CS-583 Deep Learning

Tian Han

Outline

- Course Logistics
- Machine Learning basics
- Linear Regression

Course logistics

- Instructor: Tian Han (than6@stevens.edu)
- Office Hour: 11-12, 2-2:45PM Friday, GS248 or via Canvas Zoom
- Teaching Assistant (TA):

  Section A: Nan Cui (ncui@stevens.edu) OH: TBD

  Mengjiao Zhang(mzhang49@stevens.edu) OH: TBD

  Section B: Jiali Cui (jcui7@stevens.edu) OH: TBD

  Meijie Shih(mshih@stevens.edu). OH: TBD

Prerequisite

In general, you need to have a solid understanding of linear algebra, optimization and prob/stats.

- CS 556: mathematically foundation
- Math 232: Linear algebra
- Python programming is **required.**

If you have not learned Python before, please take CS 515 first!

For the online MS program, please enroll in CS583-WS instead.

Coursework

Grading:

- Homeworks (10%, 15%, 15%, 15%)
- Midterm (20%)
- Final Exam (25%)

Policy:

- Submit the file through Canvas (NOT by email).
- You have a total of 4 late days (not including weekends) for entire class (4 homeworks), but after using 4 late days, NO credit will be given for late submission.
- You are encouraged to work and discuss in a group, but each person have to write down his/her OWN solutions. Don’t cheat on the homeworks and exams.

Machine Learning Basics

Learning from Data

Future Data

Function f

Predict

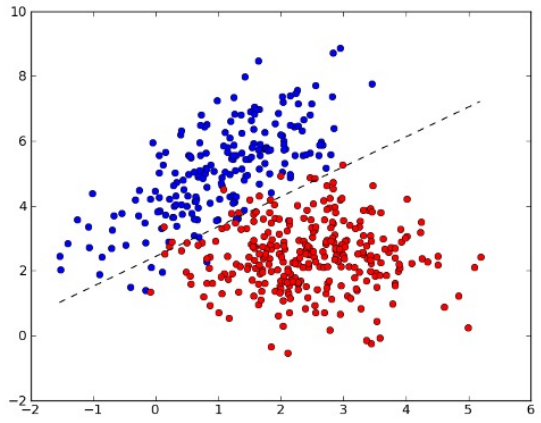
"car"

Train

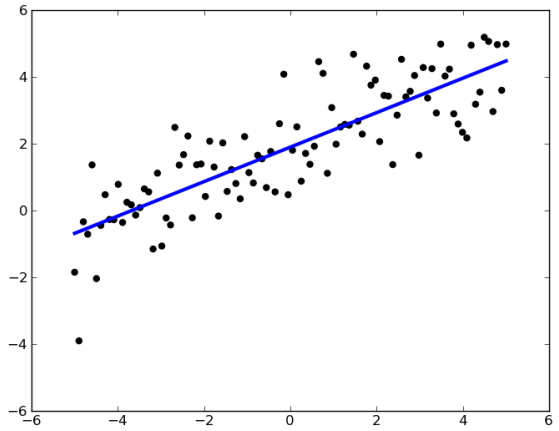
Past Data
(perhaps with **labels**)

Machine Learning Tasks

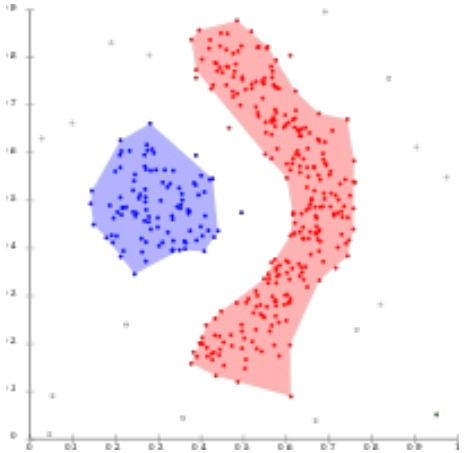
Machine Learning Tasks



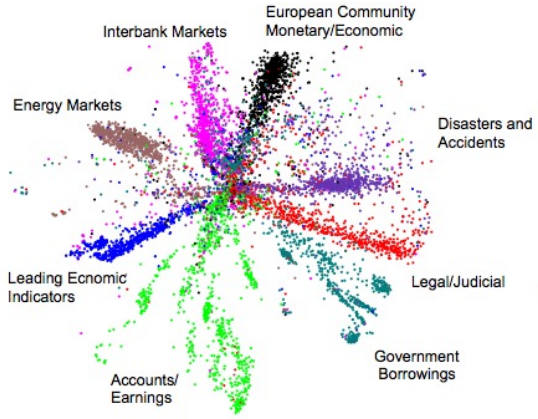
Classification



Regression



Clustering



Dimensionality Reduction

- spam detection
- face recognition
- ...

Classification: Spam Recognition

Sir / Madam,

We invite you to submit your manuscript(s) for publication. The journals include research papers, review articles, technical projects and short communications containing new insight into any aspect of the covered scope of the journal. Our objective is to inform authors of the decision on their manuscript(s) within weeks of submission. After acceptance, the paper will be published in the current issue immediately.
Keywords: English, Literature, Science, Economics, Engineering, Management, Agriculture, Horticulture, Environment .....

International Journal of Advanced Engineering Research and Science (IJAERS) ISSN: 2456-1908(O) | 2349-6495 (P)

DOI (CrossRef): 10.22161/ijaers

Thomson Reuters ResearcherID: P-3738-2015

Impact Factor: 4.192, SJIF: 4.072, IBI: 3.2, PIF: 2.465, ISRA-JIF: 1.317,

Website: http://www.ijaers.com

Kindly submit research articles to **http://ijaers.com/submit-paper/** or mail us at **editor.ijaers@gmail.com**

International Journal of English, Literature and Science (IJELS) ISSN: 2456-7620

New Email

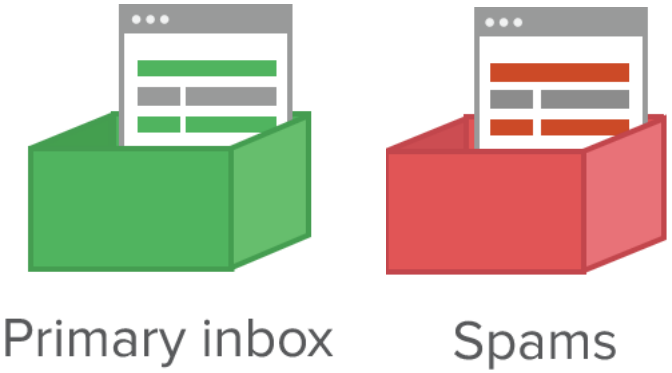
Classifier

Predict

Is Spam

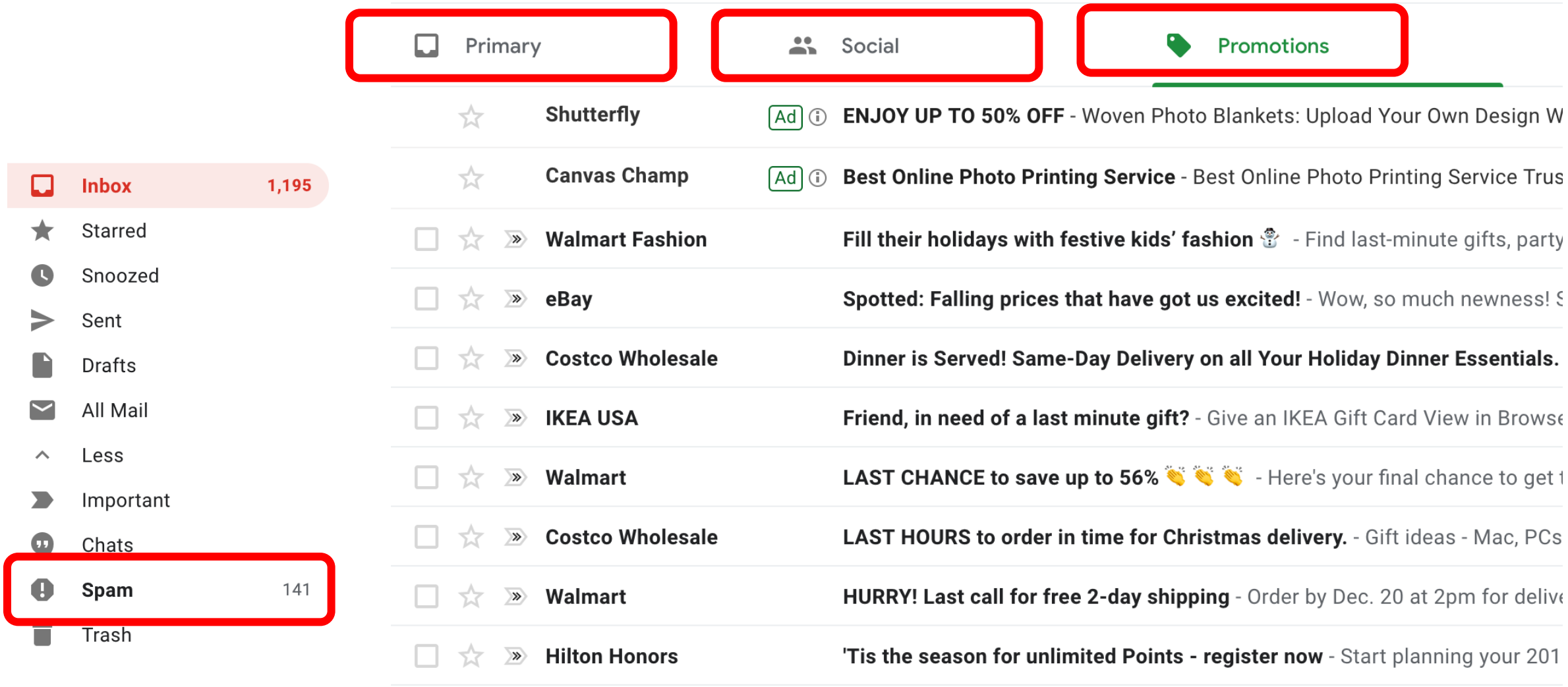
Train

Binary classification



Emails with Labels

Classification: Spam Recognition



Classification: Face Recognition

Classifier

Predict

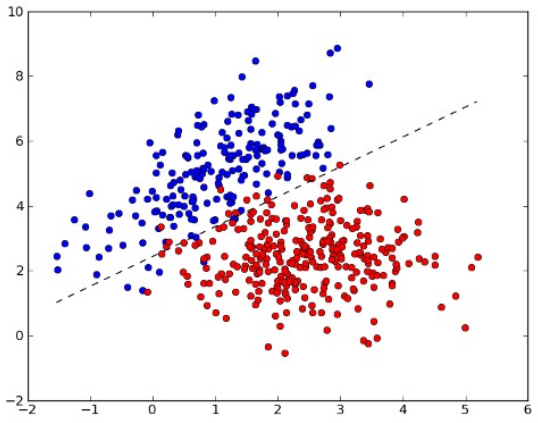
"Matt Damon"

Train

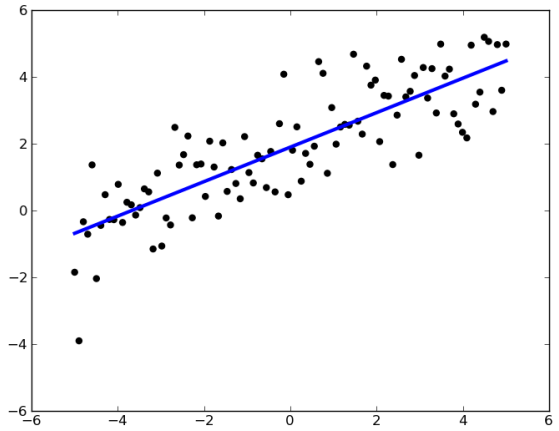
multi-class

Faces & Names

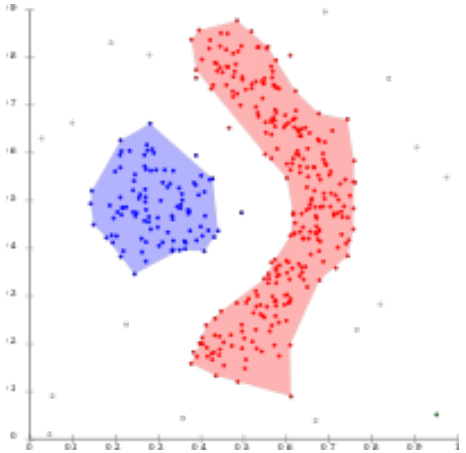
Machine Learning Tasks



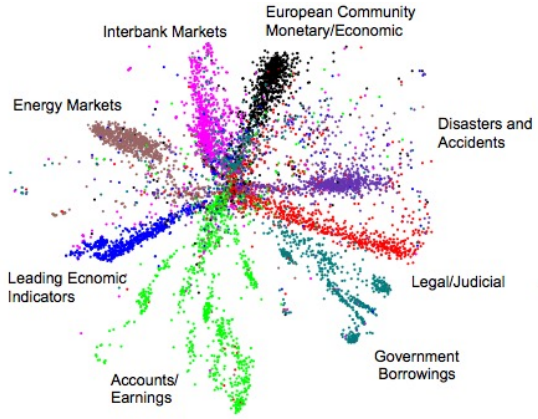
Classification



Regression



Clustering



Dimensionality Reduction

Regression: Housing Price

Features of a House

Regressor

Predict

Price = $0.5M

Train

label (target)

features

Price	Selling price of the property ($1,000)
Beds	Number of bedrooms in the house
Baths	Number of bathrooms in the house
Square Feet	Size of the house in square feet
Miles to Resort	Miles from the property to the downtown resort area
Miles to Base	Miles from the property to the base of the ski resort's mountain
Acres	Lot size in number of acres
Cars	Number of cars that will fit into the garage
Years Old	Age of the house, in years, at the time it was listed
DoM	Number of days the house was on the market before it sold

