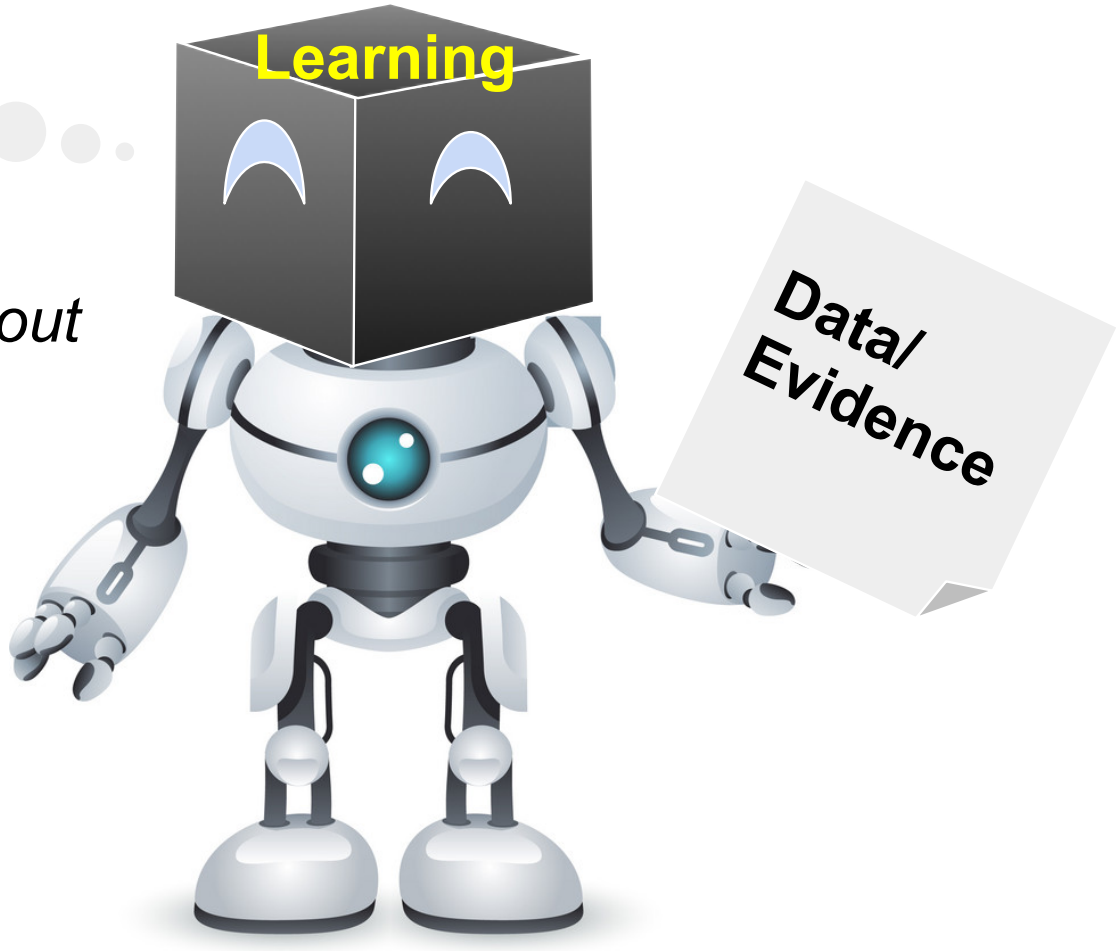
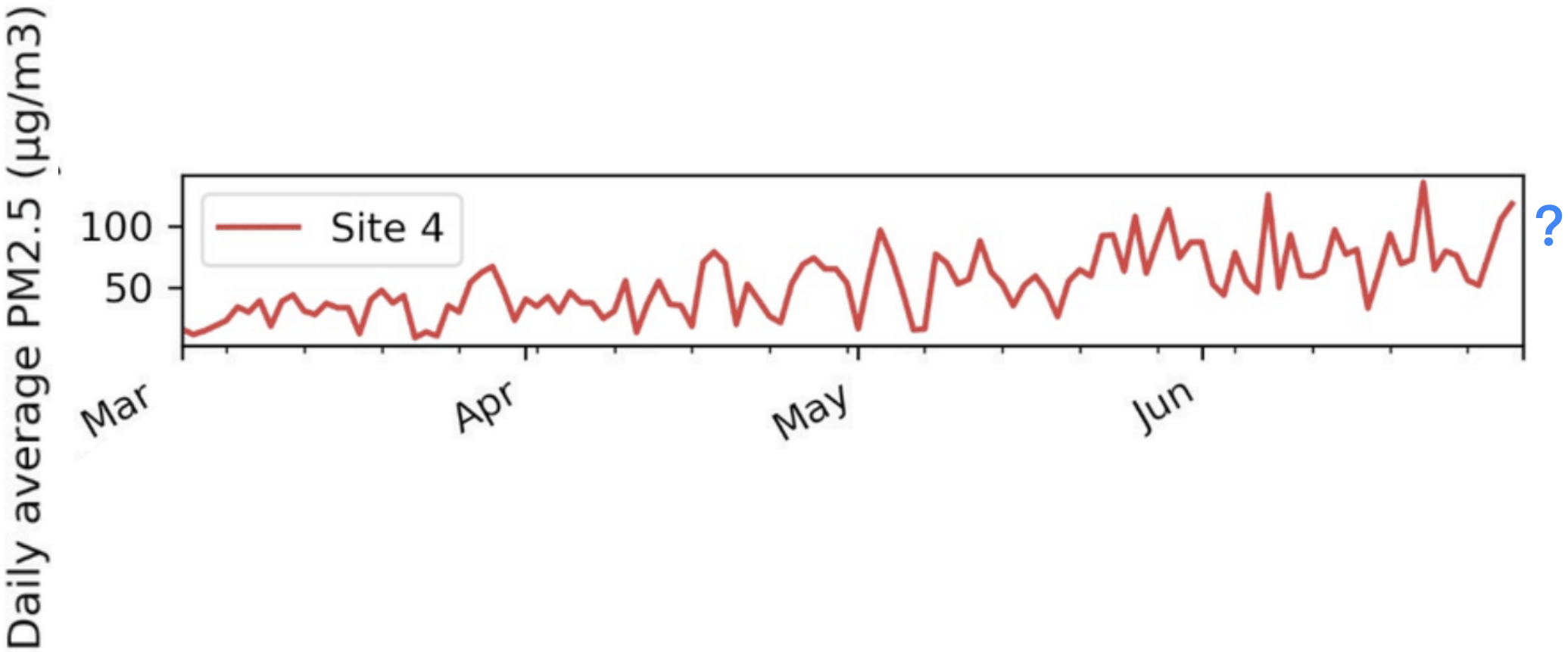
Machine Learning

Model

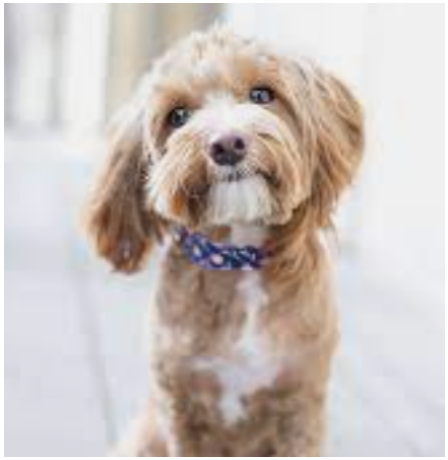
A function or a hypothesis about the world can solve problems

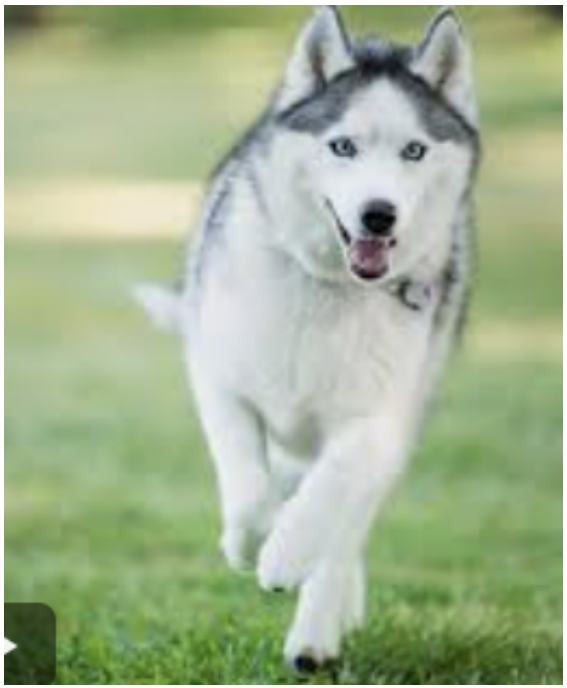


Source: https://cdn2.vectorstock.com/i/1000x1000/91/81/sale-cyber-monday-screen-monitor-head-robot-text-vector-18349181.jpg


Daily average PM2.5 (µg/m3)
100
50
Site 4
?
Mar
Apr
May
Jun





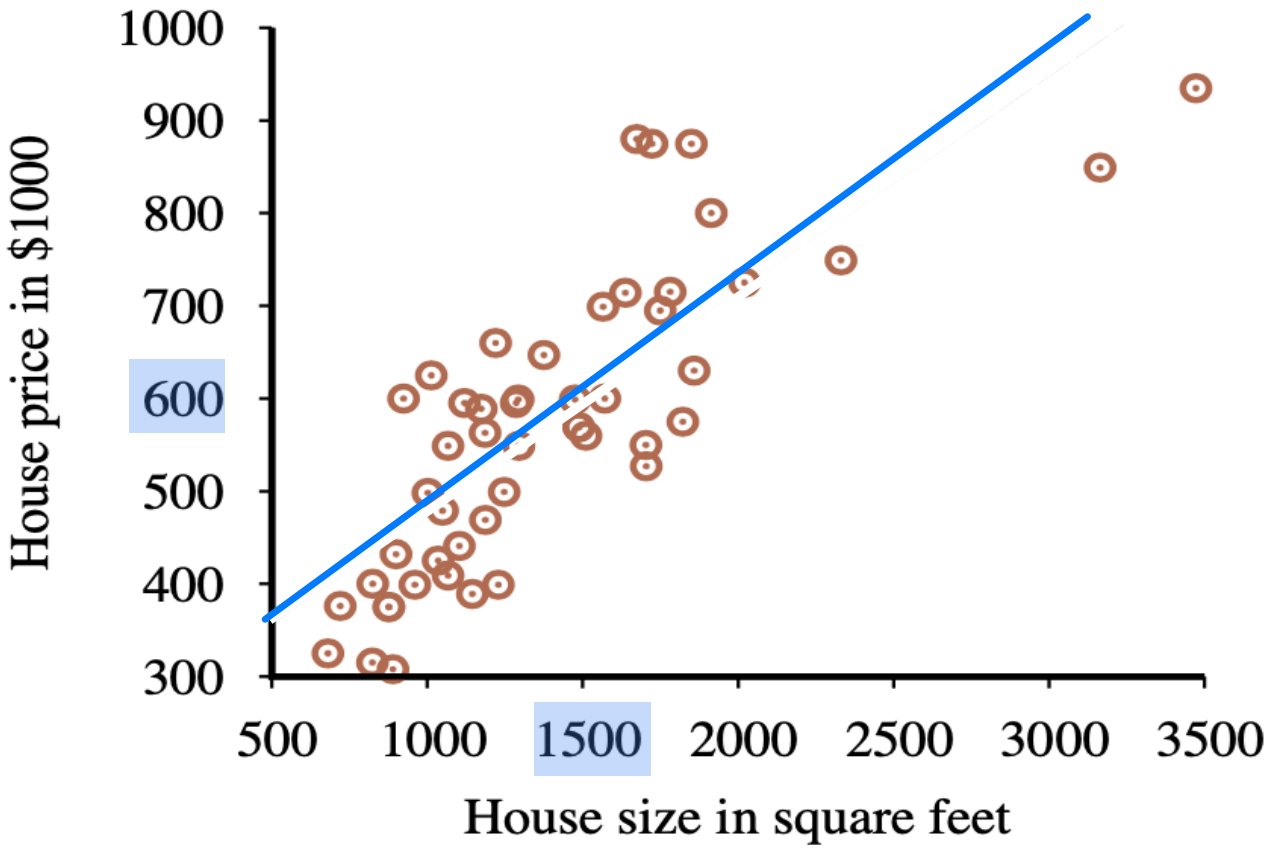

Types of Learning Problems

Supervised Learning

Supervised Learning

- Given a data set of input-output pairs,
  learn a function/hypothesis to map inputs to outputs
- Tasks
  - Regression
    - Learning predictor with **real-valued** outputs

Example: Regression



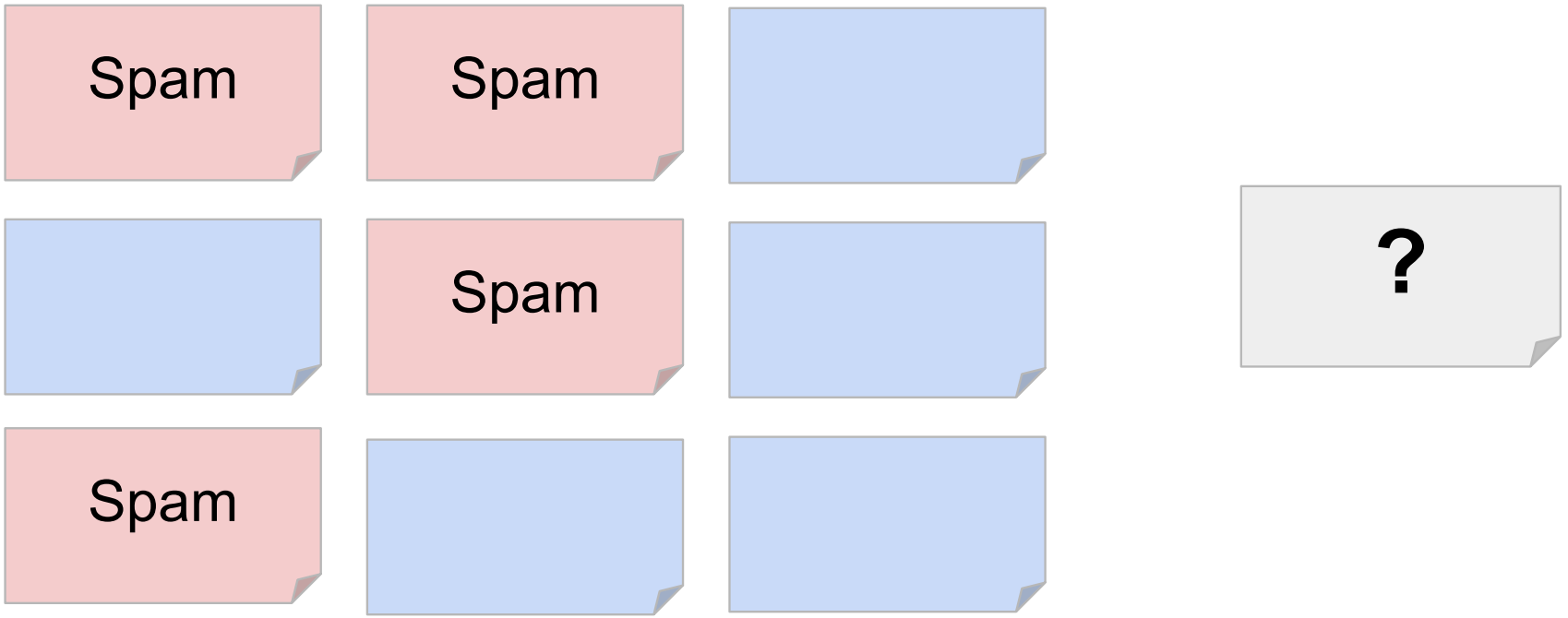
1sqft	0.0281坪
100sqft	2.81坪
1000sqft	28.1坪
1200sqft	33.72坪
1400sqft	39.34坪
1600sqft	44.96坪
1800sqft	50.58坪
2000sqft	56.2坪
2200sqft	61.82坪
2400sqft	67.44坪
2600sqft	73.06坪
2800sqft	78.68坪
3000sqft	84.3坪
3200sqft	89.92坪
3400sqft	95.54坪
3600sqft	101.16坪

😏 : 40坪, $?

