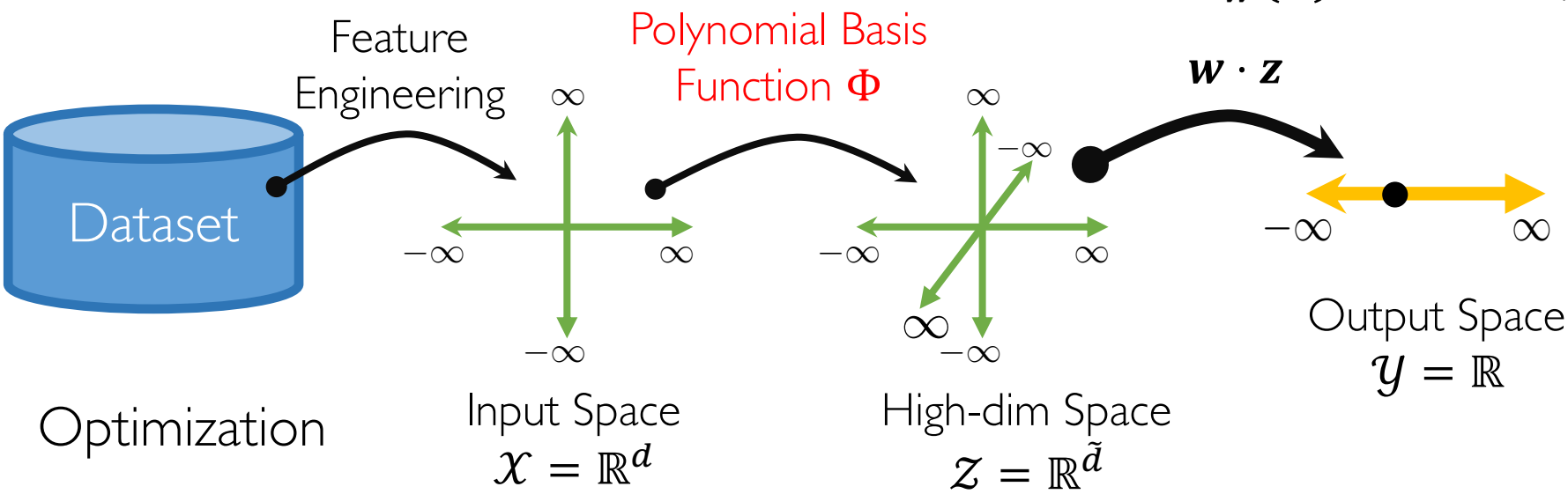
- $r \to 0 \Leftrightarrow \lambda \to +\infty$



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.

Thank You

Questions?

Mingsheng Long
mingsheng@tsinghua.edu.cn
http://ise.thss.tsinghua.edu.cn/~mlong
答疑：东主楼11区413室