Regression VS Classification

- **Regression: labels are ordered**
- Example:

House Prices: $324K < $521K < $1.2M



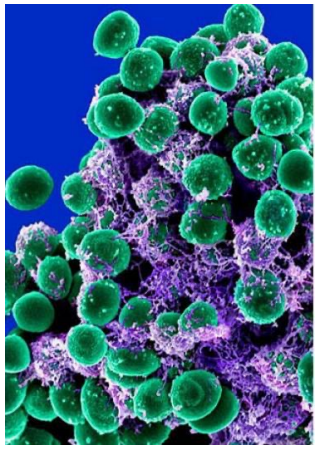
Regression VS Classification

- Regression: labels are ordered
- **Classification: labels are categorical**
- Example:

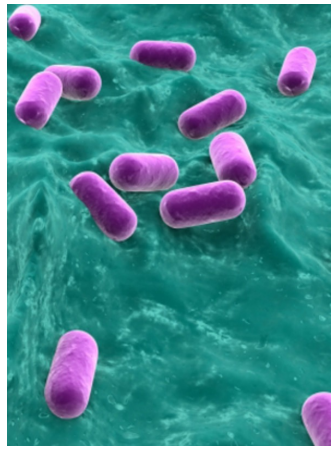
Categories:

Class 1

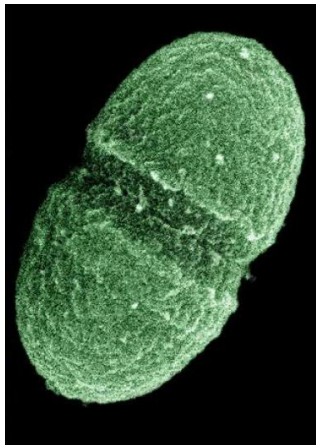
Class 2



Class 3



Class 4



Regression VS Classification

- Regression: labels are ordered
- **Classification: labels are categorical**
- Example:

Categories: Class 1 Class 2 Class 3 Class 4

"`Class 1`" and "`Class 4`" are categories. They are not ordered!

Class 1 ≠ Class 4

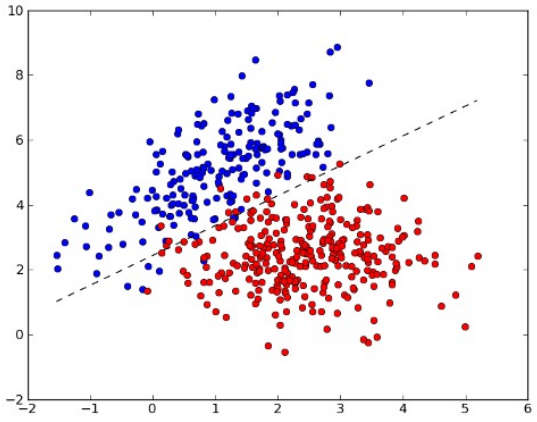
Regression VS Classification

Problem: Wine Quality Prediction.

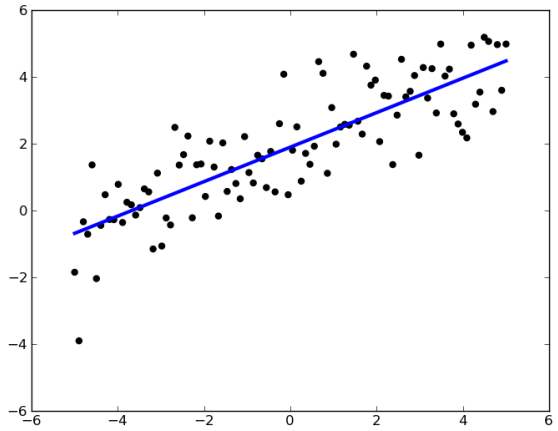
- Features: Measurements of chemicals in the wine.
- Targets: Outstanding, very good, good, fair, not recommended.

- Is wine quality prediction a regression or classification problem?

Machine Learning Tasks

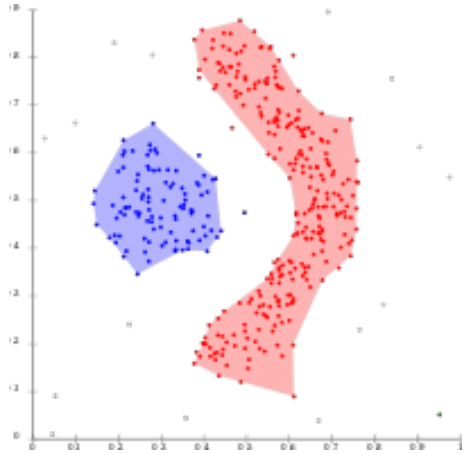


Classification



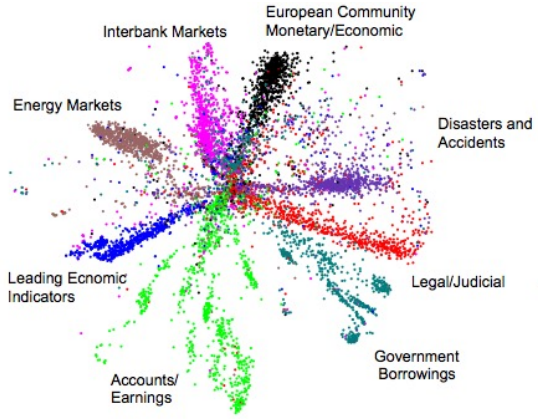
Regression

Supervised Learning



Clustering

Unsupervised Learning



Dimensionality Reduction

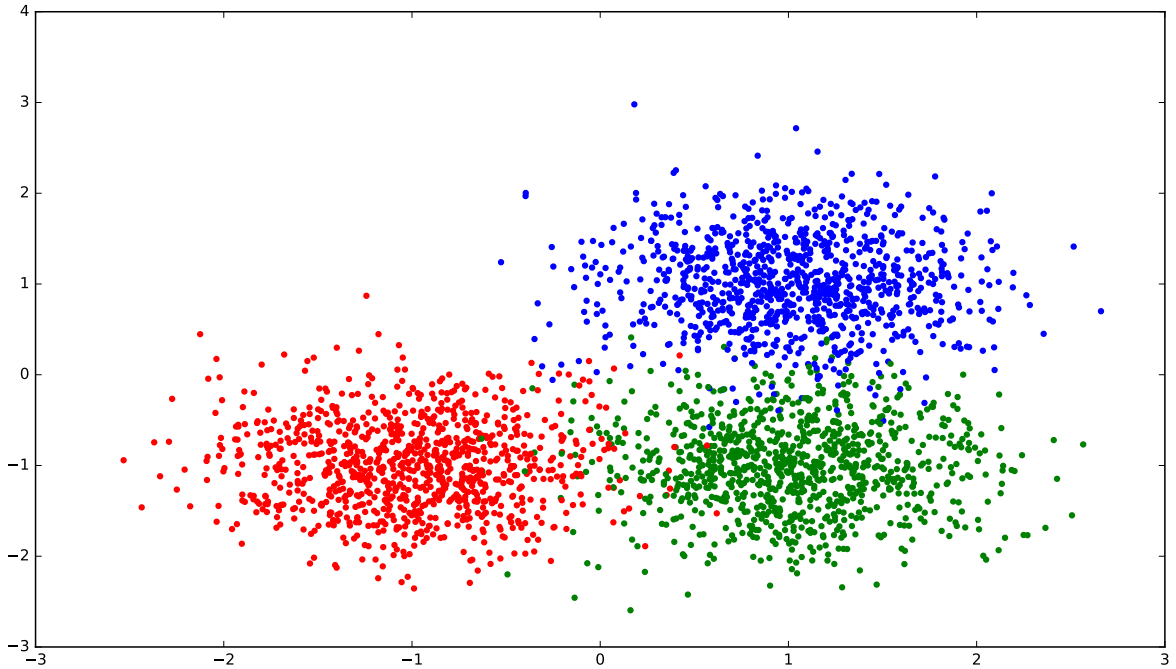
Unsupervised Learning

Supervised Learning

Clustering: Unsupervised Learning

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n$ and number of clusters k ($\ll n$).

Output: predicted classes $y_1, \cdots, y_n \in \{1, 2, \cdots, k\}$.



Clustering: Unsupervised Learning

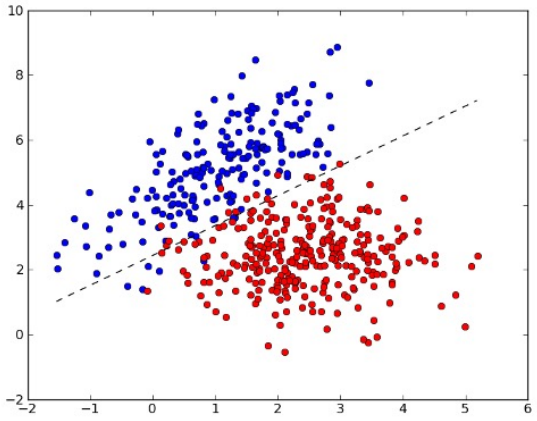
Supervised learning : learn function from **labeled** training data.

- Classification and regression are supervised learning tasks.

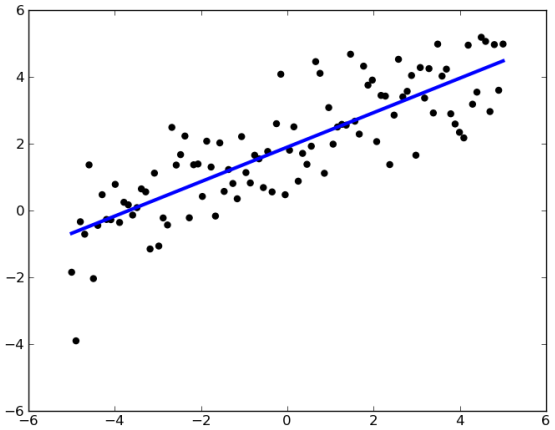
Unsupervised learning : draw inferences from datasets consisting of data **without label**.

- Clustering is an unsupervised learning task.