Supervised Learning

- Given a data set of input-output pairs,
  learn a function/hypothesis to map inputs to outputs
- Tasks
  - Regression
    - Learning predictor with **real-valued** outputs
  - Classification
    - Learning predictor with **discrete** outputs (labels)
      - Binary classification
      - Multi-class classification

Example: Binary Classification

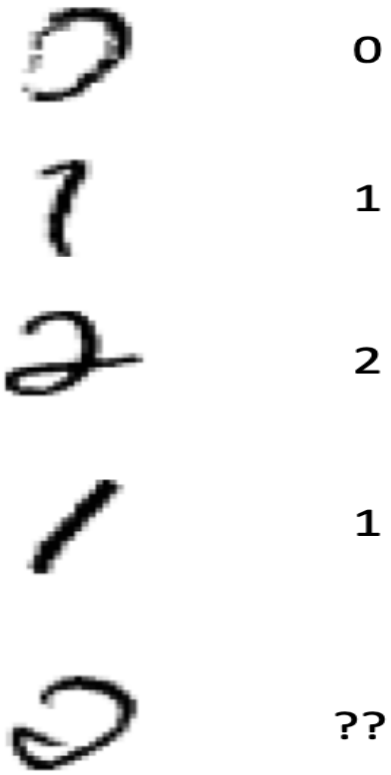


Example: Multi-Class Classification

credit@Google.News

Other Classification Applications

- Digit Recognition
  - Input: images
  - Output: a digit 0-9
- Medical diagnosis
  - Input: symptoms
  - Output: diseases
- Automatic essay grading
  - Input: documents
  - Output: grades
- Fraud detection
  - Input: account activity
  - Output: fraud / no fraud
- …



Types of Learning Problems

Supervised Learning

Unsupervised Learning

Unsupervised Learning

- Given a data set, learn/discover patterns in the input
- Tasks
  - Association analysis
    - Discover interesting relations between variables in large databases

Example: Association Rule Discovery

- Market basket analysis
  - Find items that are frequently bought together by customers

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

25歲妙齡女無明顯症狀　靠「商店會員卡」竟揪出卵巢癌

2023-02-10　健康醫療網 / 記者鄭宜芬外電報導

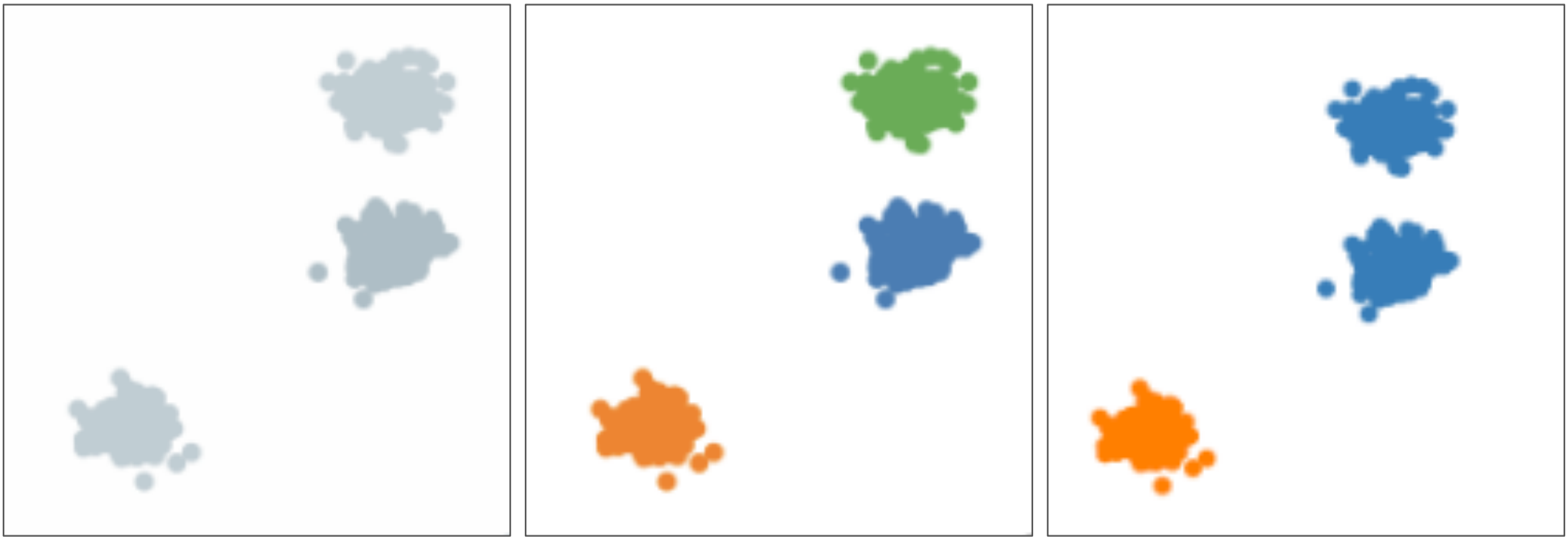
卵巢癌目前尚無可靠的篩檢，即使有腹脹症狀也不明顯，因此常與其他常見的非致命性病症混淆，等到確診時通常已為晚期。不過，根據《醫學網路研究期刊》（JMIR）發表的一項研究顯示，透過「商店會員卡」分析民眾購買的商品，可協助找出有癌症早期跡象的人，例如經常購買止痛成藥和消化不良藥品的民眾，罹患卵巢癌的風險也較高。一名25歲的英國女子就是這樣撿回一命。

https://www.healthnews.com.tw/article/56713

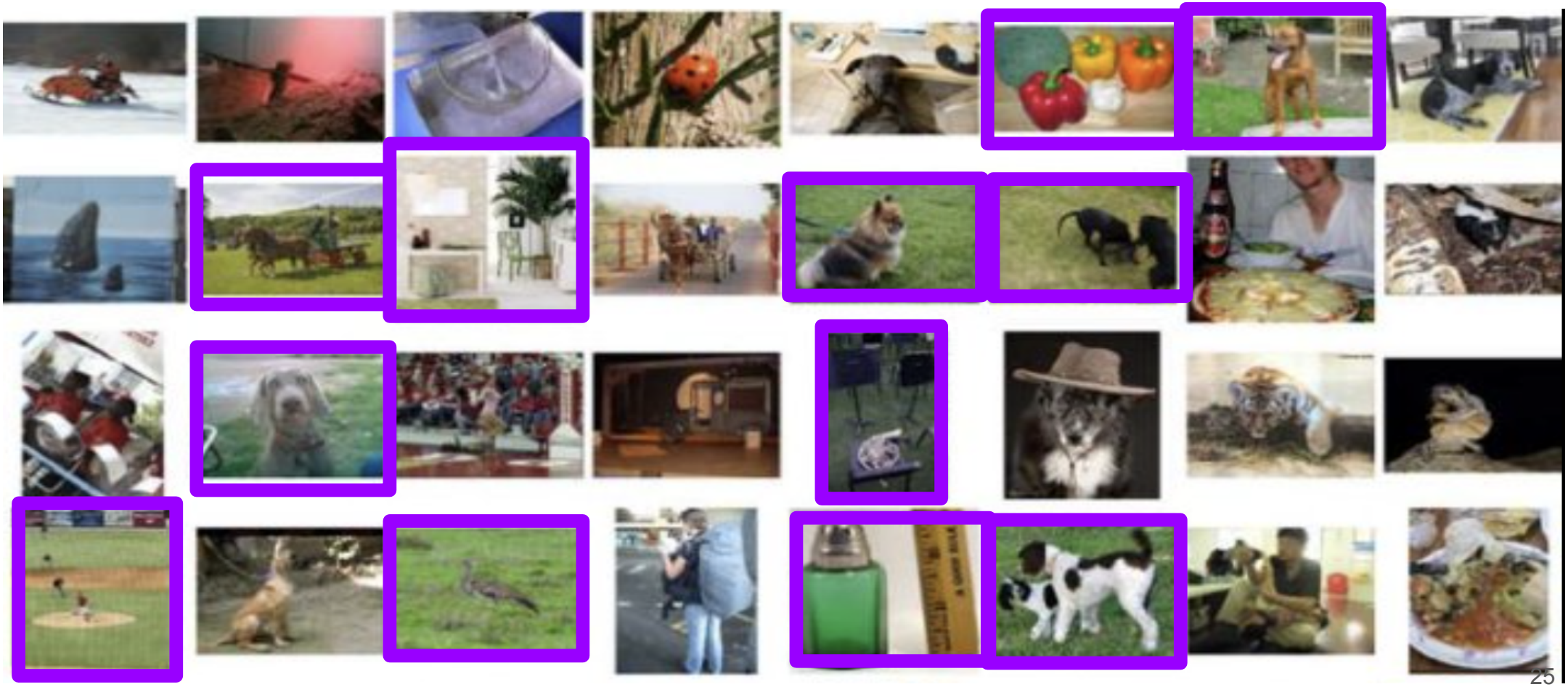
Unsupervised Learning

- Given a data set, learn/discover patterns in the input
- Tasks
  - Association analysis
    - Discover interesting relations between variables in large databases
  - Clustering
    - Detect potentially useful clusters of input examples

Example: Clustering

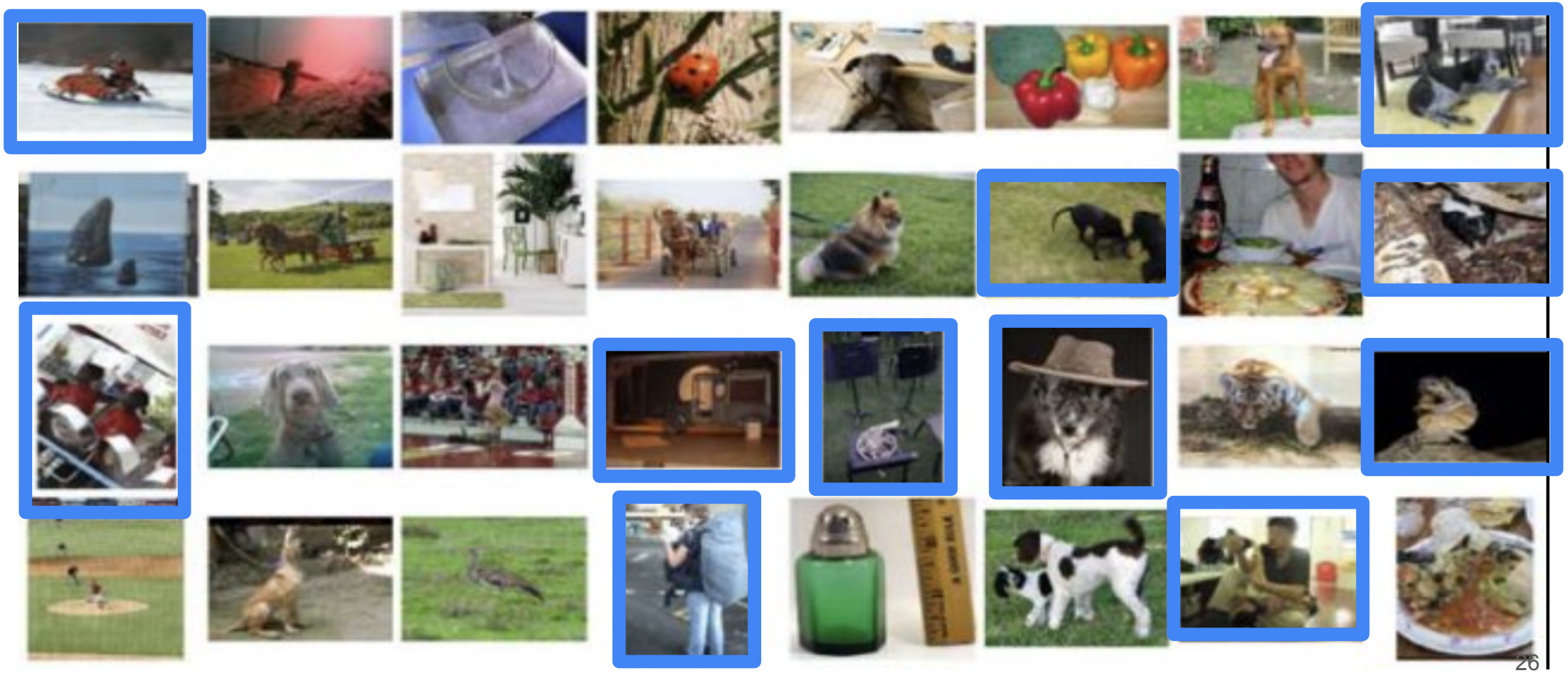


Example: Clustering



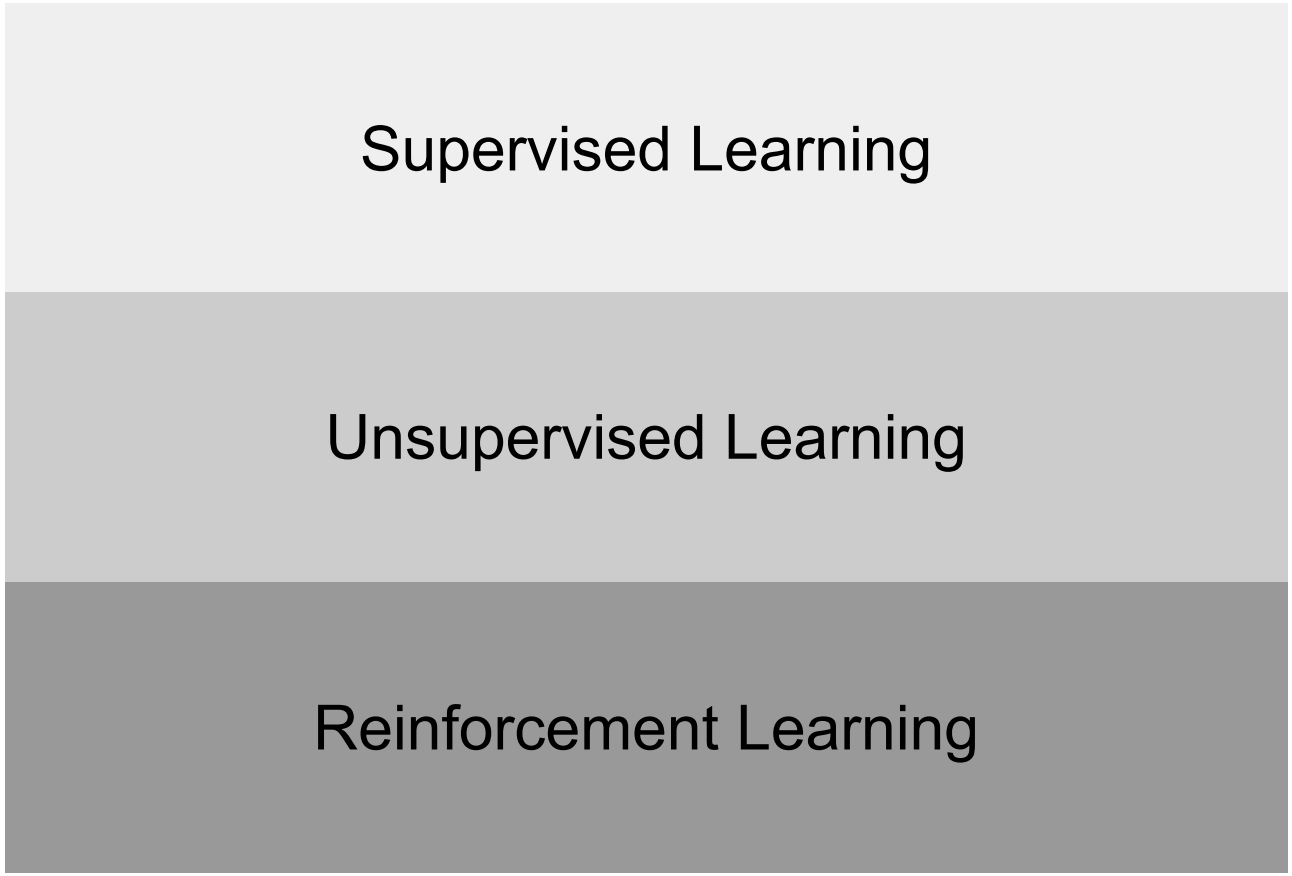
credit@imageNet

Example: Clustering



credit@imageNet

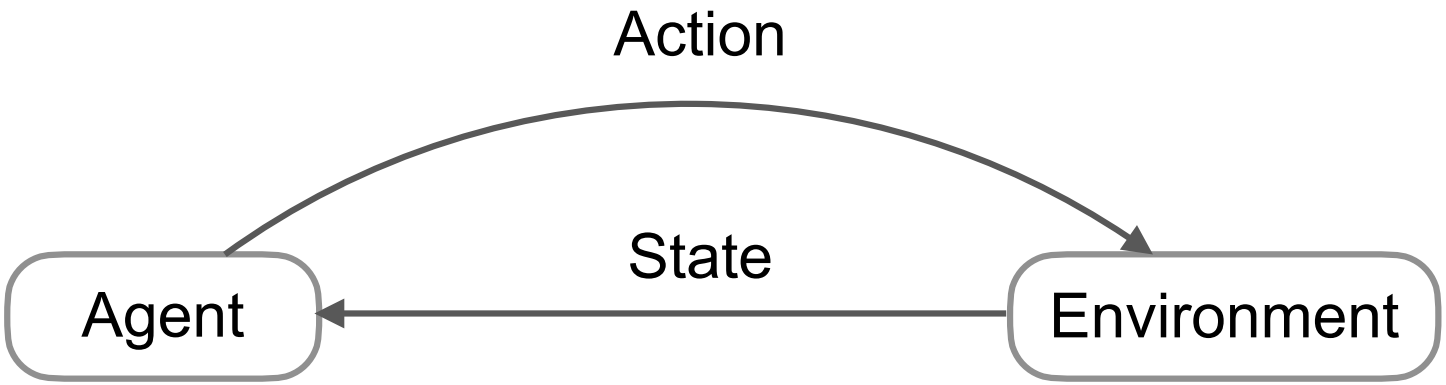
Types of Learning Problems

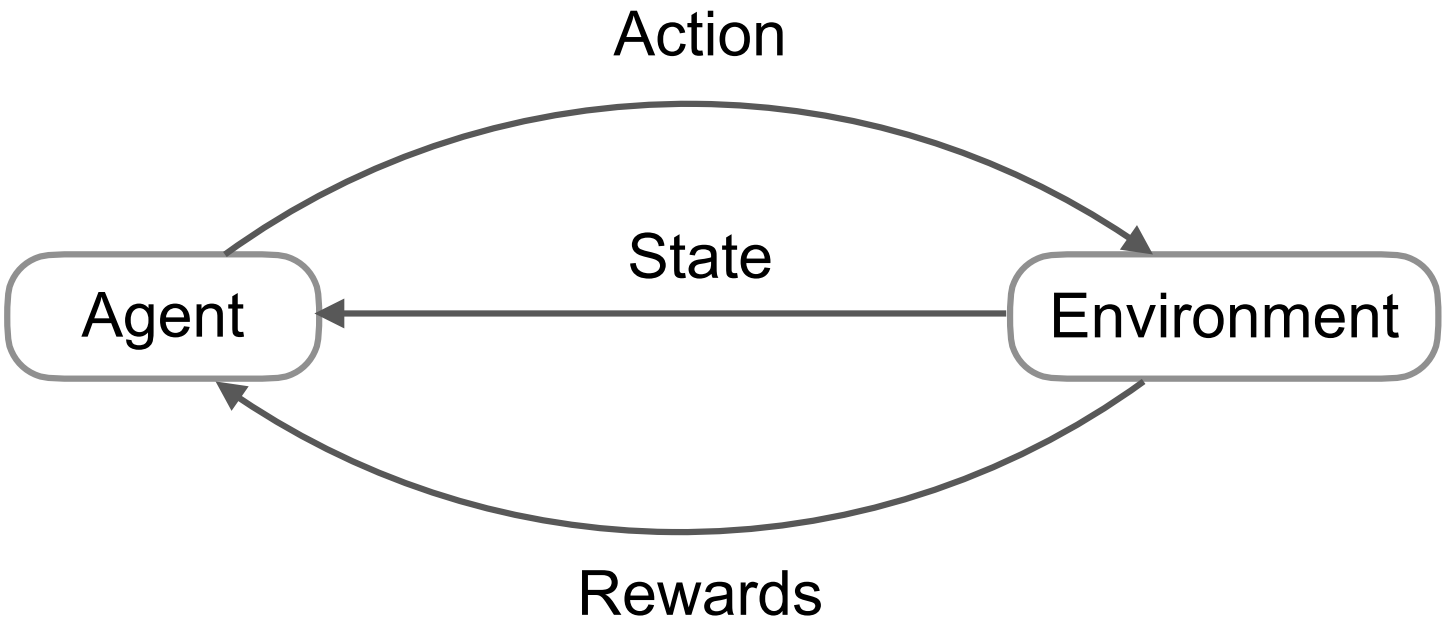


Reinforcement Learning

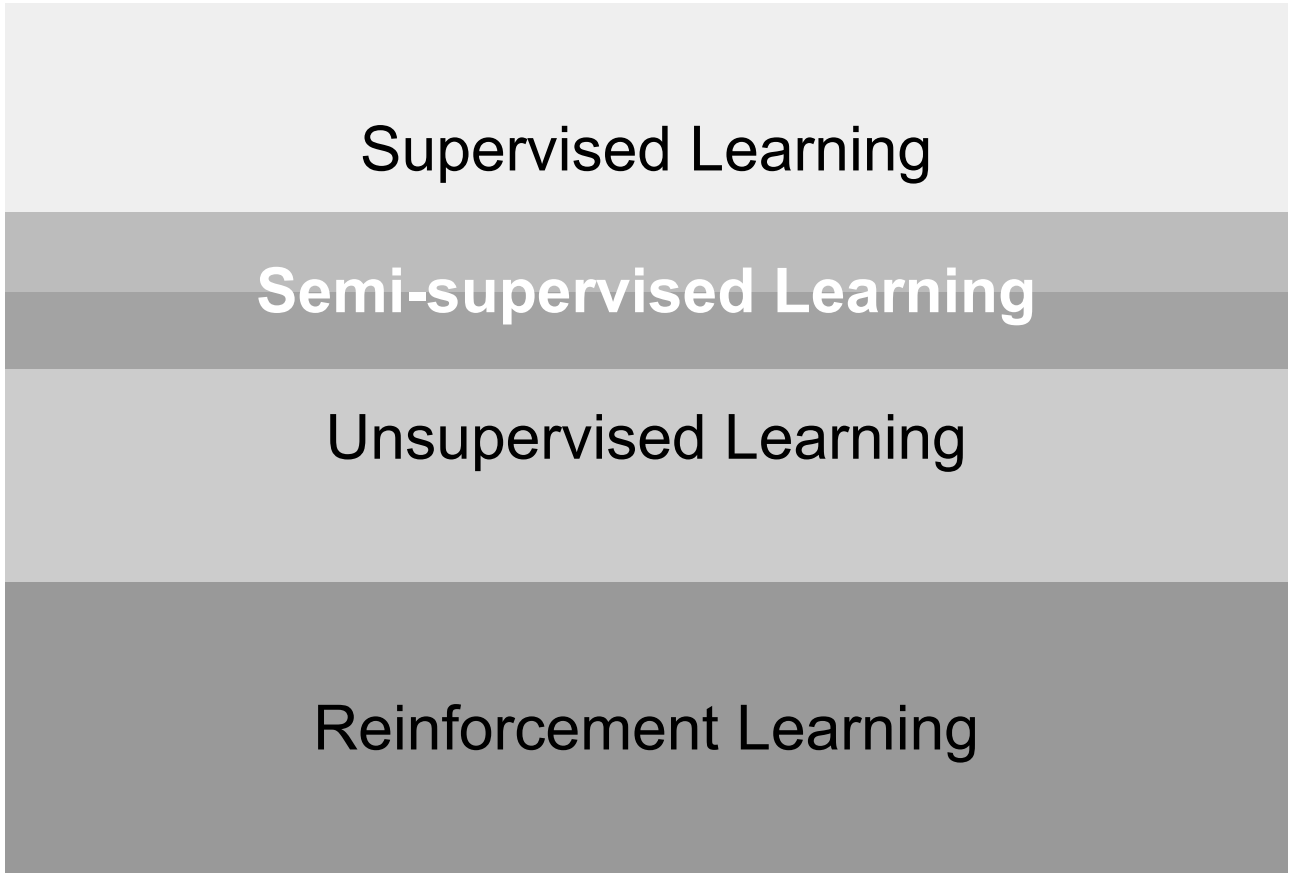
- Given a set of rewards or punishments, learn what actions to take in the future (no correct answers)
- Tasks
  - Go
  - Self-driving cars

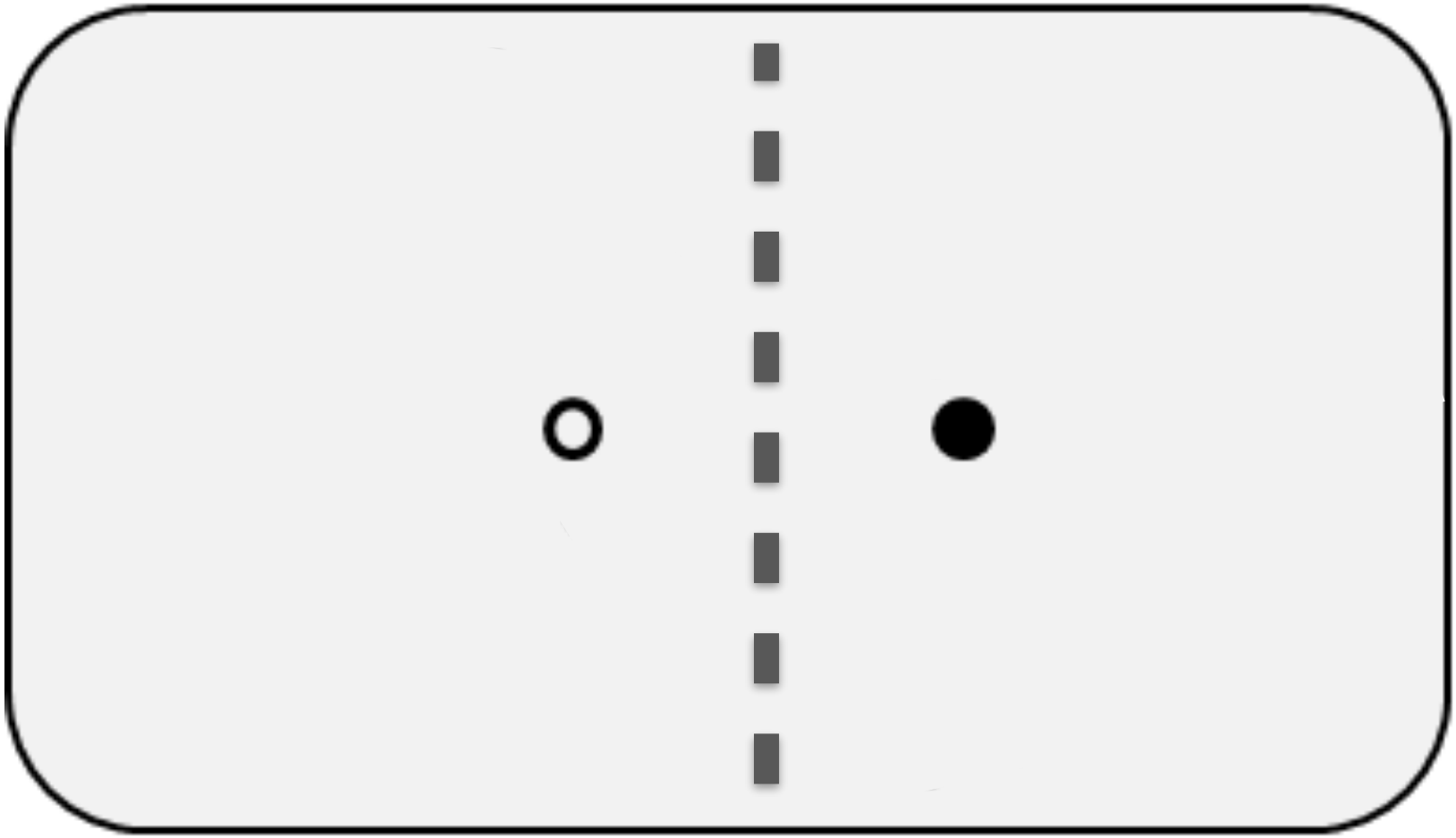
Source: https://media.istockphoto.com/id/1256202565/photo/miniature-golden-doodle-playing-in-the-park.jpg?s=612x612&w=0&k=20&c=9wQM_wslQVNF73Zu_aQ0IwZ_qdsyBCDdXKRUyDevJwA=