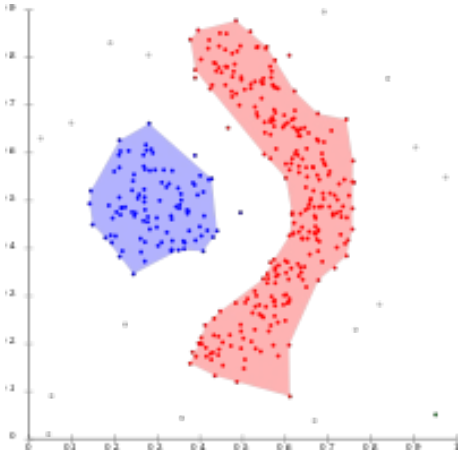
Machine Learning Tasks



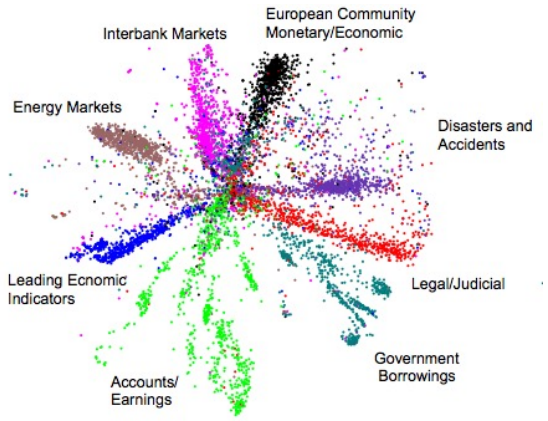
Classification



Regression



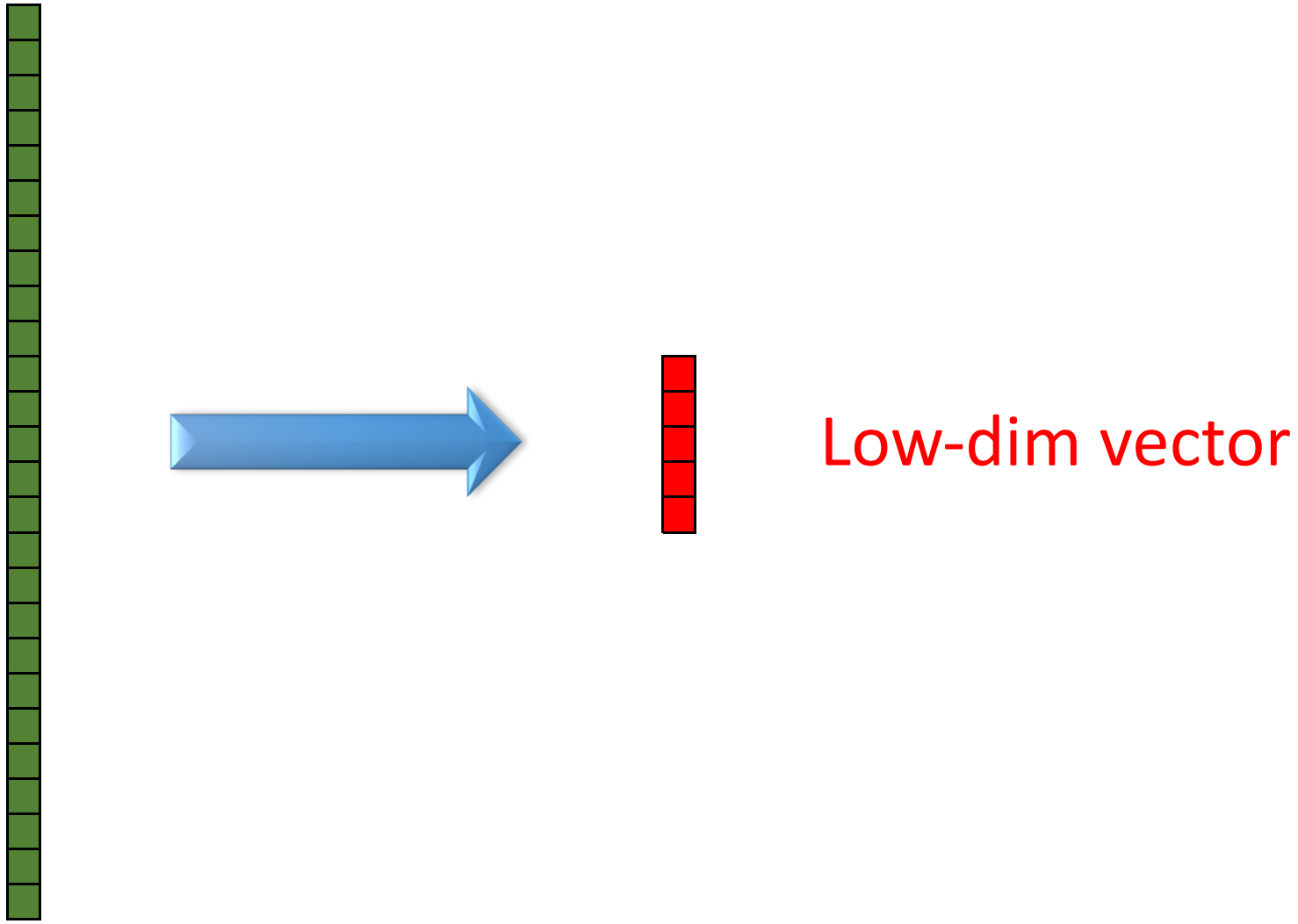
Clustering



Dimensionality Reduction

Dimensionality Reduction

High-dim vector



Low-dim vector

Dimensionality Reduction

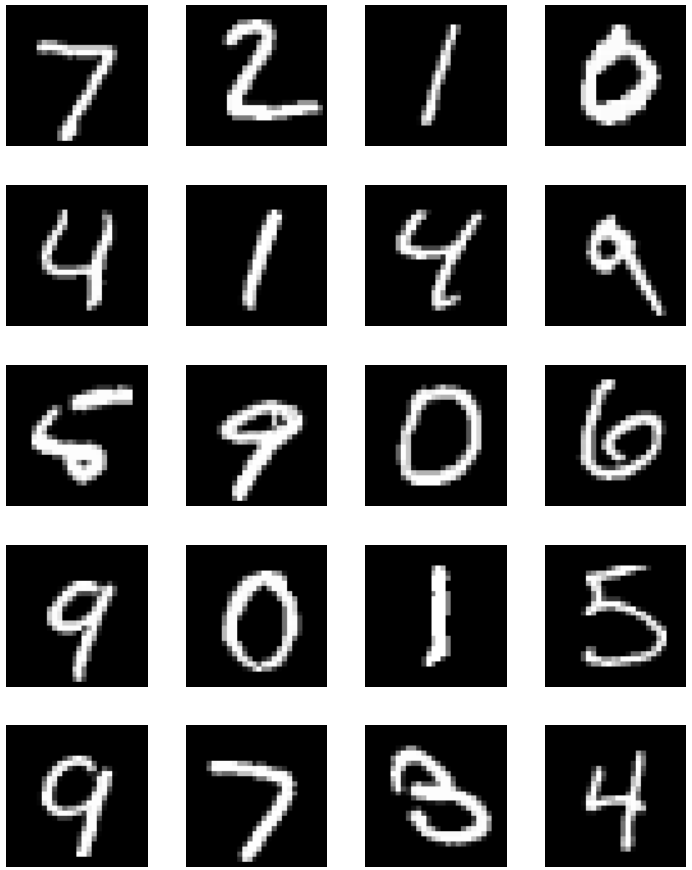
- Applications
  - Visualization and analysis
  - Data processing (to make downstream ML more efficient or accurate)

Dimensionality Reduction

- Applications
  - Visualization and analysis
  - Data processing (to make downstream ML more efficient or accurate)

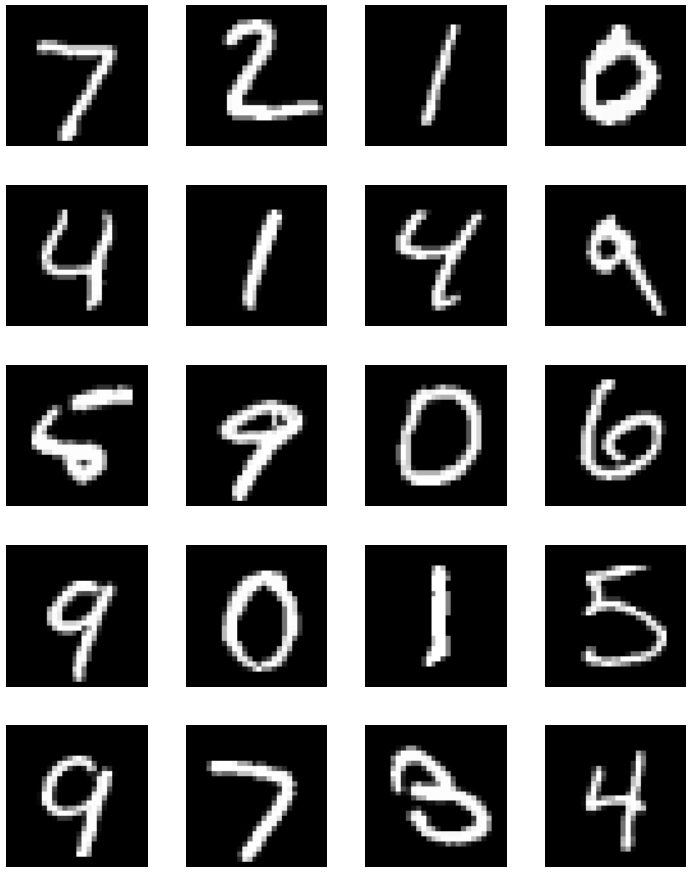
- Methods
  - Principal component analysis (PCA)
  - Manifold learning
  - Autoencoder
  - Fisher linear discriminant analysis (FLD)

Autoencoder for Hand-Written Digit Data

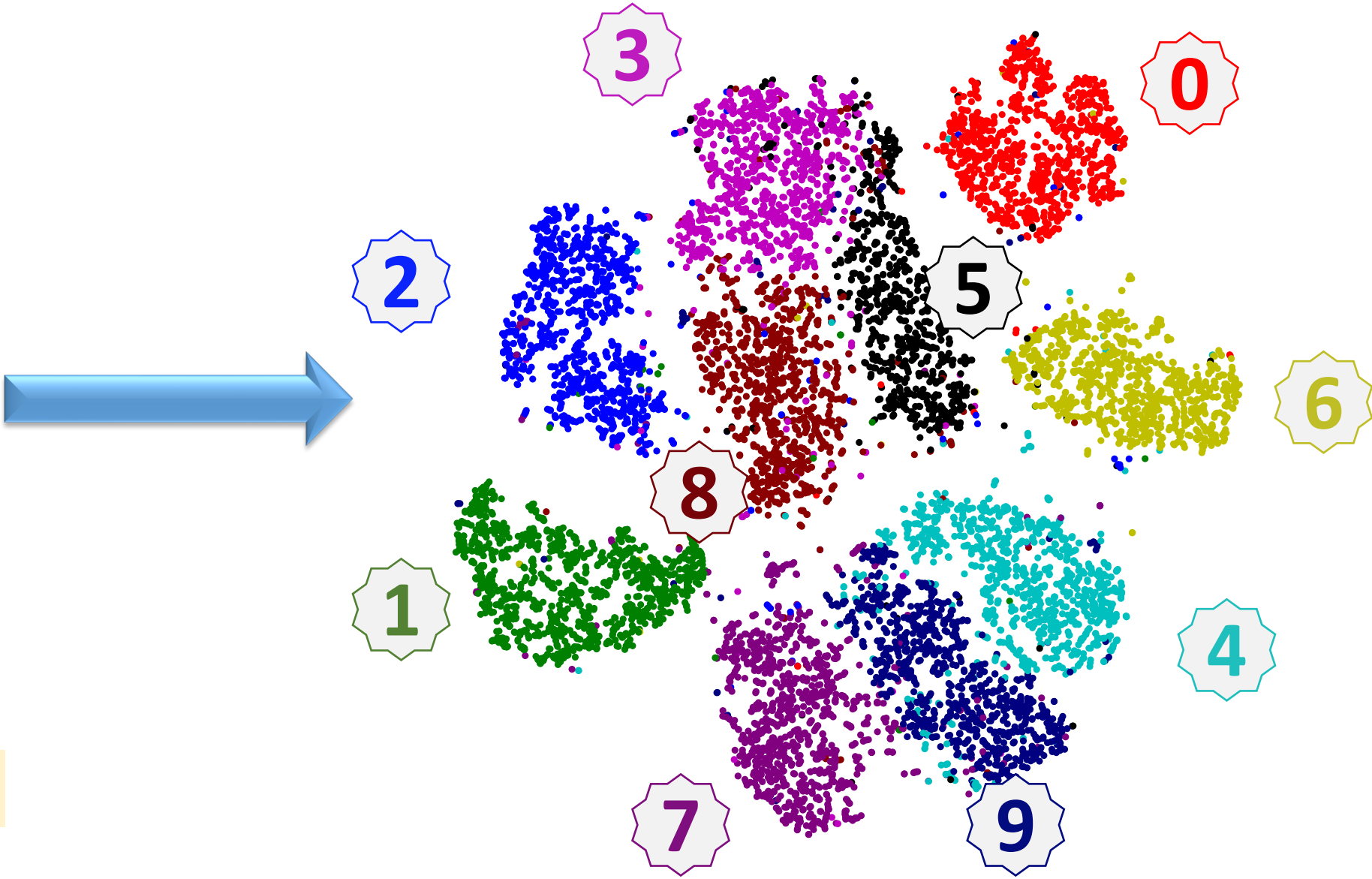


60K images (size 28×28)

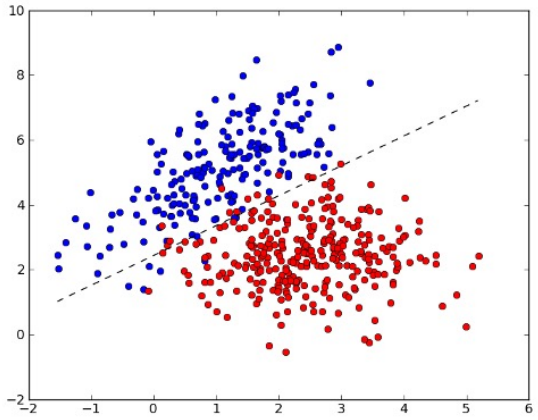
Autoencoder for Hand-Written Digit Data



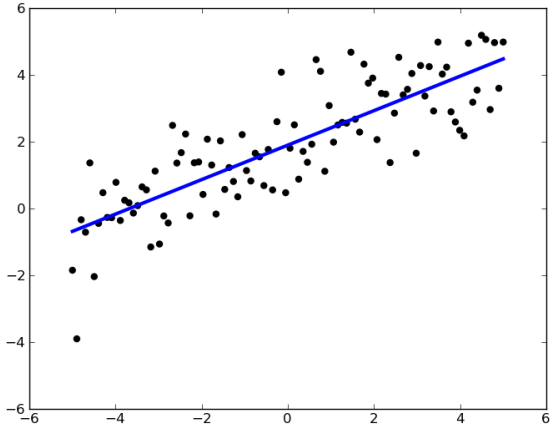
60K images (size 28×28)