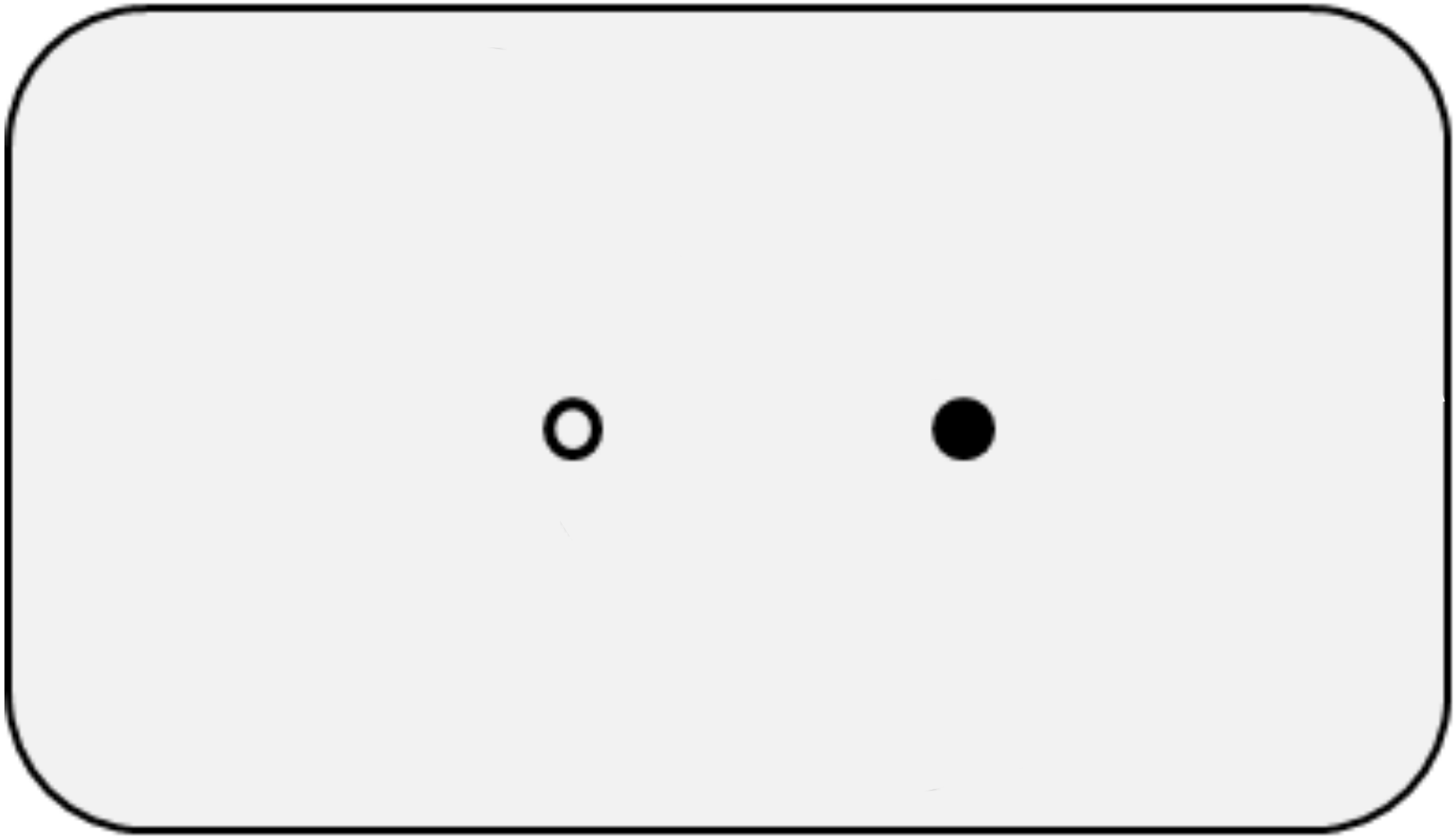

Action
State
Agent
Environment


Action
State
Agent
Environment
Rewards

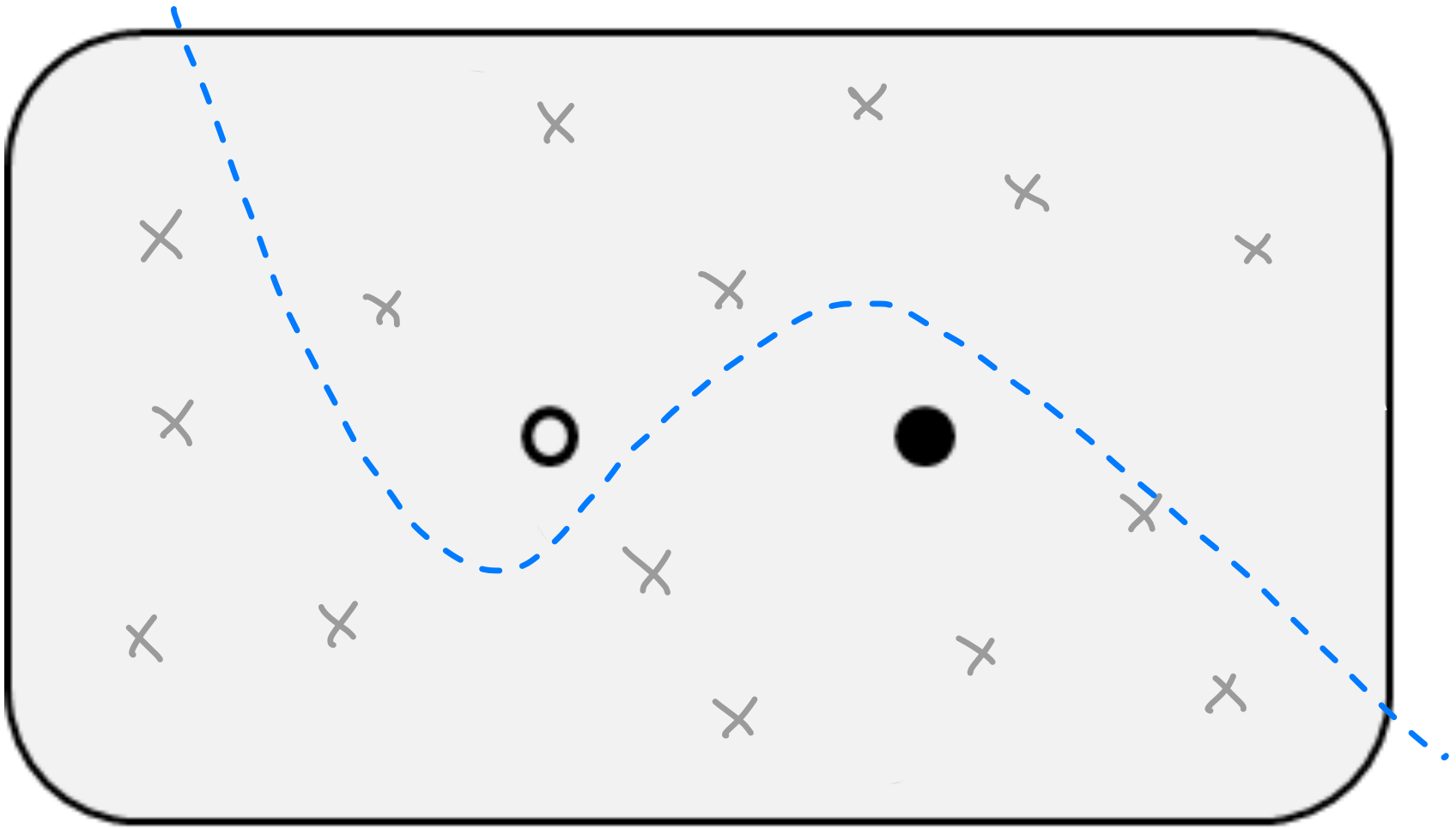
Types of Learning Problems







Semi-supervised Learning



Semi-supervised Learning

- Data set
  - A set of independently identically distributed examples X
  - $x_1,\ldots, x_l \in X$ with corresponding $y_1,..,y_l \in Y$: labeled
  - u unlabeled examples $x_{l+1},\ldots, x_{l+u} \in X$: unlabeled
- Types
  - Transductive learning (incorporates testing set during training)
    - Infer the correct labels for the given unlabeled data $x_{l+1},\ldots,x_{l+u}$ only
  - Inductive learning (popular)
    - Infer the correct mapping from X to Y

Supervised Learning

- Regression
- Classification

Supervised Learning

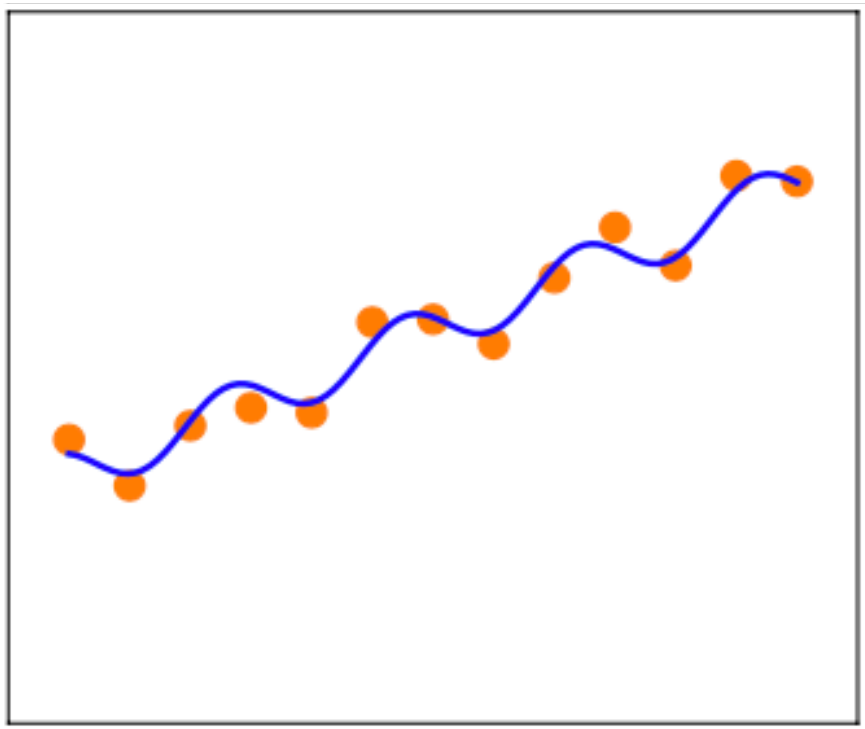
- Given a training set of input-output pairs

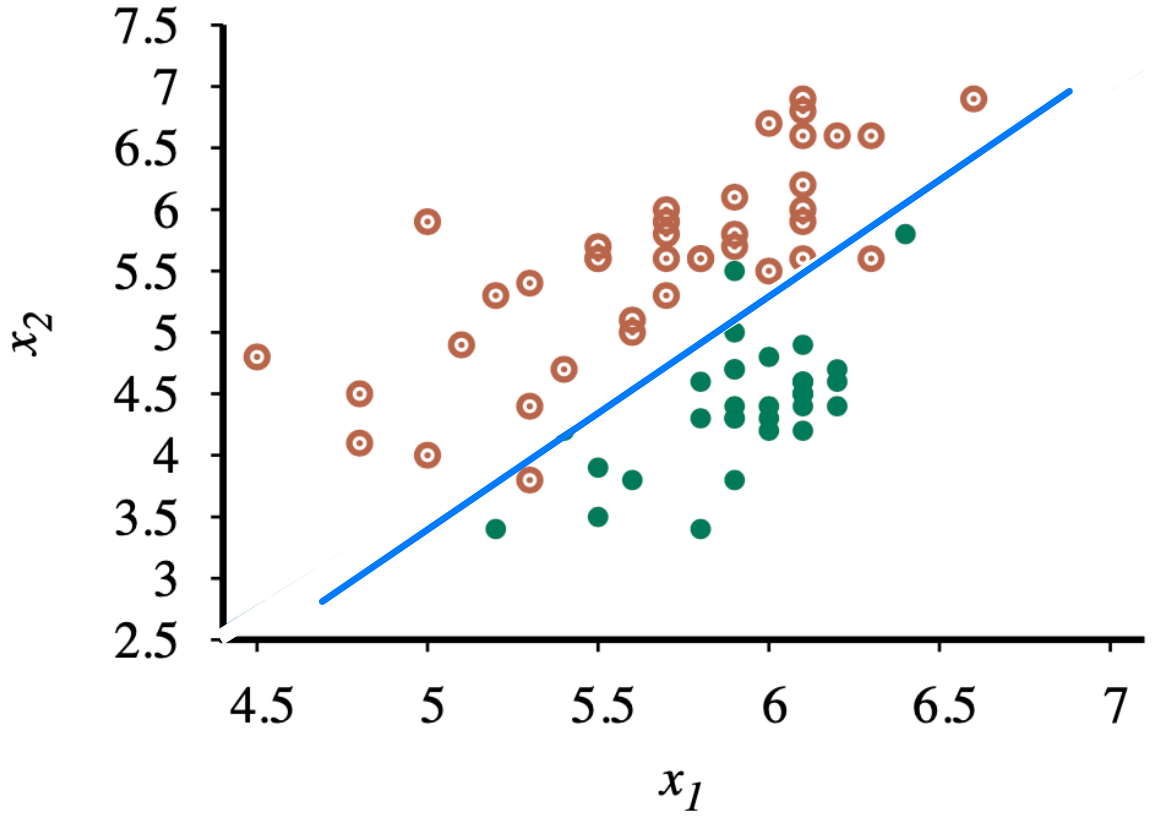
$$(x_1, y_1), (x_2, y_2) \ldots, (x_N, y_N),$$

where each y_i was generated by an unknown function $y = f(x)$,

discover a function h that approximates the true function f

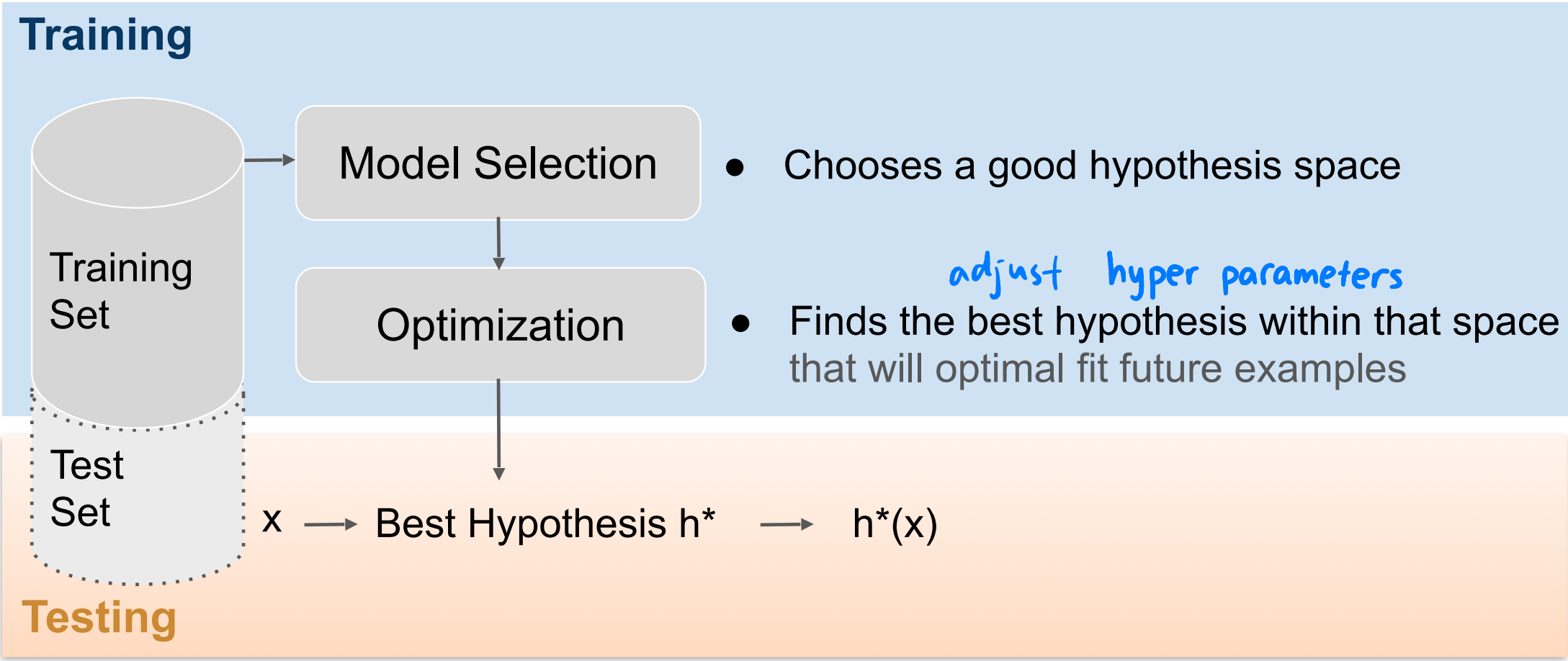
Example: Supervised Learning





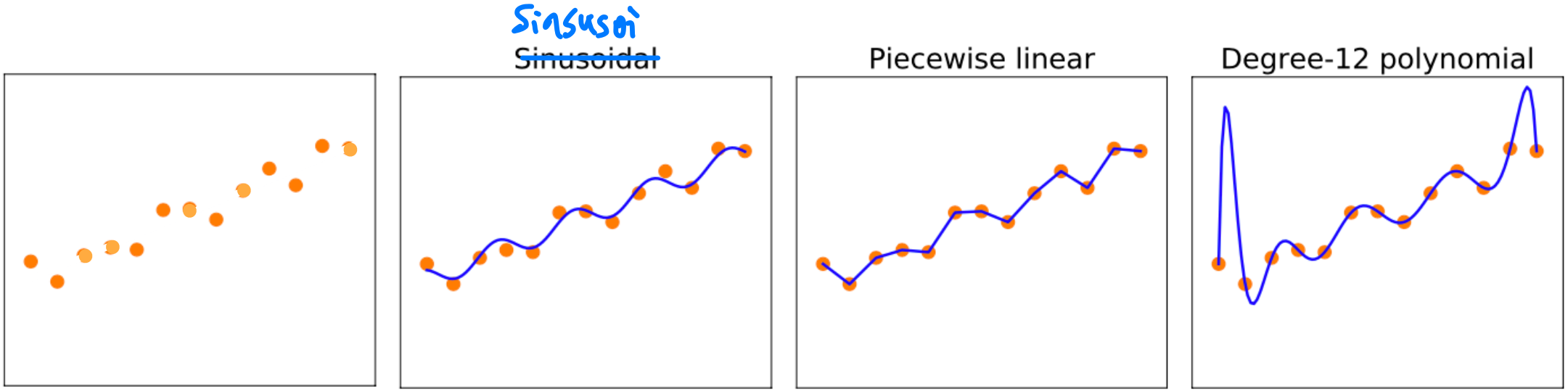
Learning Framework

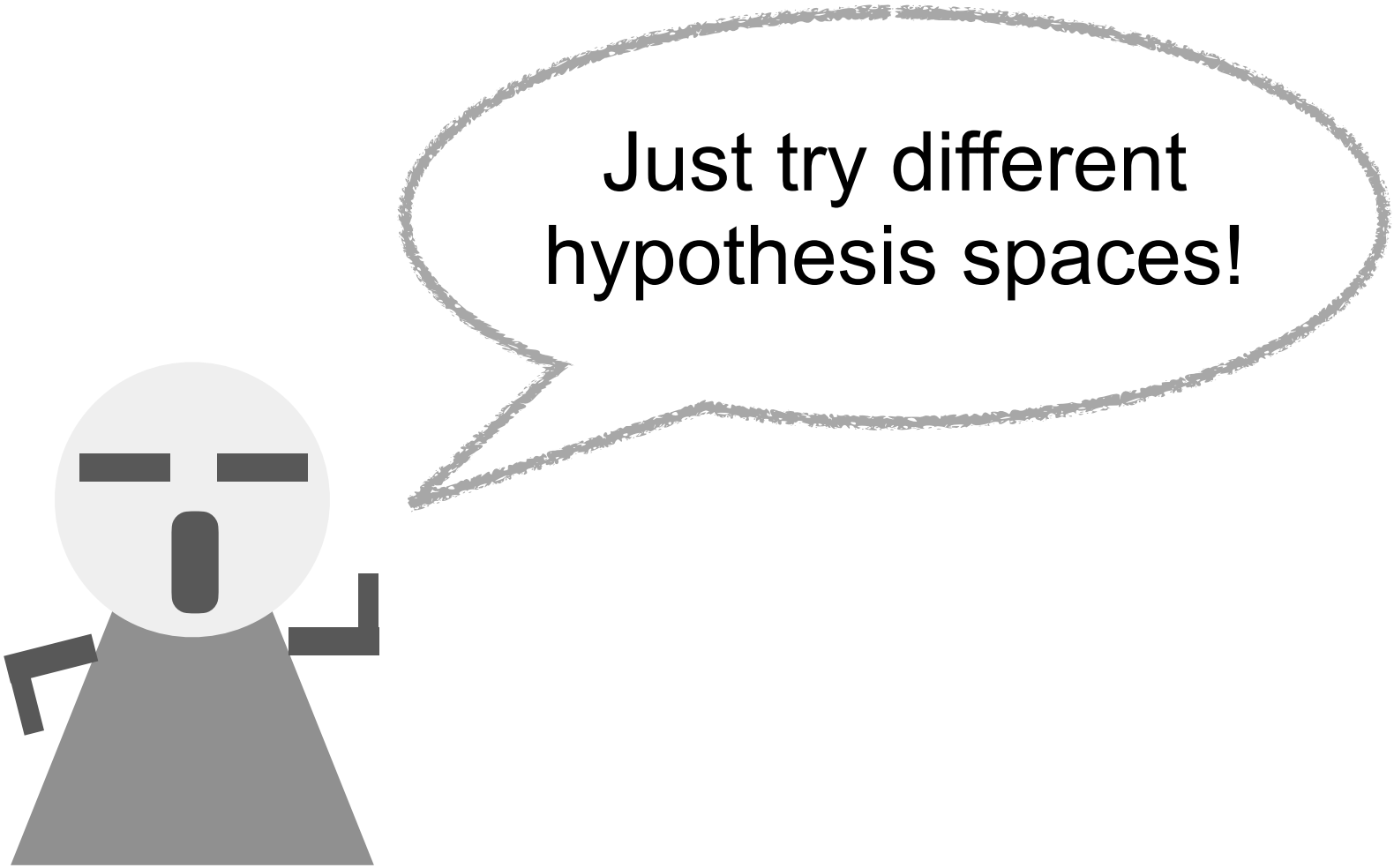
Learning Framework



Hypothesis Spaces: Exploratory Data Analysis (EDA)

- Statistical test
- Visualizations: histograms, scatter plots, box plots, etc.




Just try different
hypothesis spaces!

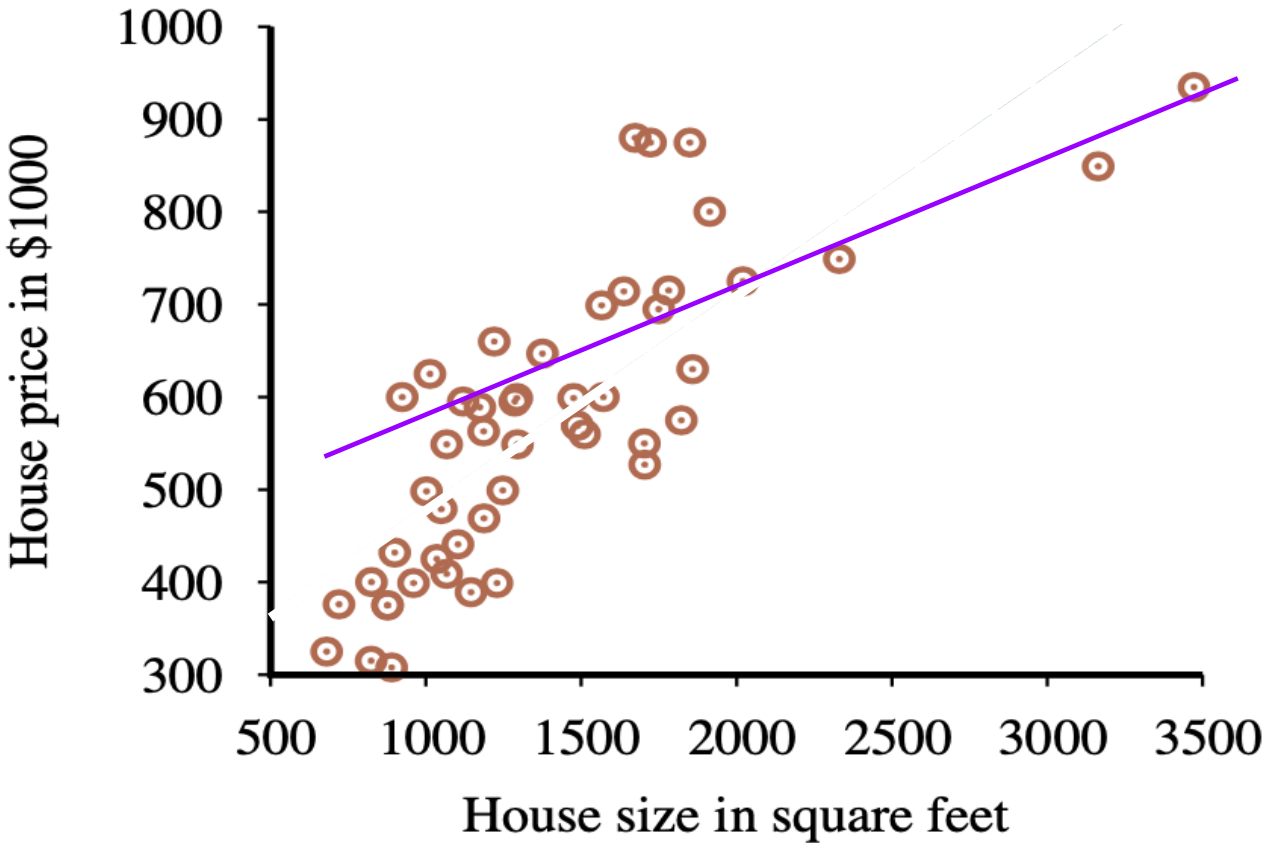
Supervised Learning

- **Regression**
- Classification

Regression: Linear Function

- **Univariate linear regression**
  - $\hat{y} = h_{\mathbf{w}}(x) = w_1 x + w_0$

    weights: $w_0, w_1 \in R$

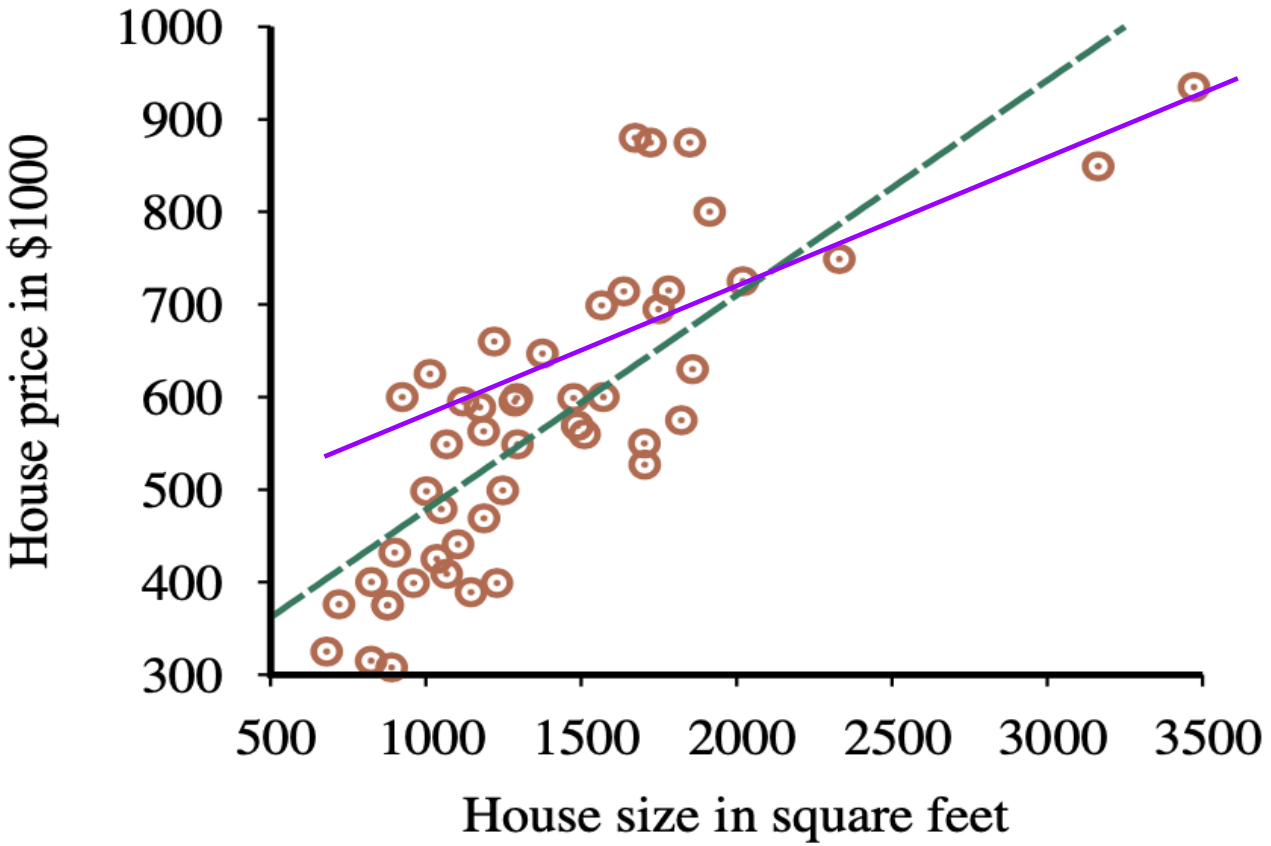


Regression: Linear Function

- **Univariate linear regression**
  - $\hat{y} = h_{\mathbf{w}}(x) = w_1 x + w_0$

    weights: $w_0, w_1 \in R$
- Find an optimal function such that

  $\Sigma_y Error(y, \hat{y})$ is minimized



Loss Function

- Expresses how poorly our hypothesis performs
- Examples

$$\text{Absolute-value loss:} \quad L_1(y, \hat{y}) = |y - \hat{y}|$$

$$\text{Squared-error loss:} \quad L_2(y, \hat{y}) = (y - \hat{y})^2$$

$$0/1 \text{ loss:} \quad L_{0/1}(y, \hat{y}) = 0 \text{ if } y = \hat{y}, \text{ else } 1$$

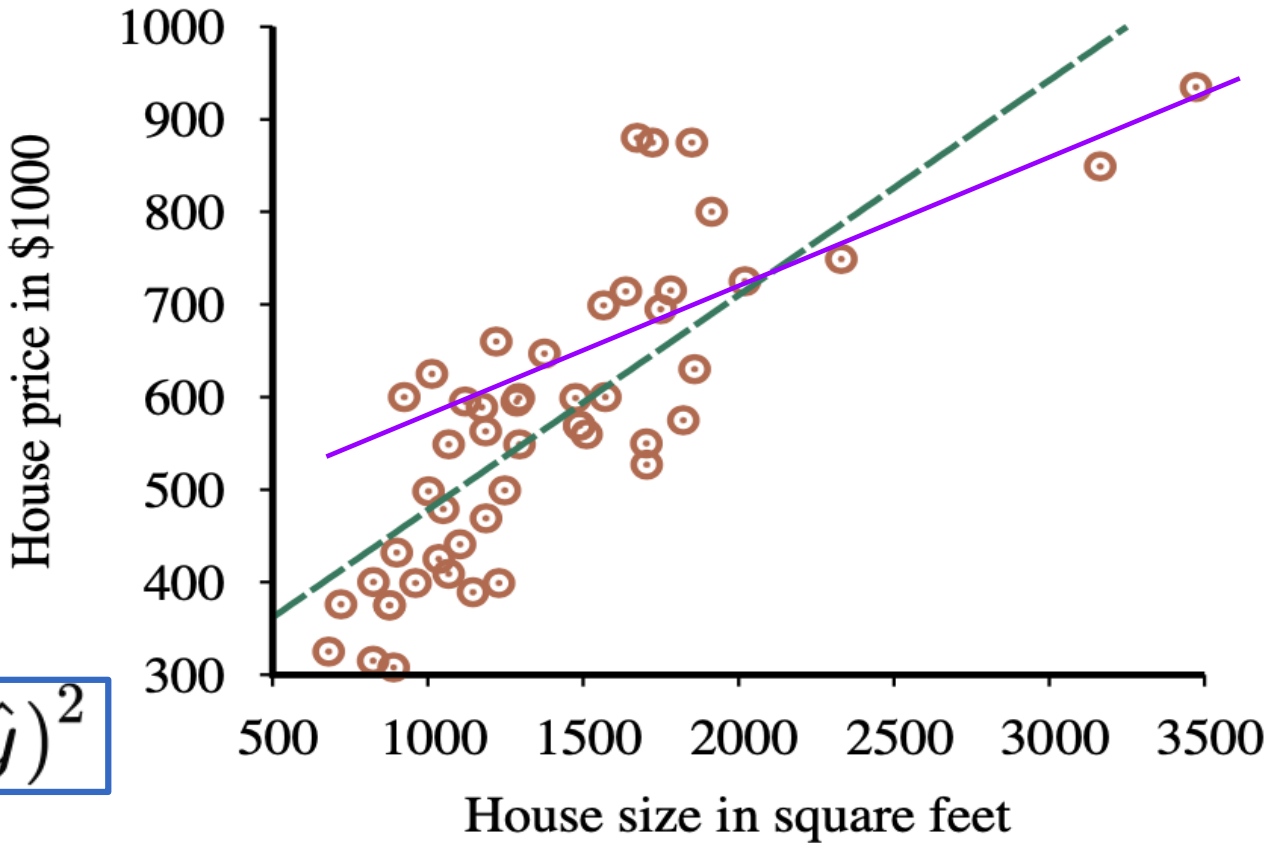
Regression: Linear Function

- Univariate linear regression
  - $\hat{y} = h_{\mathbf{w}}(x) = w_1 x + w_0$

    weights: $w_0, w_1 \in R$
- Optimization
  - Find an optimal function such that

    $\Sigma_y Error(y, \hat{y})$ is minimize

$$L_2(y, \hat{y}) = (y - \hat{y})^2$$



Optimization

- Minimize the loss function

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2\left(y_j, h_{\mathbf{w}}\left(x_j\right)\right) = \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right)^2 = \sum_{j=1}^{N} \left(y_j - \left(w_1 x_j + w_0\right)\right)^2$$

to find the best function by

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}}\, Loss(h_{\mathbf{w}})$$

Goal: $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \, Loss(h_{\mathbf{w}})$

Minimize $\sum_{j=1}^{N} \left(y_j - \left(w_1 x_j + w_0\right)\right)^2$

Let its partial derivatives with respect to weights be zero

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$

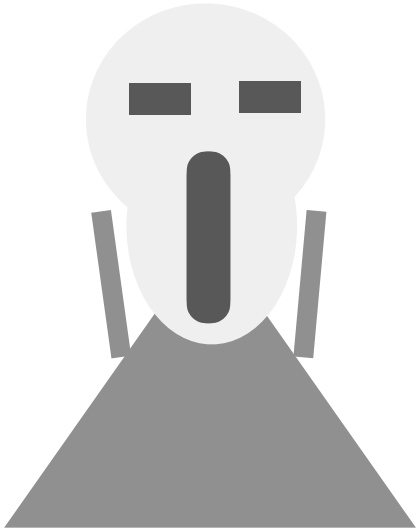
Closed form:

$$w_1 = \frac{N\left(\sum x_j y_j\right) - \left(\sum x_j\right)\left(\sum y_j\right)}{N\left(\sum x_j^2\right) - \left(\sum x_j\right)^2}; \qquad w_0 = \left(\sum y_j - w_1\left(\sum x_j\right)\right)/N.$$