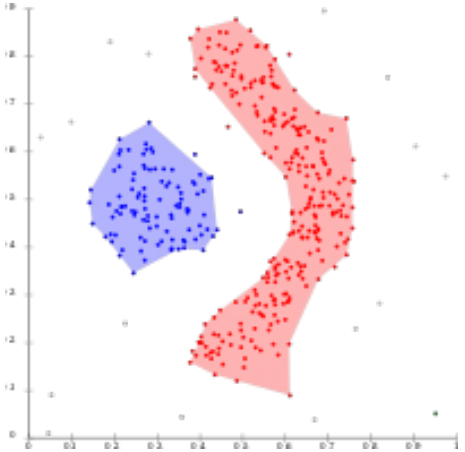
Machine Learning Tasks



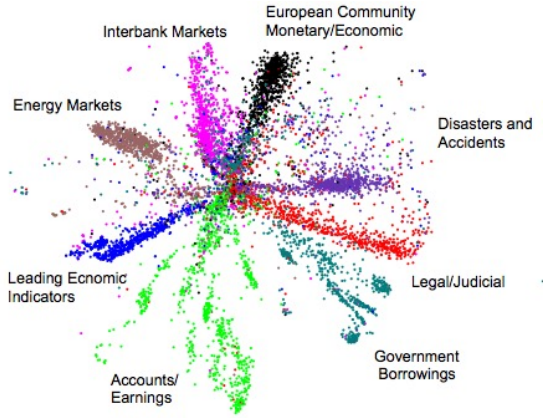
Classification



Regression

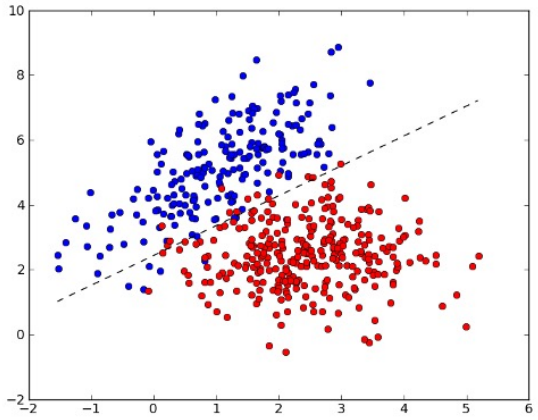


Clustering

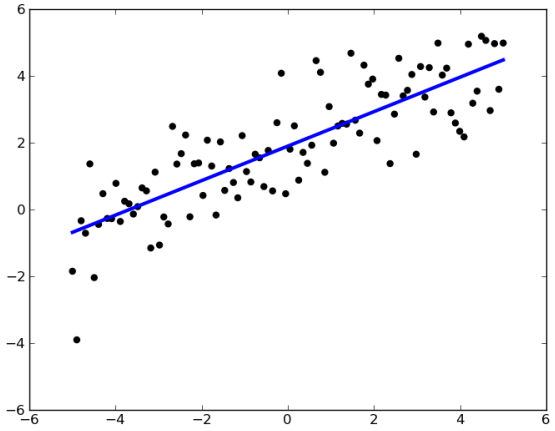


Dimensionality Reduction

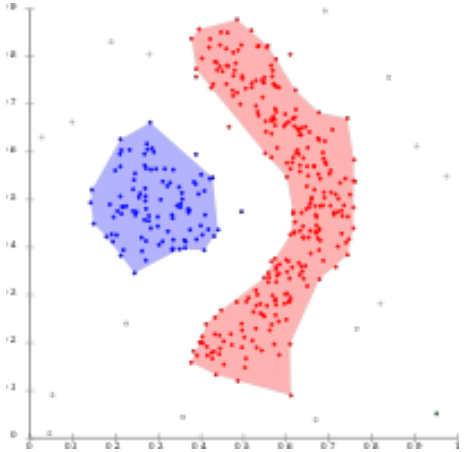
Machine Learning Tasks



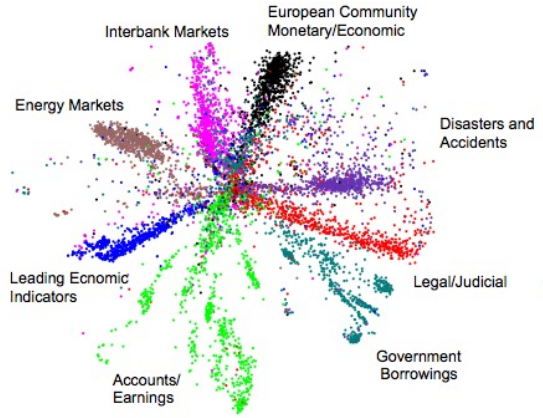
Classification



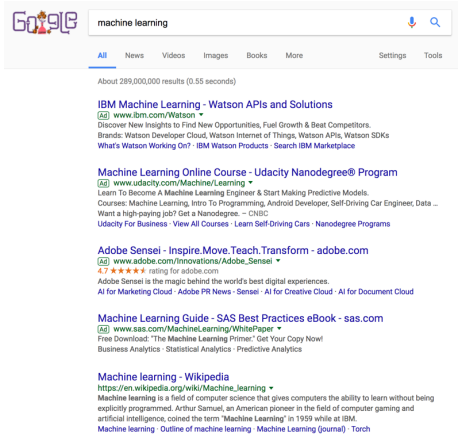
Regression



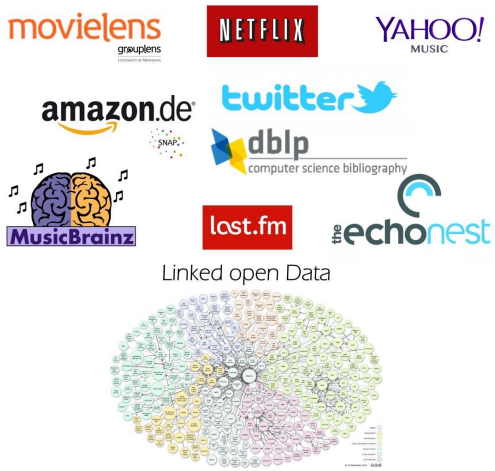
Clustering



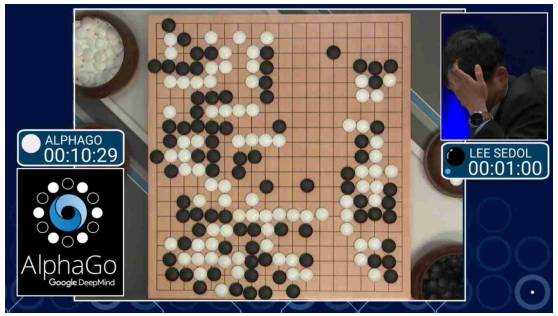
Dimensionality Reduction



Ranking



Recommendation



Reinforcement Learning

Tasks, Methods, & Algorithms

Tasks
(or problems)

- Regression
- Classification
- Clustering
- Dim Reduction

Tasks
(or problems)

- Regression
- Classification
- Clustering
- Dim Reduction

Methods
(or models)

- Neural Networks
- SVM
- Logistic Regression
- Decision Tree
- Nearest Neighbor

Tasks
(or problems)

- Regression
- Classification
- Clustering
- Dim Reduction

Methods
(or models)

- Neural Networks
- SVM
- Logistic Regression
- Decision Tree
- Nearest Neighbor

Algorithms

- Gradient Descent (GD)
- Stochastic GD
- Coordinate Descent (CD)
- Dual CD

Modeling & Computation

Learning from Data

Future Data

Model

Predict

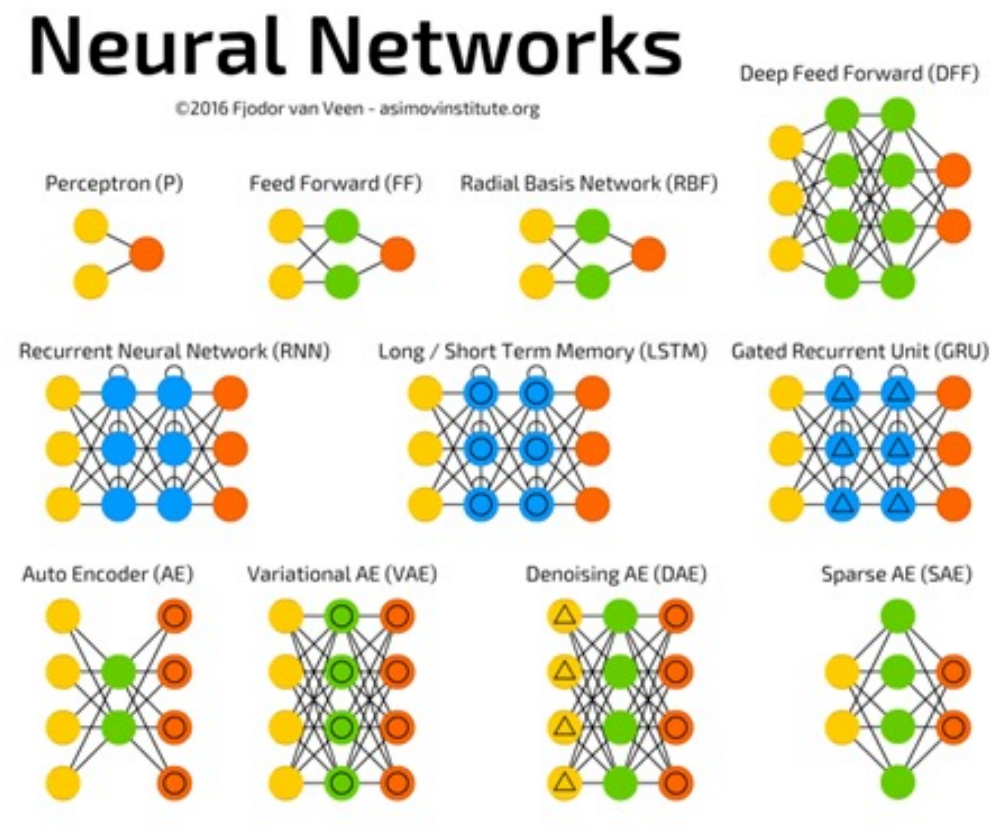
"car"

Train

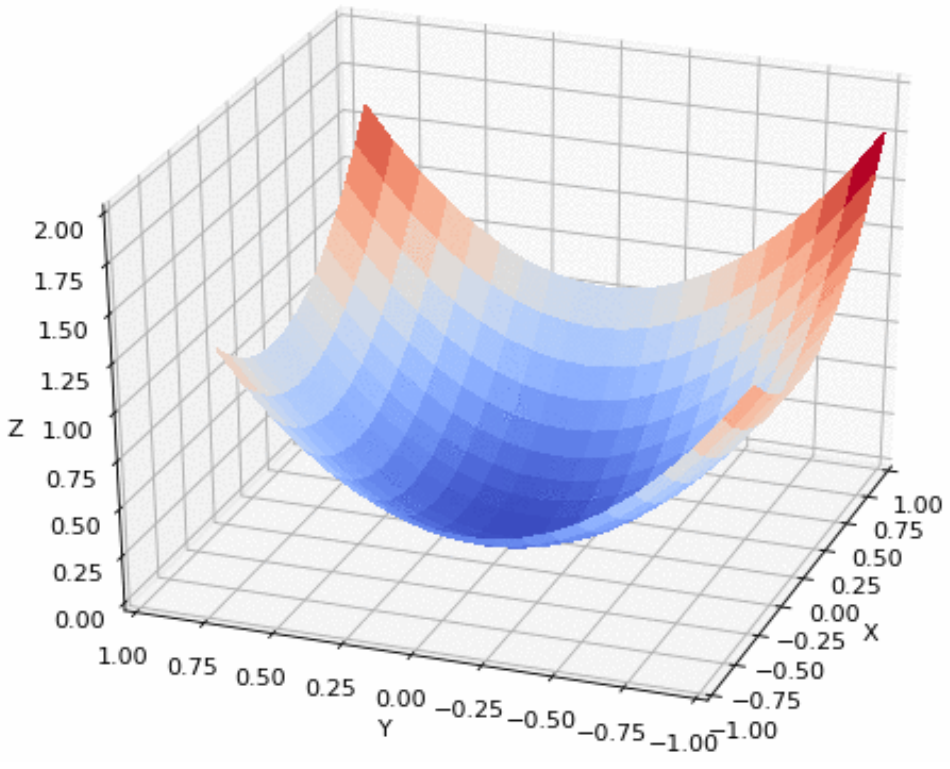
Past Data
(perhaps with **labels**)

Machine Learning in Practice

Modeling



Computation



Modeling

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$

# of bedrooms

# of bathrooms

square feet

$\vdots$

age of house

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix} \quad \begin{matrix} \text{\# of bedrooms} \\ \text{\# of bathrooms} \\ \text{square feet} \\ \vdots \\ \text{age of house} \end{matrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_d \end{bmatrix}$$

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

  - $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$
  - $w_1, w_2, \cdots, w_d$: weights

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

Question: How to find $\mathbf{w}$?

training is about finding the best model weights
then we use the optimal model weigths with the testing data and we get the testing output

Price = $0.5M

Features of a House

$$\mathbf{x} \in \mathbb{R}^d$$

Prediction:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$$

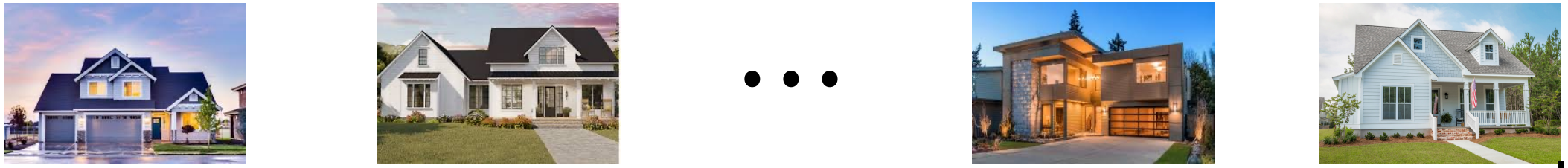
Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

Question: How to find $\mathbf{w}$?

- Training features: $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$.
- Training targets: $y_1, \cdots, y_n \in \mathbb{R}$.

for training we have input feature vector x and the target value y (house price in this example



totally n houses

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

Question: How to find $\mathbf{w}$?

- Training features: $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$.
- Training targets: $y_1, \cdots, y_n \in \mathbb{R}$.
- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T \mathbf{w} - y_i\right)^2$.

model prediction - the target value

if we are super accurate means the difference is zero
hence we want to minimize this to make the model even better
if model is large then we need to tune it to make it closer to actual output

Linear Models

- Feature vector: $\mathbf{x} \in \mathbb{R}^d$ (e.g., features of a house).
- Prediction: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ (e.g., housing price).

Question: How to find $\mathbf{w}$?

- Training features: $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$.
- Training targets: $y_1, \cdots, y_n \in \mathbb{R}$.
- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T \mathbf{w} - y_i\right)^2$.

least square regression is minimizing this difference

- Least squares regression: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.

w* is optimal model weights

Linear Models Are Not Expressive

Example: Given a person's photo, predict her/his age.

Photo (features)

Age = 36

if we tried to use the pixels of the photograph say its 100x100 and with 3 rgb channels as the features and then dot product it with the weight we would have 30K computations to do which isnt useful and is too cumbersome for an ML model so we need to extract only the useful features to make this model work

Linear Models Are Not Expressive

Example: Given a person's photo, predict her/his age.

Photo (features)

Age = 36

Question: Can we use linear regression?

- Linear regression assumes the target is a weighted average of every pixel in the photo.
- Linear regression works poorly for age prediction.

Traditional Approaches

feature extractors

SIFT Feature

HOG Feature

LBP Feature

gives high level featues

Traditional Approaches

SIFT Feature

HOG Feature

LBP Feature

this time x transpose w doesnt represent raw features but represents the high level features which were extracted

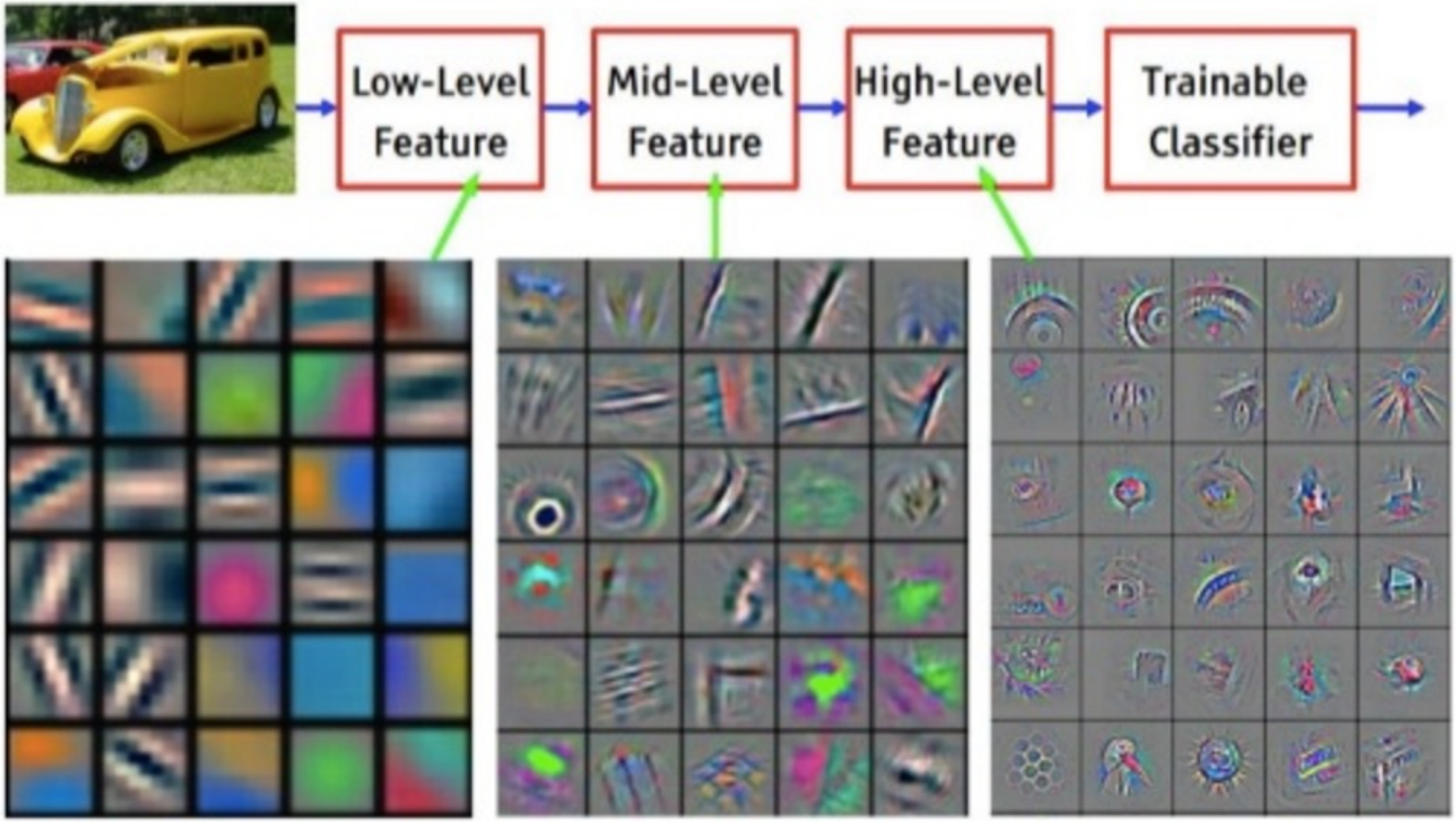
Linear Model

"dog"

Convolutional Neural Networks (CNNs)

much more powerful than traditional approach. we let the model learn from the data. high level = semantic

traditional we design feature extractors where as in CNN with let it do it automatica lly



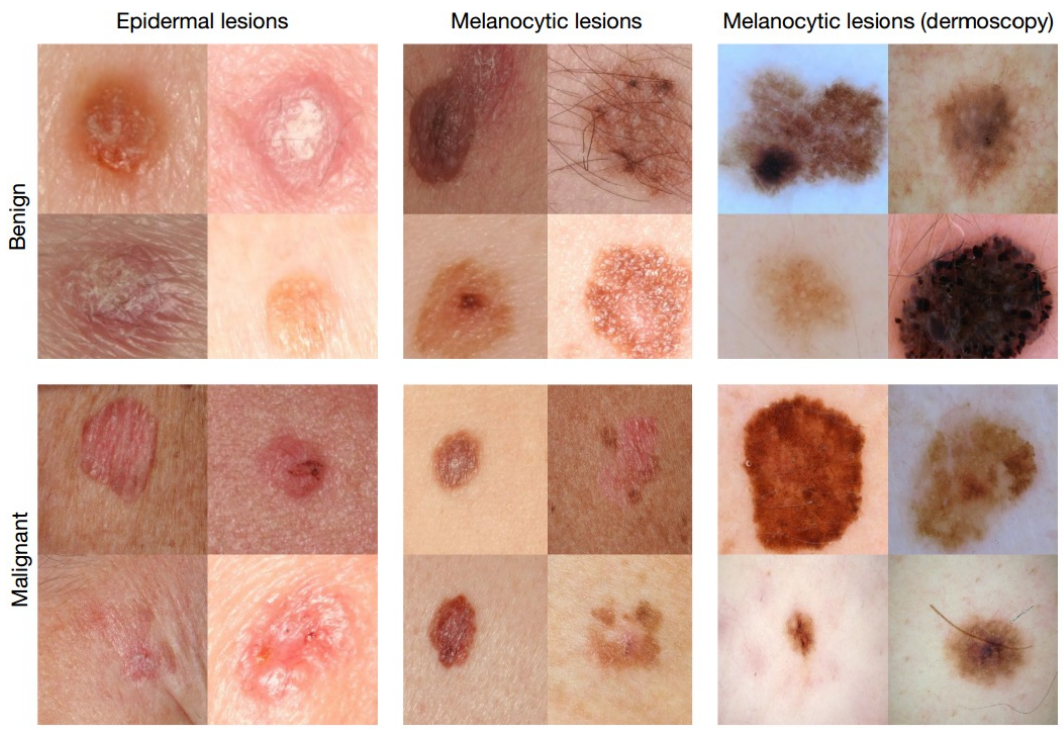
Applications of CNNs

feature extraction is automatic from the input data

- CNNs are suitable for image data.
- CNNs convert images to effective representations. (Feature extraction.)

- Applications:
  - Image/video recognition.
  - Face recognition.
  - Image generation.
  - …

Applications of CNNs: Medical Diagnosis



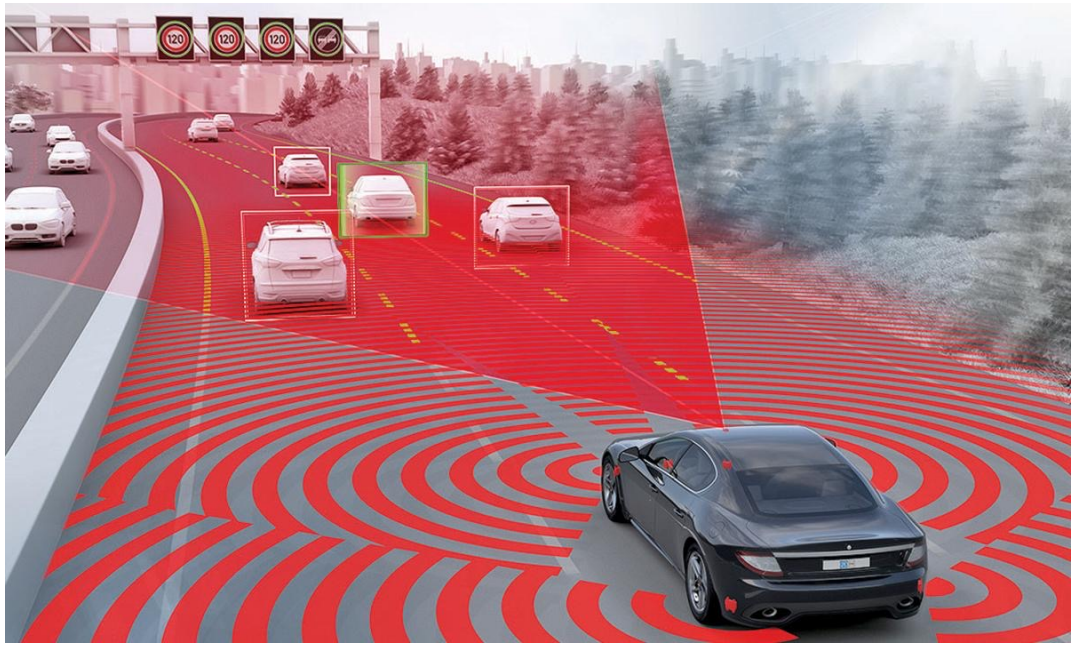
Example: Skin cancer diagnosis

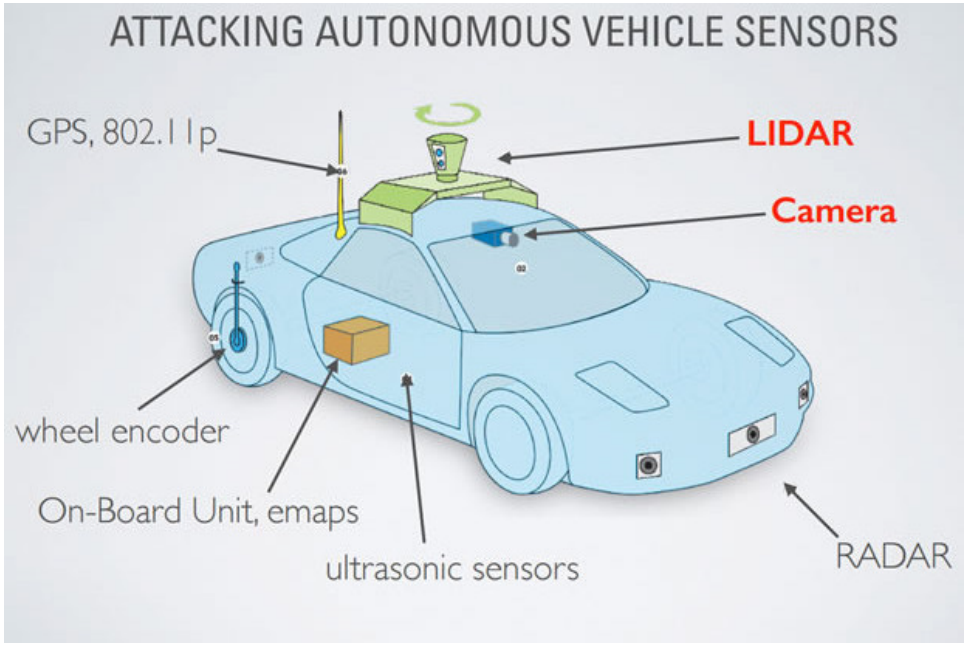
- Input: an image.
- Outputs:
  - Is it skin cancer?
  - Benign or malignant?
- The same accuracy as human experts.