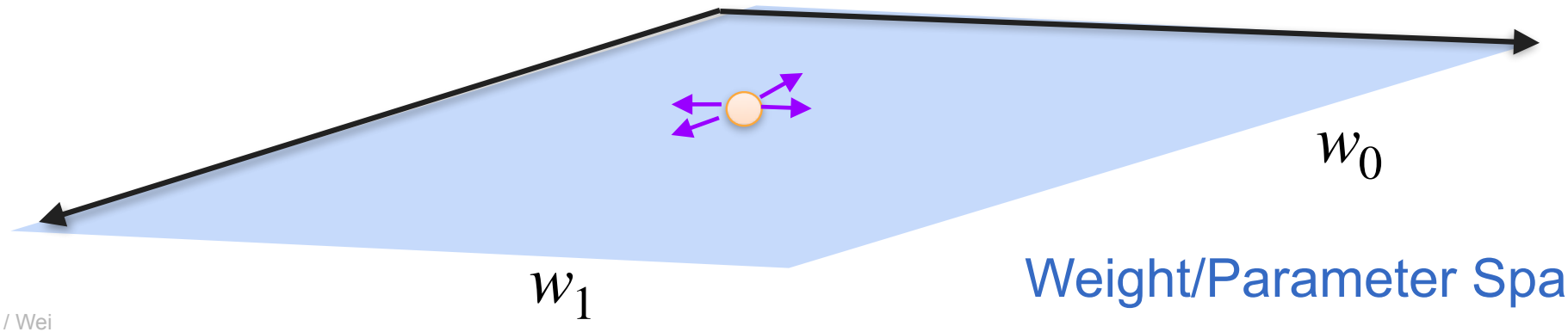
Issue:

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2\left(y_j, h_{\mathbf{w}}\left(x_j\right)\right) = \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right)^2$$

Let its partial derivatives with respect to weights be zero

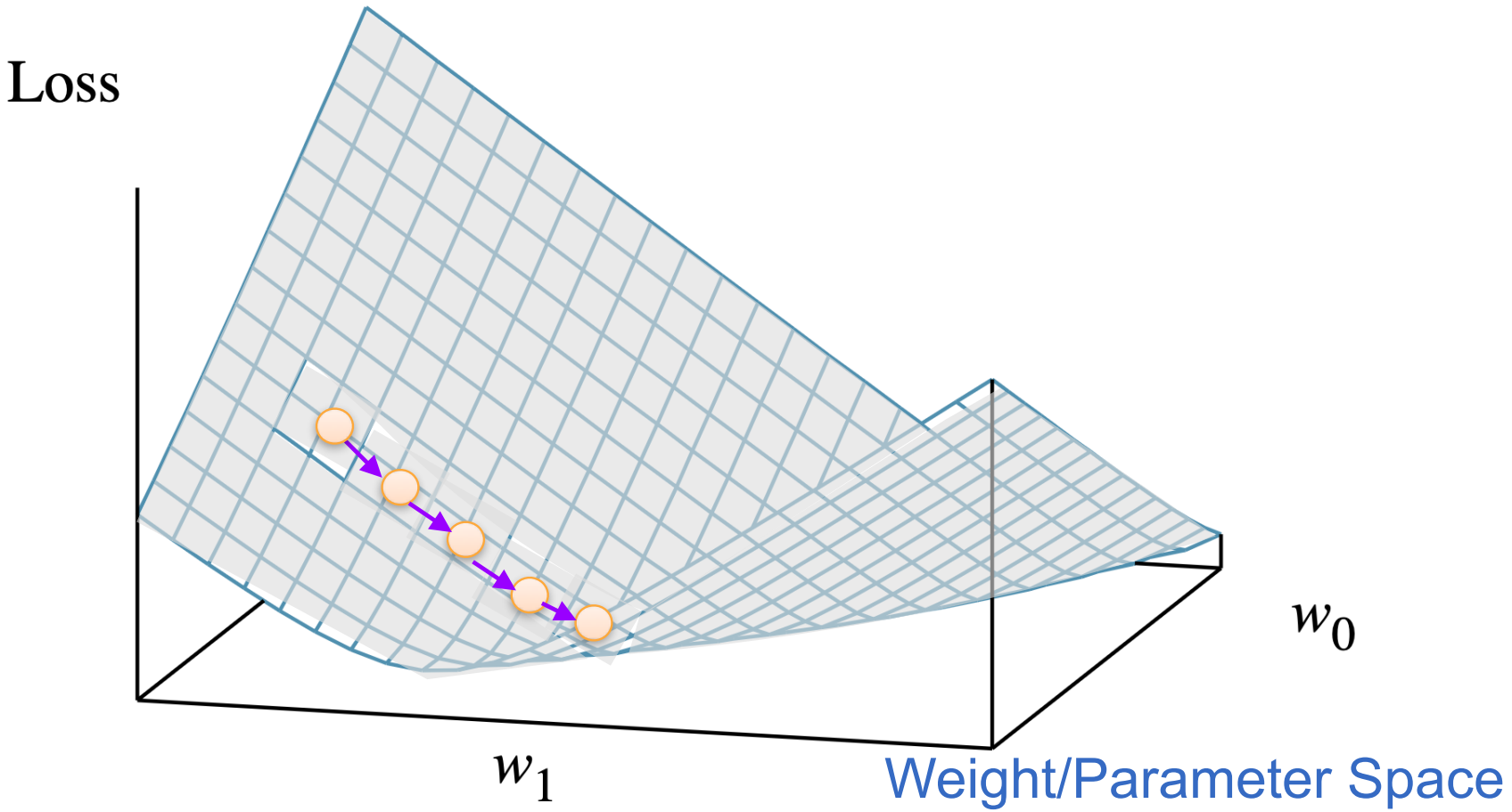


Maybe…
NO closed-form expression!



Weight/Parameter Space

Gradient Descent



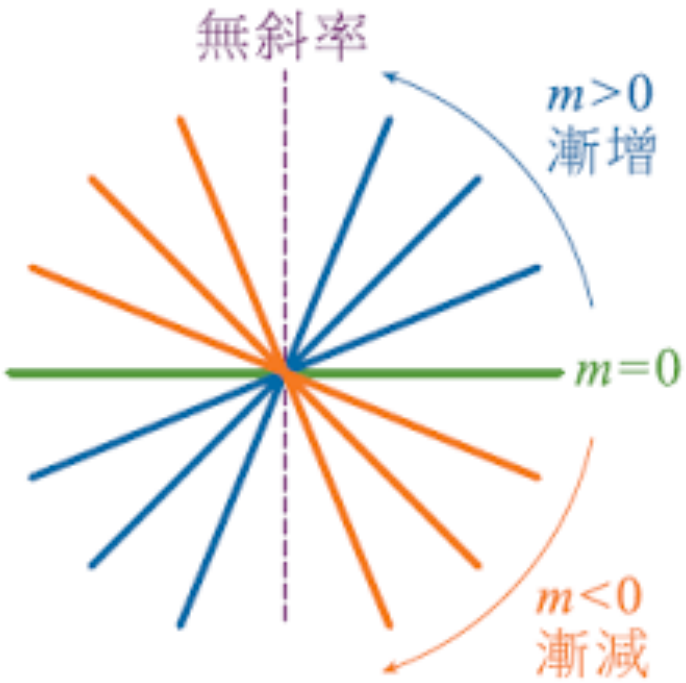
Gradient Descent (Cont.)

$\mathbf{w} \leftarrow$ any point in the parameter space

for each w_i **in** $\mathbf{w}$ **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ5PNf_bO5ToNfsqzUi-S967qZ1HgmjgmMCQqKsP0GuYw&s



Gradient Descent (Cont.)

$$\mathbf{w} \leftarrow \text{any point in the parameter space}$$

$$\textbf{for each } w_i \textbf{ in } \mathbf{w} \textbf{ do}$$

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss\left(\mathbf{w}\right)$$

Note. α: learning rate (step size)

- Fixed constant
- Decay over time as the learning process proceeds

Gradient Descent (Cont.)

$\mathbf{w} \leftarrow$ any point in the parameter space

while not converged **do**

  for each w_i **in** $\mathbf{w}$ **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss\,(\mathbf{w})$$

Note. α: learning rate (step size)

Gradient Descent (Cont.)

chain rule: $\partial g(f(x))/\partial x = g'(f(x))\,\partial f(x)/\partial x$

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$

$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)).$$

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2\,(y - h_{\mathbf{w}}(x)) \qquad \frac{\partial}{\partial w_1} Loss\,(\mathbf{w}) = -2\,(y - h_{\mathbf{w}}(x)) \times x.$$

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss\,(\mathbf{w})$$

$$w_0 \leftarrow w_0 + \alpha\ (y - h_{\mathbf{w}}(x)) \qquad w_1 \leftarrow w_1 + \alpha\ (y - h_{\mathbf{w}}(x)\ ) \times\ x.$$

Batch Gradient Descent (N at a time)

- For N training examples, we wan to minimize the sum of the individual losses for each example
- The derivate of a sum is the sum of the derivates

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))$$

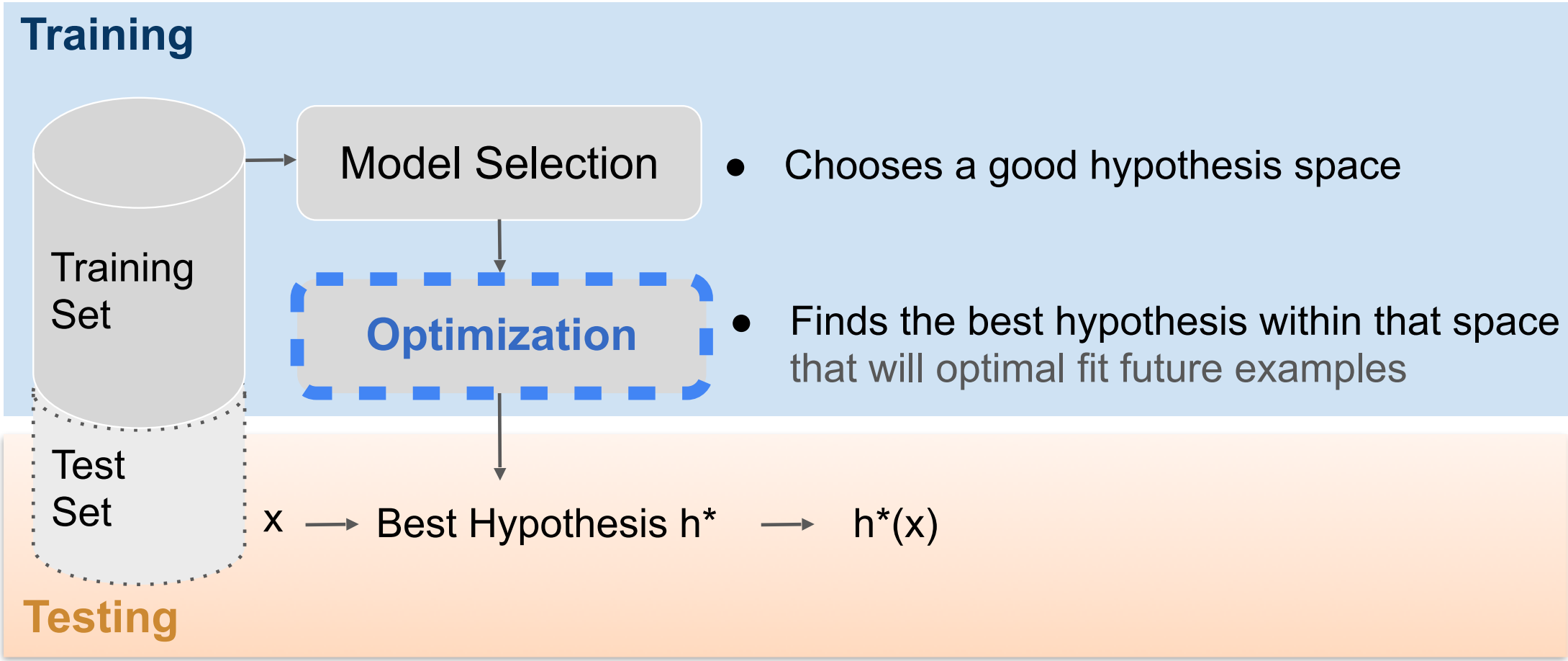
$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$$

- A step that covers all the training examples is called **an epoch**
- Issue: N is larger than the processor's memory

Variations of Gradient Descent

- Stochastic gradient descent (SGD) (1 each round)
  - Randomly selected only one training example for each step
  - It can be used in an online setting, where new data are coming in one at a time
- Mini-batch stochastic gradient descent (middle ground)
  - Randomly selects a small number of training examples at each step, and updates

Learning Framework: Optimization



Issue: Generalization and Overfitting

- Underfitting
  - A function fails to find a pattern in the data
- Overfitting
  - A function pays too much attention to the particular data set it is trained on, causing it to **perform poorly on unseen data**

Regularization

- Minimize the weighted sum of empirical loss and the complexity of the hypothesis to prevent overfitting or underfitting

$$Cost(h) = Loss(h) + \lambda\, Complexity(h)$$

$$\hat{h}^* = \operatorname*{argmin}_{h \in H} Cost(h).$$

penalize model complexity

where a hyperparameter λ is a positive number that serve as a conversion rate between loss and hypothesis complexity

Optimization Strategy: Method of Lagrange Multipliers

- Minimize

$$Loss(h) \quad + \lambda \, Complexity(h)$$

↳ next slide

≡ Minimize

$$Loss(h)$$

subject to $Complexity(h) \leq c$ for some constant c that is related to λ

Example: Regularization

- Regularization function

$$Complexity(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q.$$

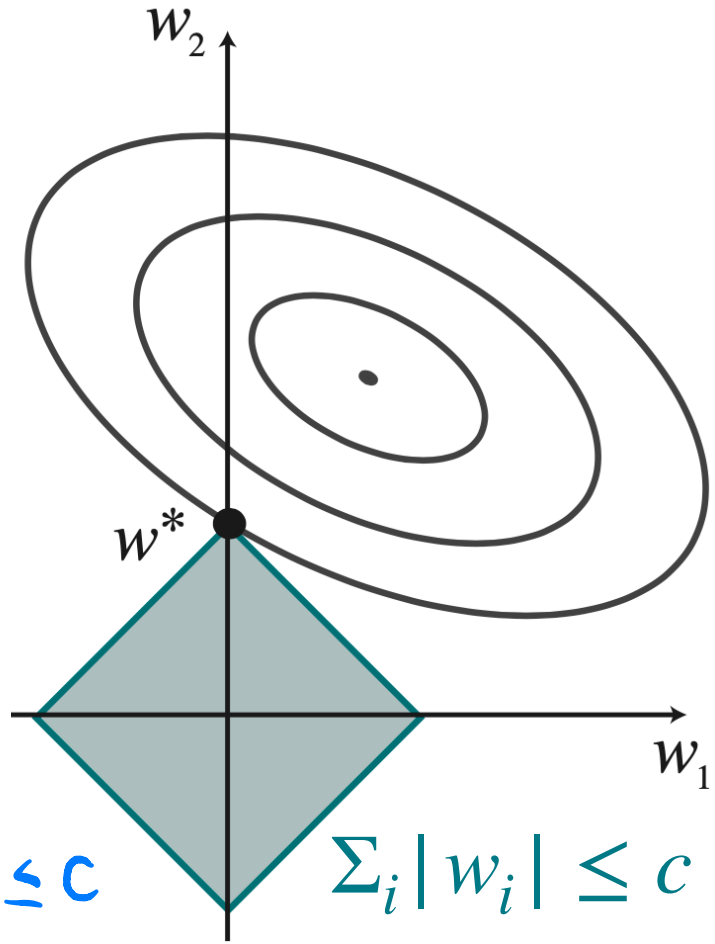
  - $q = 1$ (L_1 regularization, Lasso regression)
    - Minimize the sum of the absolute values
  - $q = 2$ (L_2 regularization, Ridge regression)
    - Minimize the sum of squares

L_1 Regularization (Lasso Regression)

- L_1 regularization

$$L_1(w) = \sum_i |w_i|$$

  - Sparse model
    - Often set many weights to zero
  - Effectively declare the corresponding attributes to be completely irrelevant



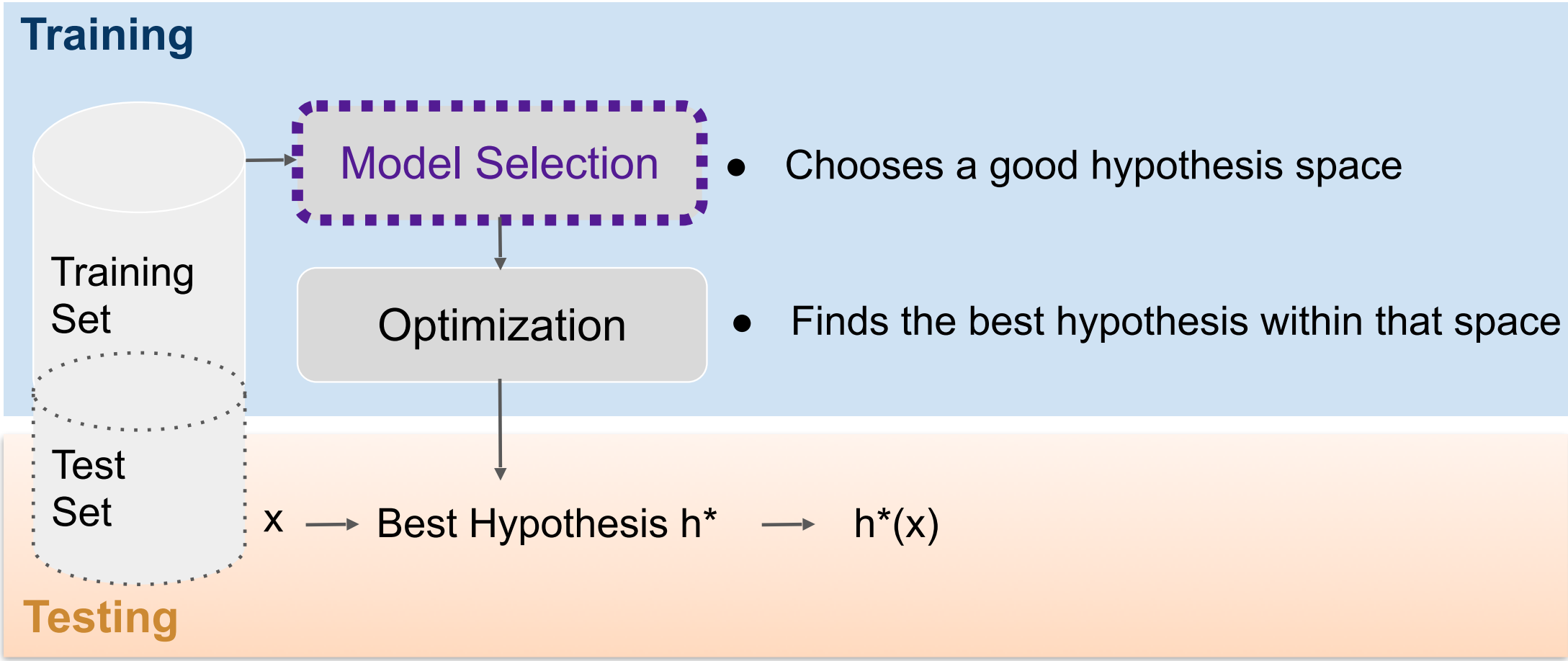
Components of the Graph

1. **Contour Plot**: The elliptical contours represent levels of the loss function $Loss(h_w)$ in the weight space defined by w_1 and w_2. Each contour encircles weight combinations that yield the same loss value, with the innermost contour representing the lowest loss.
2. L_1 **Penalty Region (Diamond Shape)**: The diamond-shaped region represents the constraint set by the L_1 penalty, $\sum |w_i| \leq C$. This region is defined by the absolute values of w_1 and w_2 being less than or equal to a constant C. The corners of the diamond lie along the axes, which reflects the characteristic effect of the L_1 penalty — promoting sparsity by making some weights exactly zero.

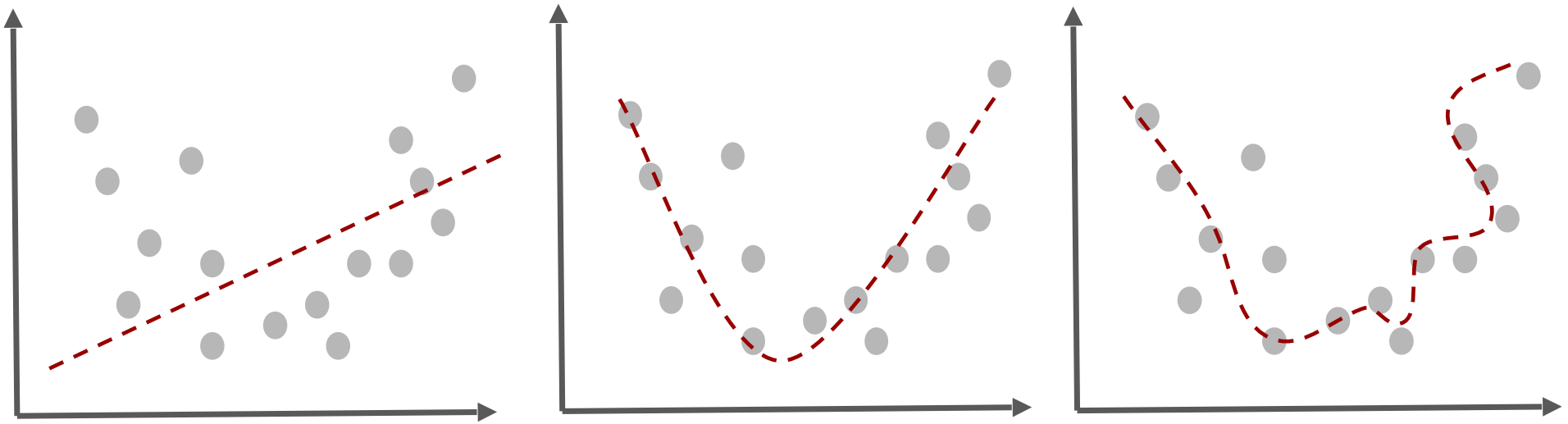
Interaction Between Loss and Regularization

- **Optimal Solution (w^*)**: The point where the innermost loss contour just touches the boundary of the diamond is the optimal solution considering both the loss minimization and the regularization constraint. This point, w^*, represents the weights of the model that balance between fitting the model well to the data (minimizing loss) and maintaining the sparsity imposed by the regularization (keeping the model simple).
- **Sparsity**: Because the L_1 norm promotes sparsity, it's common to find the optimal w^* at the corners or edges of the diamond, leading to one or more weights being zero. This effectively reduces the complexity of the model by ignoring less relevant features (weights).

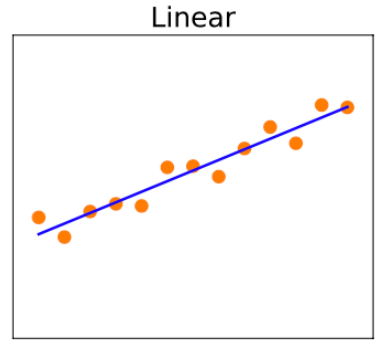
Learning Framework: Model Selection



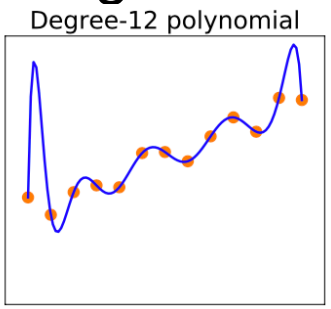
Model Complexity



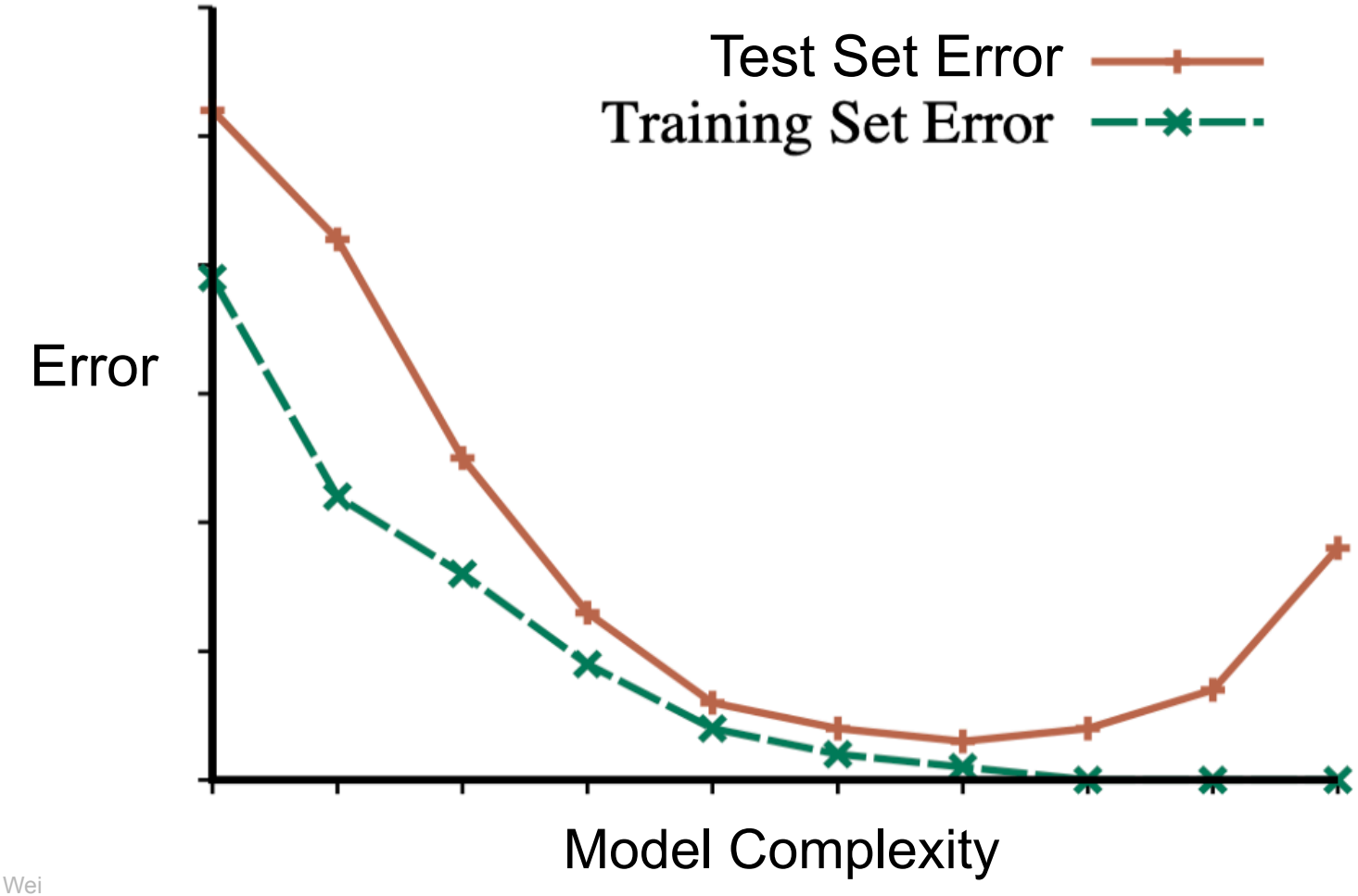
Bias-Variance Tradeoff



- **Bias**
  - An error results from restrictions imposed by the hypothesis space
  - e.g., High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)
- **Variance**
  - An error means the amount of change in the hypothesis due to small fluctuation in the training data
  - e.g., High variance may result from an algorithm modeling the random noise in the training data (overfitting)



Issue: Generalization and Overfitting (Cont.)



*The supreme goal of all theory is to make **the irreducible basic elements as simple and as few as possible** without having to surrender the adequate representation of a single datum of experience.*

— Albert Einstein, 1933

Ockham's Razor:

Plurality [of entities] should not be posited without necessity

— English Philosopher William of Ockham, 14th-Century

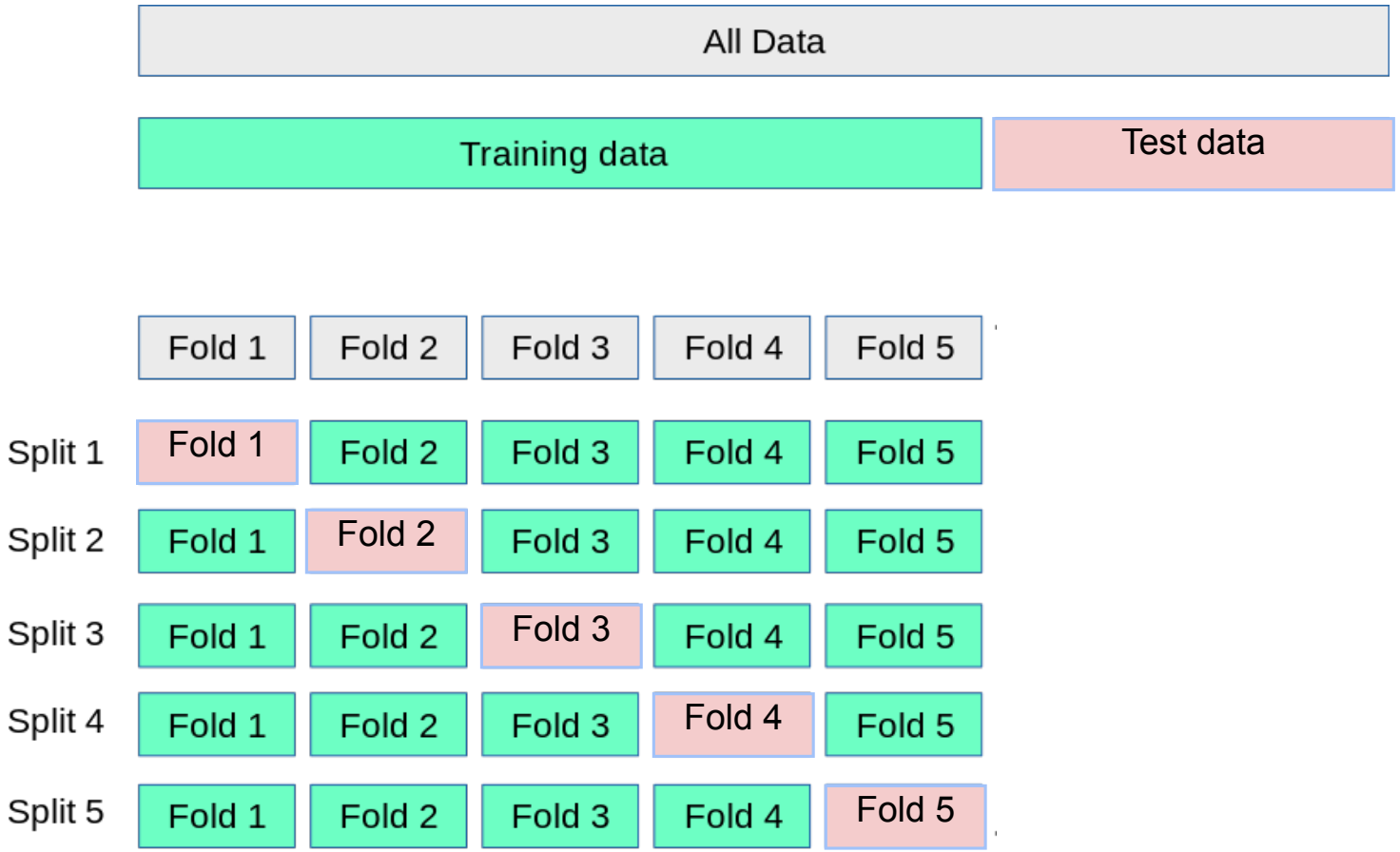
Cross-Validation

- A training set
  - Train candidate models
- A validation set
  - Evaluate the candidate models and choose the best one
- A test set
  - Final unbiased evaluation of the best model

k-fold Cross-Validation

1. Split the data into k equal subsets
2. Perform k rounds of learning
    - Each round:
        - $1/k$ of the data are held out as a validation set
        - The remaining examples are used as the training set
3. Average test set score of the k rounds