Reference

[1] Esteva et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.

Applications of CNNs: Self Driving Cars



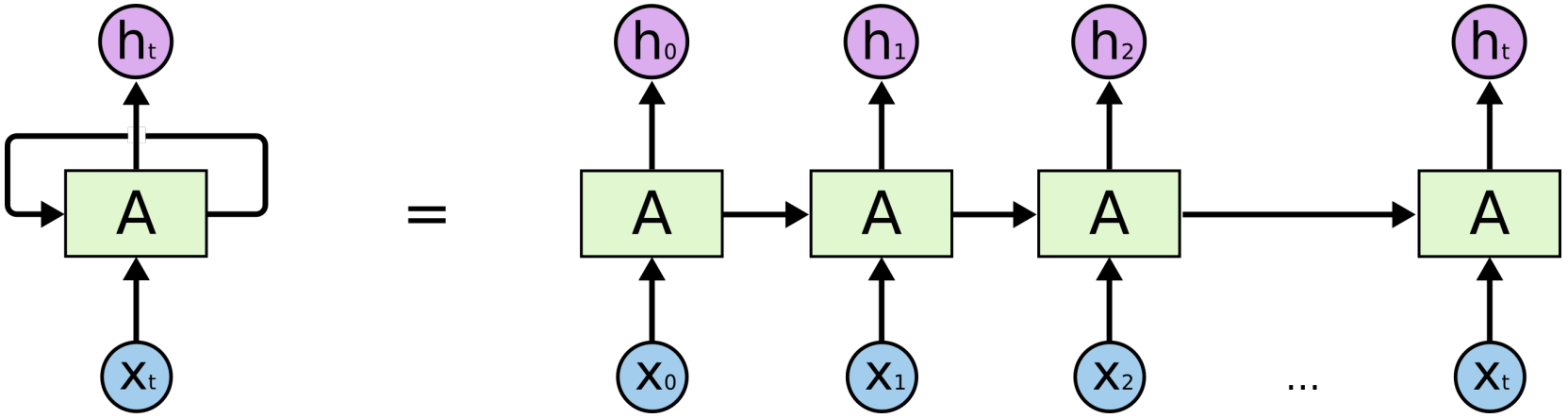


- CNNs play an import role in self driving cars.
  - Understand images taken by the cameras.
  - Recognize signs, cars, pedestrians, and obstacles.
- CNN is not everything; self-driving car is a sophisticated system.

Recurrent Neural Networks (RNNs)

CNN for image data where as RNN for sequential data

- RNNs naturally fit sequence data, e.g.,
  - time series data (e.g., stock price, weather),
  - text data,
  - speech data…



Applications of RNNs: Machine Translation

Applications of RNNs: Machine Translation

cnn recognise the sentence and RNN translates the sentence for you

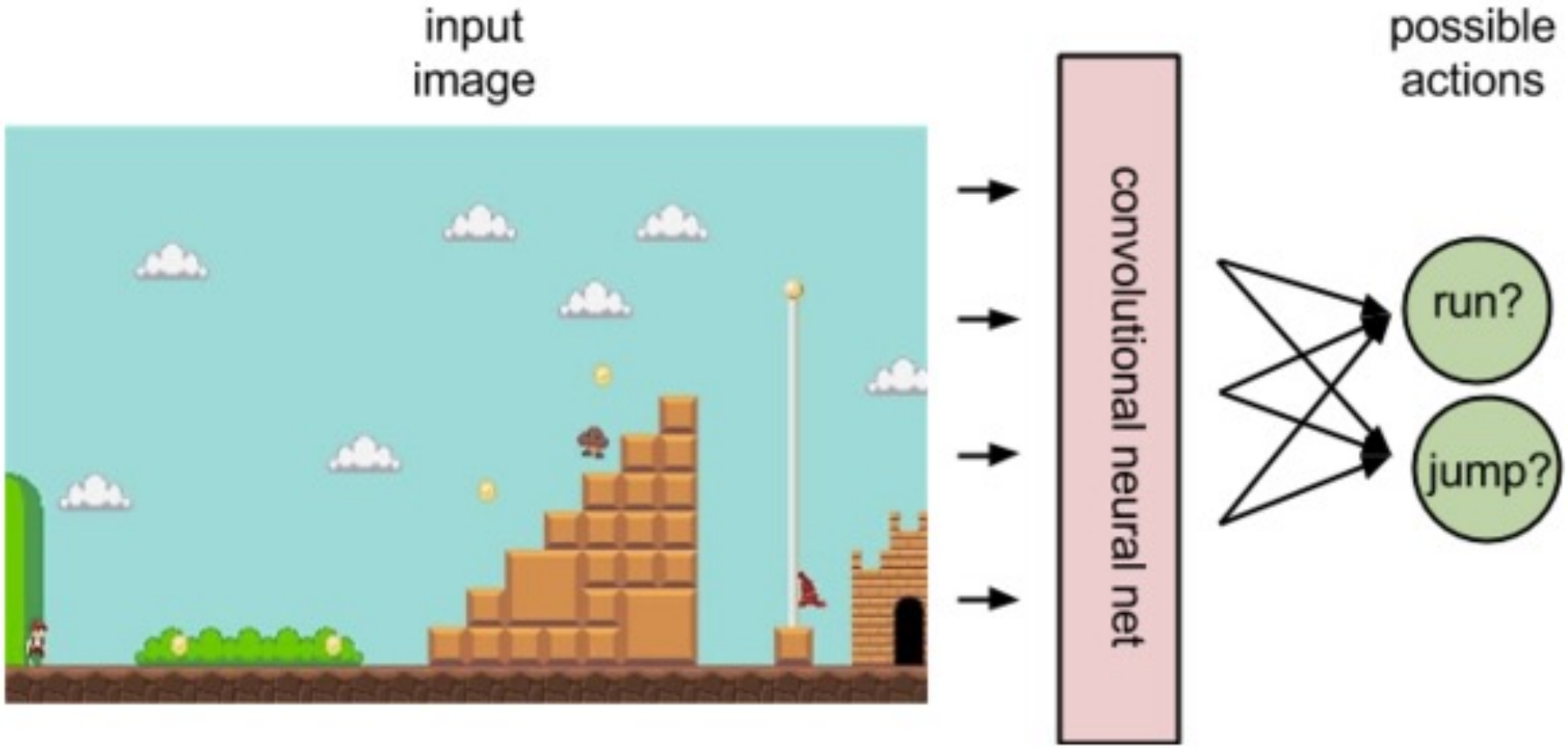
Applications of RNNs: Speech Recognition

amazon alexa

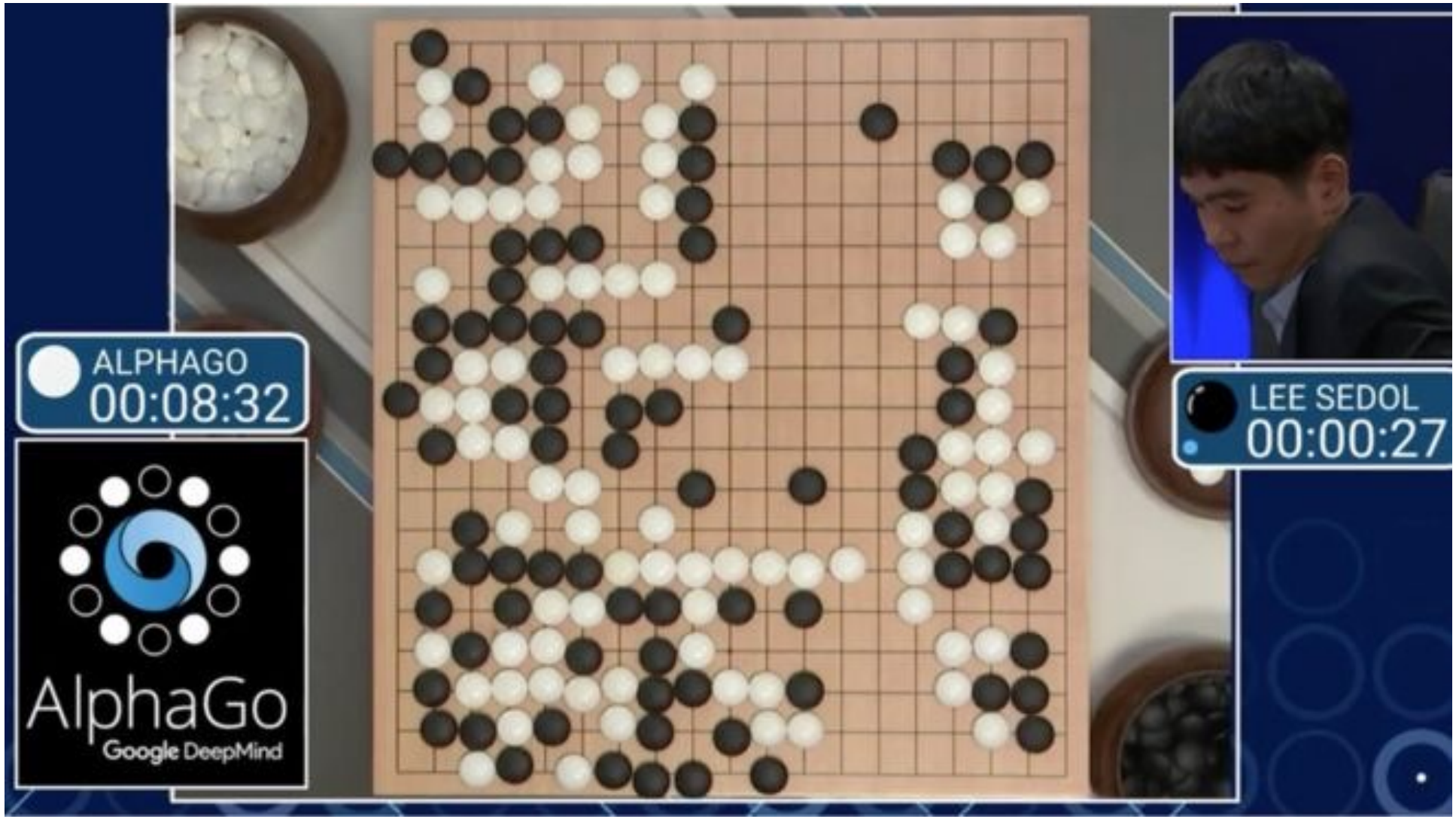
Hey Siri

Deep Reinforcement Learning (DRL)

- DRL has applications in robotics, video game, and finance.