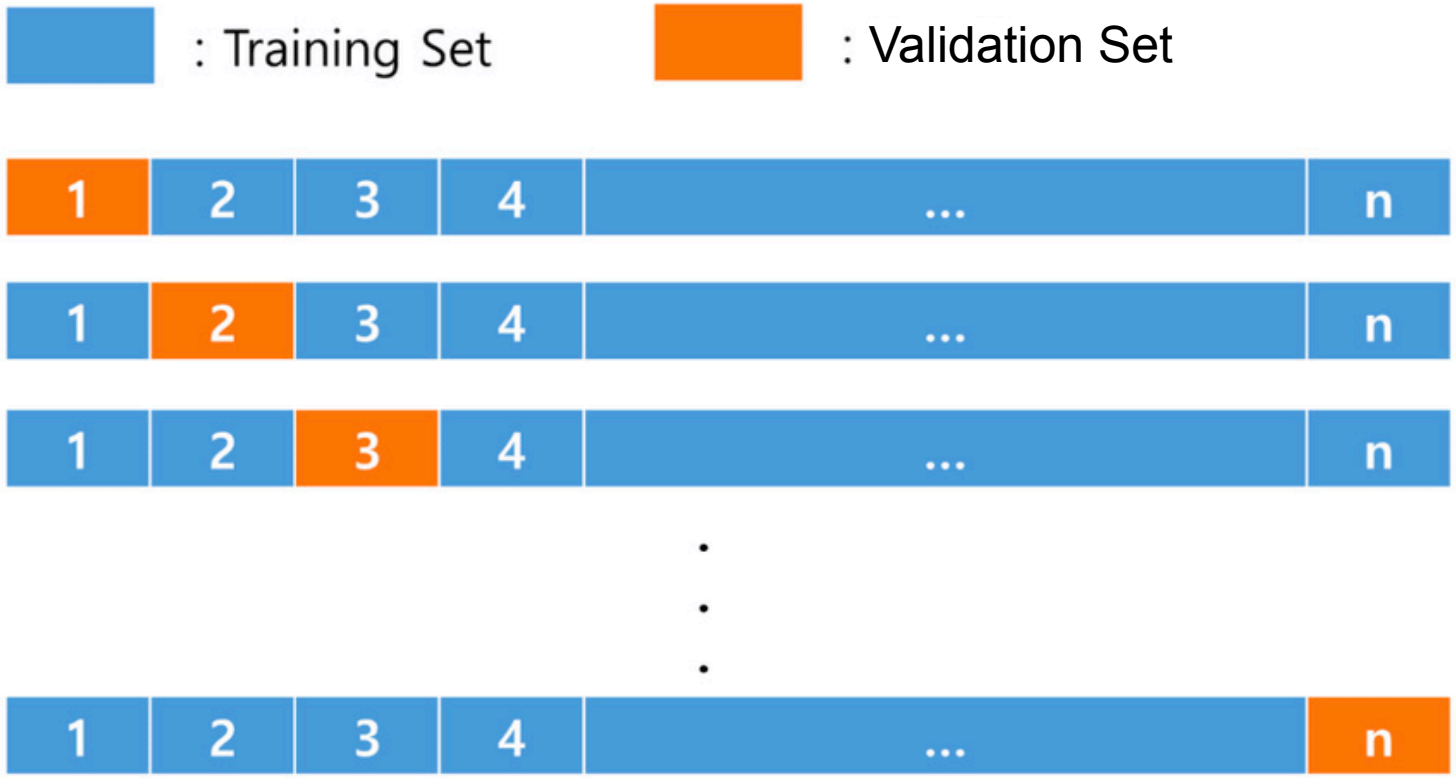
5-fold Cross-Validation



Source: https://scikit-learn.org/stable/modules/cross_validation.html

Leave-One-Out Cross Validation (LOOCV)

- k = n (n: data size)



Source: https://www.researchgate.net/figure/Schematic-representation-of-the-leave-one-out-cross-validation-LOOCV-method_fig1_344613547

Cross Validation Algorithm

function CROSS-VALIDATION($Learner$, $size$, $examples$, k) **returns** error rate

$N \leftarrow$ the number of $examples$
$errs \leftarrow 0$
for $i = 1$ **to** k **do**
  $validation\_set \leftarrow examples[(i-1) \times N/k : i \times N/k]$
  $training\_set \leftarrow examples - validation\_set$
  $h \leftarrow Learner(size, training\_set)$
  $errs \leftarrow errs +$ ERROR-RATE(h, $validation\_set$)
return $errs$ / k // *average error rate on validation sets, across k-fold cross-validation*

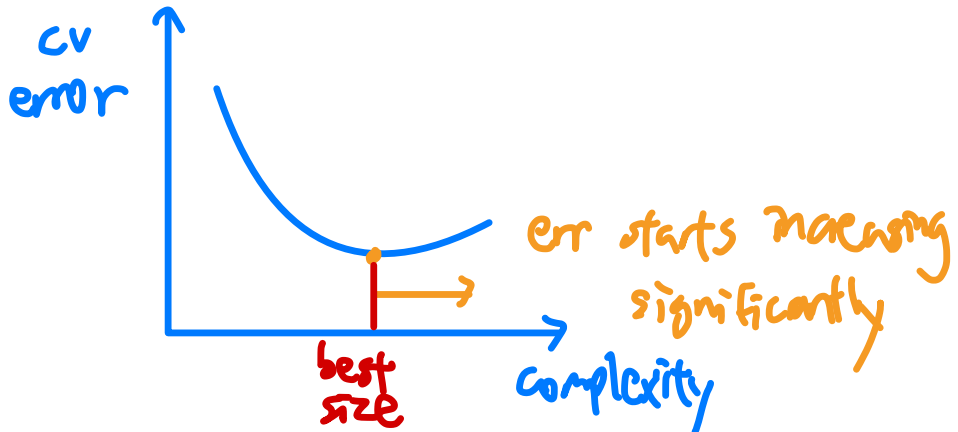
0 : N/k (i = 1)

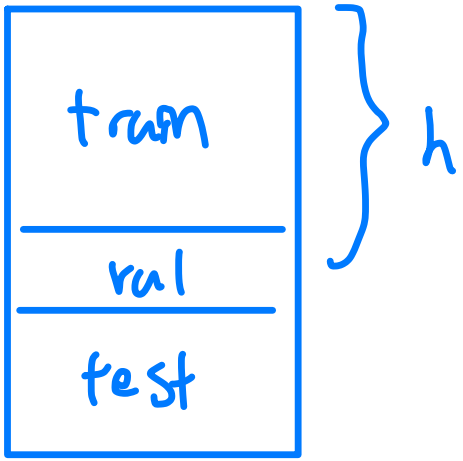
range

Model Selection Algorithm

function MODEL-SELECTION($Learner$, $examples$, k) **returns** a (hypothesis, error rate) pair

$err \leftarrow$ an array, indexed by $size$, storing validation-set error rates
$training\_set,\ test\_set \leftarrow$ a partition of $examples$ into two sets
for $size = 1$ **to** ∞ **do**
  $err[size] \leftarrow$ CROSS-VALIDATION($Learner$, $size$, $training\_set$, k)
  if err is starting to increase significantly **then**
    $best\_size \leftarrow$ the value of $size$ with minimum $err[size]$
    $h \leftarrow Learner(best\_size, training\_set)$
    return h, ERROR-RATE(h, $test\_set$)

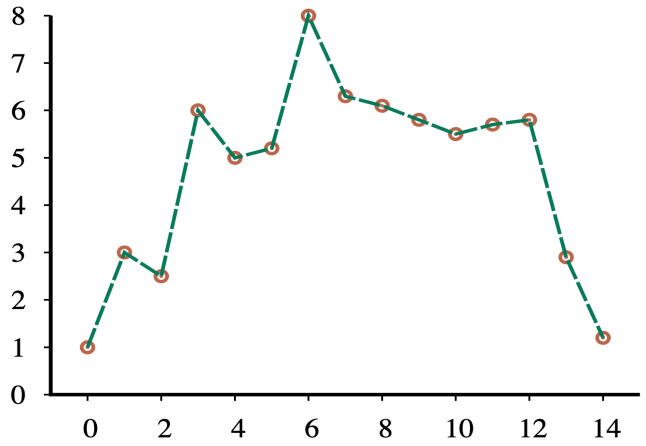




Parametric Models vs. Nonparametric Models

- A parametric model is a learning model that summarizes data with a set of parameters of fixed size
  - e.g., linear regression
    - Estimate a fixed set of parameters **w**

- A nonparametric model is cannot be characterized by a bounded set of parameters
  - e.g., piecewise linear function

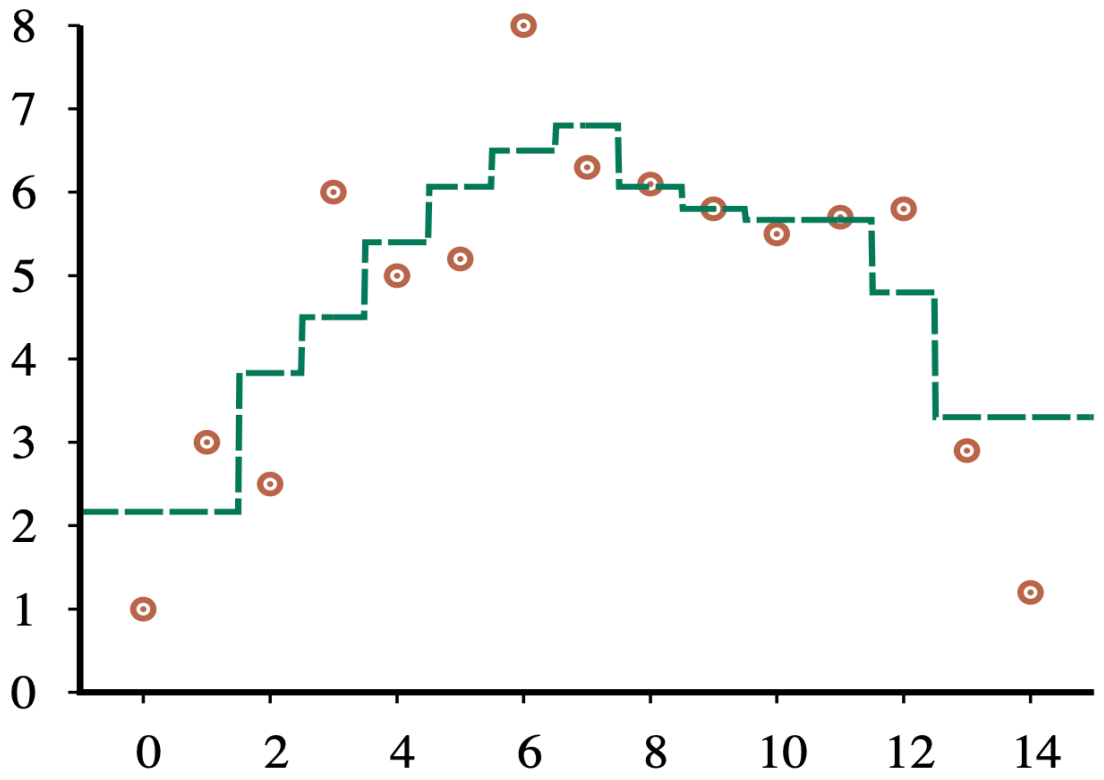


Nonparametric Regression

- Piecewise linear function
- k-nearest-neighbors regression
- Locally weighted regression

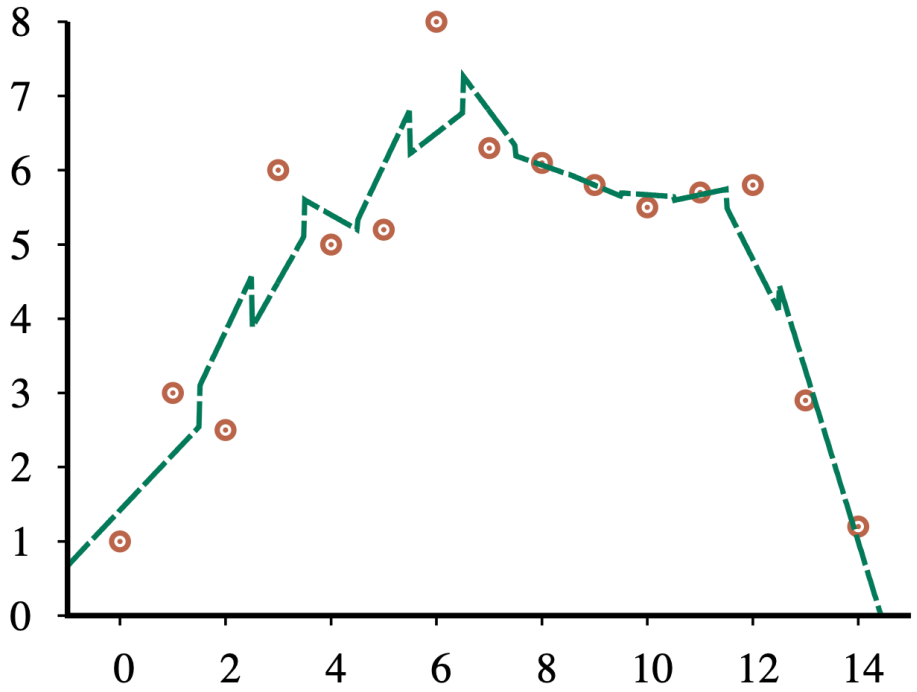
k-Nearest-Neighbors Regression

- k-nearest-neighbors average: $\Sigma y_j / k$
- e.g., $k = 3$



k-Nearest-Neighbors Regression (Cont.)

- k-nearest-neighbors linear regression
  - Find the best line through the k examples
- e.g., $k = 3$



Euclidean Distance

- Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

where *n* is the number of dimensions and x_k and y_k are, respectively, the k^{th} attributes of data object **x**, and **y**

Minkowski Distance

- A generalization of Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r}$$

  where *r* is a parameter, *n* is the number of dimensions (attributes), and x_k and y_k are, respectively, the k^{th} attributes of data object **x**, and **y**
- Examples
  - *r* = 1: Manhattan distance (L_1 norm)
  - *r* = 2: Euclidean distance (L_2 norm)

Locally Weighted Regression

- Idea
  - At each query point $\mathbf{x}_q$, the examples that are close to are weighted heavily, and the examples that are farther away are weighted less heavily
- Kernel function, $K(Distance(\mathbf{x}_j, \mathbf{x}_q))$
  - Decide how much to weight each example $\mathbf{x}_j$
- For a given query point, we solve the following weighted regression problem:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j K(Distance(\mathbf{x}_q, \mathbf{x}_j))\,(y_j - \mathbf{w} \cdot \mathbf{x}_j)^2$$

The answer is $h(\mathbf{x}_q) = \mathbf{w}^* \cdot \mathbf{x}_q$

The objective function for LWR is:

$$w^* = \arg\min_w \sum_j K(x_j, x_q)(y_j - w \cdot x_j)^2$$

Here's a breakdown of the notation:

- $K(x_j, x_q)$: The kernel function (typically Gaussian or other) that assigns a weight based on the distance between each example point x_j and the query point x_q.
- x_j: The feature vector of the j-th data point in the training set.
- y_j: The output (or target) value for the j-th data point.
- w: The parameter vector that we aim to optimize, which represents the coefficients for the features.
- $w \cdot x_j$: The dot product between the parameter vector w and the feature vector x_j, giving the predicted output for x_j.

Thus, w^* is the set of coefficients that minimizes the sum of the weighted squared errors across all training data points, where each weight is based on the distance to the query point x_q.

Locally Weighted Regression (Cont.)

- Example:
  - $K(d) = \max(0, 1 - (2|d|/w)^2)$
  - $w = 10$