Applications of DRL: Games



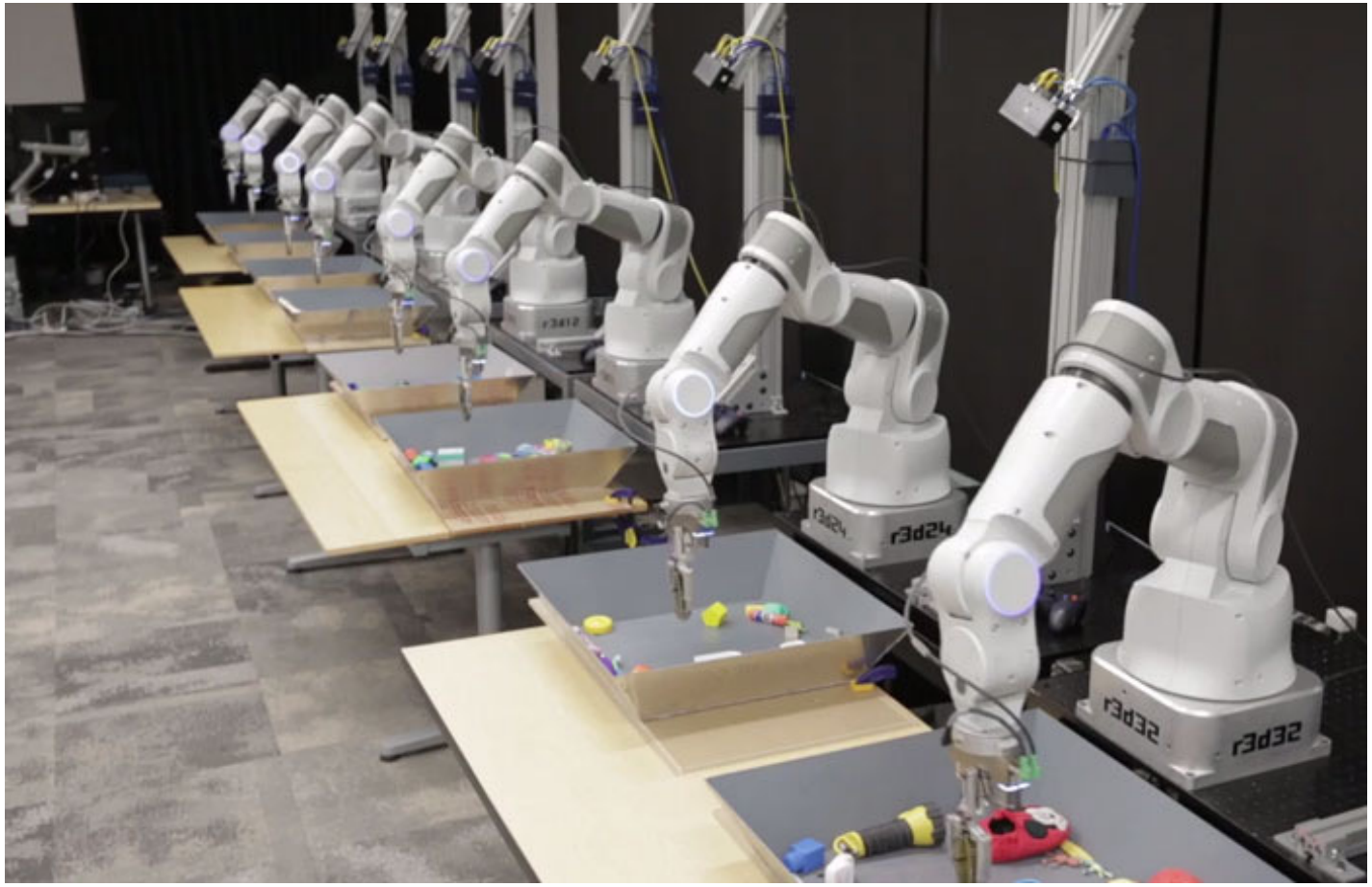
Applications of DRL: Robotics

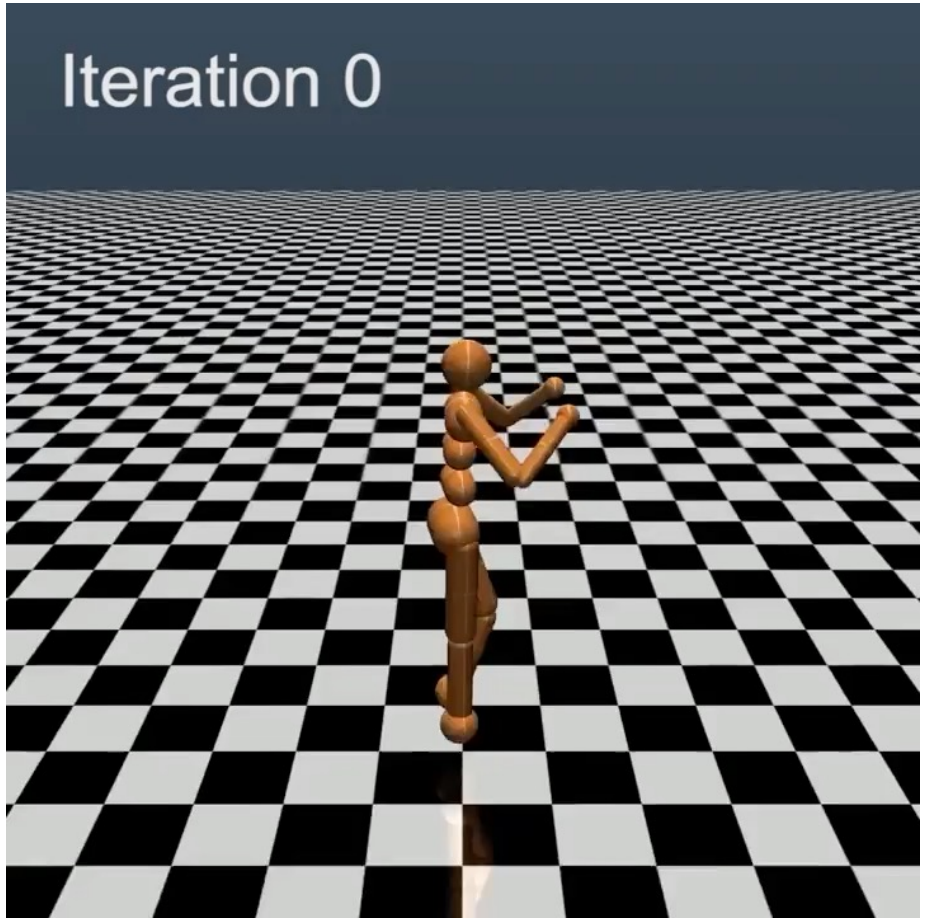
Control Theory v.s. DRL

Boston Dynamics' Atlas

Applications of DRL: Robotics

Control Theory v.s. DRL





What we have learned so far…

- **ML tasks**: regression, classification, …
- **ML models**: linear models, CNNs, RNNs, …

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{w} - y_i\right)^2$.
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.

- How to solve the model?

Computations

Computational Methods

- **ML tasks**: regression, classification, …
- **ML models**: linear models, CNNs, RNNs, …

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{w} - y_i\right)^2.$
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w}).$

- **Computations**: solve the model using numerical algorithms, e.g., gradient descent (GD) or stochastic descent (SGD).

Gradient Descent

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{w} - y_i\right)^2$.
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.

Gradient: $\frac{\partial L}{\partial \mathbf{w}}$

- $\mathbf{w}$ is a d-dimensional vector.
- $L(\mathbf{w})$ is a scalar.
- Thus $\frac{\partial L}{\partial \mathbf{w}}$ is a d-dimensional vector.

it is same as model width

Gradient Descent

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T \mathbf{w} - y_i\right)^2$.
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.

Gradient: $\frac{\partial L}{\partial \mathbf{w}}$

- $\mathbf{w}$ is a d-dimensional vector.
- $L(\mathbf{w})$ is a scalar.
- Thus $\frac{\partial L}{\partial \mathbf{w}}$ is a d-dimensional vector.

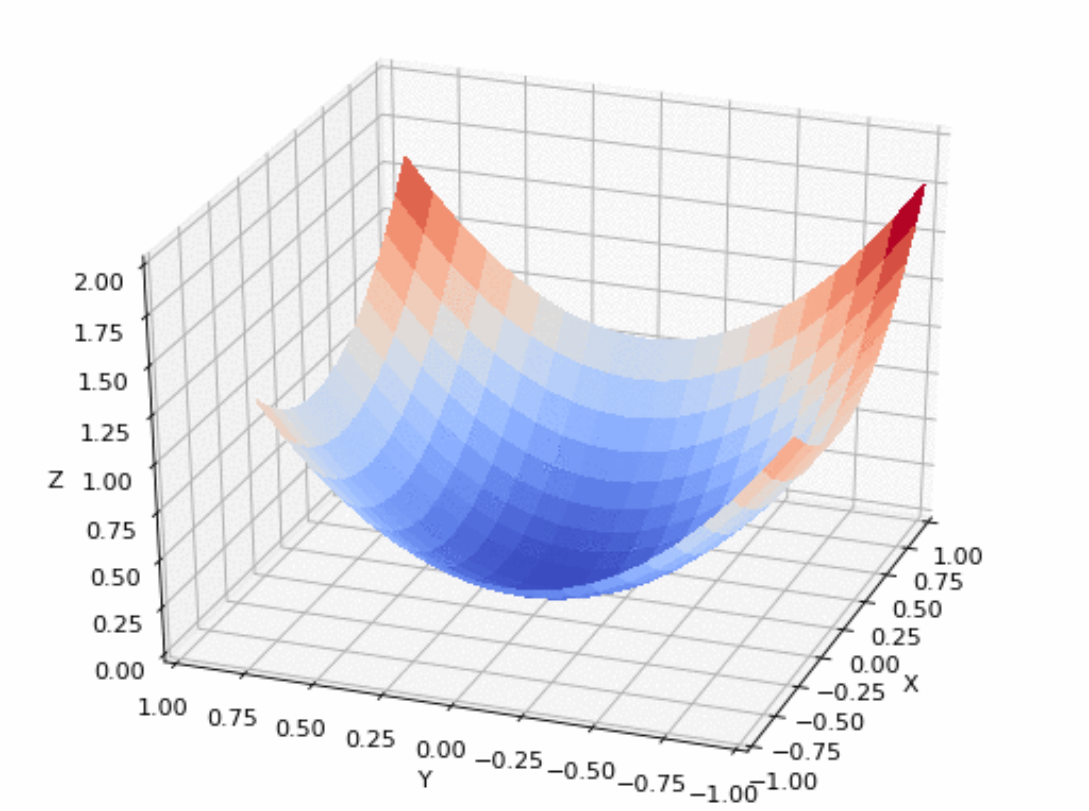
Gradient descent algorithm

- Randomly initialize $\mathbf{w}_0$.
- For $t = 0$ to T:
  - Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{w}}\Big|_{\mathbf{w}_t}$;
  - $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\, \mathbf{g}_t$.

Gradient Descent

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{w} - y_i\right)^2$.
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.



Gradient descent algorithm

- Randomly initialize $\mathbf{w}_0$.
- for each epoch
- For $t = 0$ to T:
  - Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{w}}\Big|_{\mathbf{w}_t}$;
  - $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\,\mathbf{g}_t$.

Gradient Descent

Example: least squares regression model

- Loss function: $L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{w} - y_i\right)^2$.
- Optimization: $\mathbf{w}^\star = \min_{\mathbf{w}} L(\mathbf{w})$.

Variants of Gradient Descent

- Stochastic gradient descent (SGD).
- SGD with momentum.
- RMSProp.
- ADAM…

Computational Challenges

- Big data: too many training samples.
  - ImageNet: 14 million 256×256 images.
- Big model: too many model parameters.
  - ResNet-50 (a very popular CNN architecture) has 25 million parameters.

Computational Challenges

- Big data: too many training samples.
  - ImageNet: 14 million 256×256 images.
- Big model: too many model parameters.
  - ResNet-50 (a very popular CNN architecture) has 25 million parameters.

- Big data + big model bring computational challenges.
- Training ResNet-50 on ImageNet using a single GPU takes around 14 days.

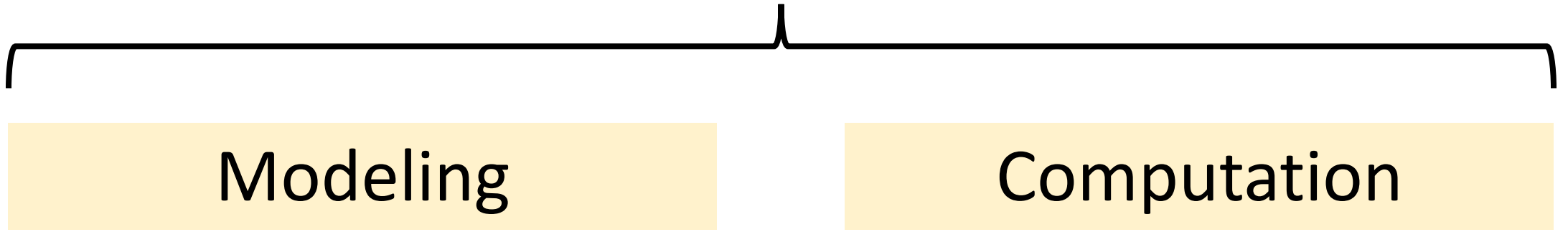
Computational Challenges

- Big data: too many training samples.
  - ImageNet: 14 million 256×256 images.
- Big model: too many model parameters.
  - ResNet-50 (a very popular CNN architecture) has 25 million parameters.

- Big data + big model bring computational challenges.
- Training ResNet-50 on ImageNet using a single GPU takes around 14 days.

➔ Efficient algorithms and software systems are necessary.

Machine Learning in Practice



Machine Learning in Practice

Modeling

Computation

- The model that fits the data and problem.
- Decide the network structures, activation functions, loss functions, etc.
- Improve the prediction accuracy.
  - Experience in ML models.
  - Understanding of the problem and data.

Machine Learning in Practice

Modeling

Computation

- Design or apply efficient algorithms.
- Implement the algorithm using systems like TensorFlow, Pytorch, etc.

Machine Learning in Practice

Modeling

Computation

- Design or apply efficient algorithms.
- Implement the algorithm using systems like TensorFlow, Pytorch, etc.
- Optimize your code.
  - Experience in algorithms.
  - Experience in systems.

Linear Regression

Warm-up: Vector and Matrix

Vector and Matrix

Vector (n-dim) $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$

Matrix ($n \times d$) $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix}$

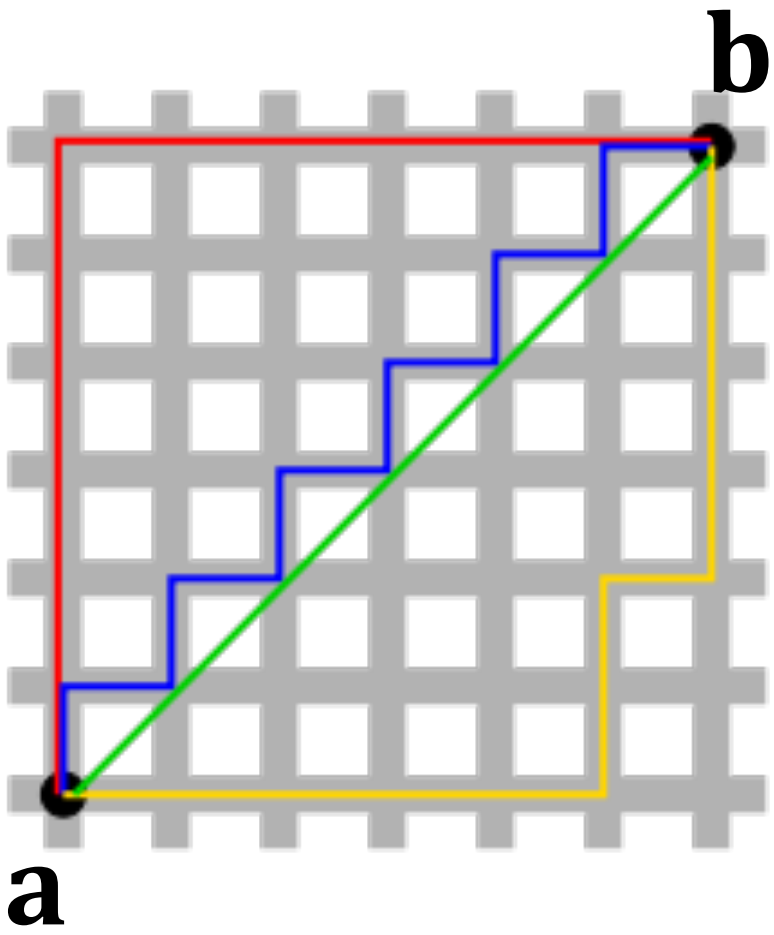
Row and columns $\mathbf{A} = \begin{bmatrix} \mathbf{a}_{:1} & \mathbf{a}_{:2} & \cdots & \mathbf{a}_{:d} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{1:} \\ \mathbf{a}_{2:} \\ \vdots \\ \mathbf{a}_{n:} \end{bmatrix}$

Vector Norms

important!

- The ℓ_p norm: $\|\mathbf{x}\|_p := \left(\sum_i |x_i|^p\right)^{1/p}$.
- The ℓ_2 norm: $\|\mathbf{x}\|_2 = \left(\sum_i x_i^2\right)^{1/2}$ (the Euclidean norm).
- The ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_i |x_i|$.
- The ℓ_∞ norm is defined by $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

Vector Norms



- The ℓ_2-distance (Euclidean distance):
  $\left|\left|\mathbf{a}-\mathbf{b}\right|\right|_2$ (green line)

- The ℓ_1-distance (Manhattan distance):
  $\left|\left|\mathbf{a}-\mathbf{b}\right|\right|_1$ (red, blue, yellow lines)

Transpose and Rank

Transpose:

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

Square matrix: a matrix with the same number of rows and columns.

Symmetric: a square matrix $\mathbf{A}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$.

Rank: the number of linearly independent rows (or columns).

Full rank: a square matrix is full rank if its rank is equal to #columns.

Eigenvalue Decomposition

- Let $\mathbf{A}$ be any $n \times n$ symmetric matrix.
- Eigenvalue decomposition: $\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$.
- Eigenvalues satisfy $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.
- Eigenvectors satisfy $\mathbf{v}_i^T \mathbf{v}_j = 0$ for all $i \neq j$.

- $\mathbf{A}$ is full rank $\longleftrightarrow$ all the eigenvalues are nonzero.

Warm-up: Optimization

Optimization: Basics

f(w) is the objective function or the loss function

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})$; s.t. $\mathbf{w} \in \mathcal{C}$.

- $\mathbf{w} = [w_1, \cdots, w_d]$: optimization variables
- $f : \mathbb{R}^d \mapsto \mathbb{R}$: objective function
- $\mathcal{C}$ (a subset of $\mathbb{R}^d$): feasible set

Optimization: Basics

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})$; s. t. $\mathbf{w} \in \mathcal{C}$.

Constraint

- $\mathbf{w} = [w_1, \cdots, w_d]$: optimization variables
- $f: \mathbb{R}^d \mapsto \mathbb{R}$: objective function
- $\mathcal{C}$ (a subset of $\mathbb{R}^d$): feasible set

Optimization: Basics

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})$; s.t. $\mathbf{w} \in \mathcal{C}$.

Optimal solution: $\mathbf{w}^\star = \operatorname{argmin}_{\mathbf{w} \in \mathcal{C}} f(\mathbf{w})$. optimal solution

- $f(\mathbf{w}^\star) \leq f(\mathbf{w})$ for all the vectors $\mathbf{w}$ in the set $\mathcal{C}$.
- $\mathbf{w}^\star$ may not exist, e.g., $\mathcal{C}$ is the empty set.
- If $\mathbf{w}^\star$ exists, it may not be unique.

Least Squares Regression

Linear Regression

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

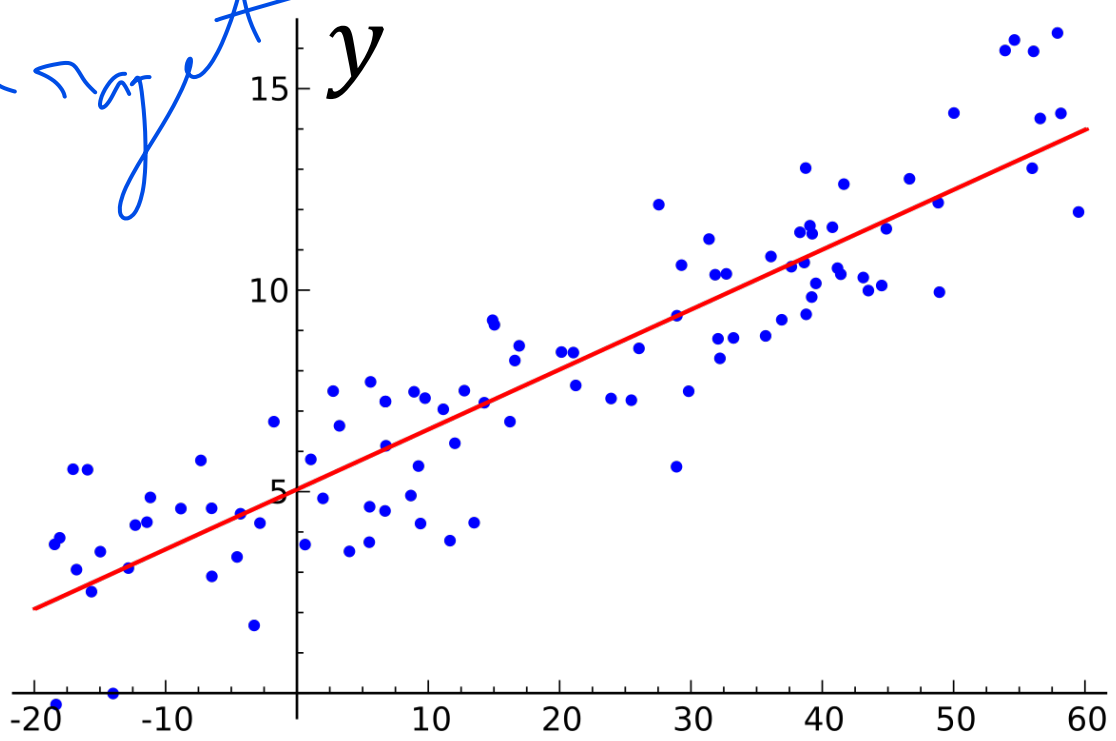
bias

1-dim ($d = 1$) example:

Solution:

$$y_i \approx 0.15\, x_i + 5.0$$

target



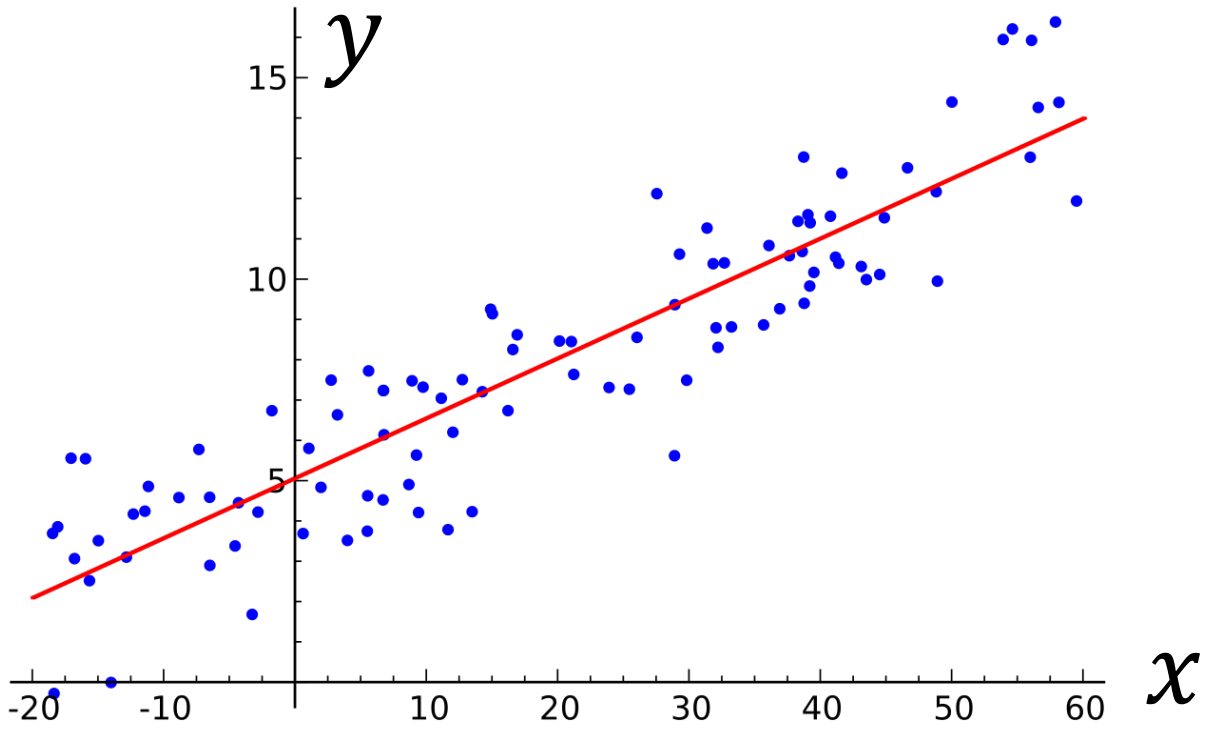
data

Linear Regression

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

Question (regard training):
how to compute $\mathbf{w}$ and b?



Least Squares Regression

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

Method: least squares regression.

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

Intercept (or bias)

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

typically data is augmented so dimension becomes d+1
so we call it x bar

$$\bar{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

the weight is also augmented with the bias and called w bar

$$\bar{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \qquad \bar{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

$$\bar{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \qquad \bar{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

$$\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} = \mathbf{x}_i^T \mathbf{w} + b$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \underbrace{\left(\mathbf{x}_i^T \mathbf{w} + b\right)}_{= \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}} - y_i)^2$$

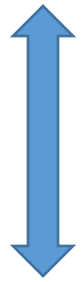
$$\bar{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \qquad \bar{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

$$\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} = \mathbf{x}_i^T \mathbf{w} + b$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \underbrace{\left(\mathbf{x}_i^T \mathbf{w} + b\right)}_{= \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}} - y_i \Big)^2$$



$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i\right)^2$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\bar{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \mathbf{x}_3^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^T & 1 \end{bmatrix}$$

augmented training data represented by each row
a total of n training data

$n \times d+1$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\bar{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \mathbf{x}_3^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^T & 1 \end{bmatrix}$$

target

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\bar{\mathbf{X}}\bar{\mathbf{w}} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} \\ \bar{\mathbf{x}}_3^T \bar{\mathbf{w}} \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} \end{bmatrix}$$

model training Data

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\bar{\mathbf{X}}\bar{\mathbf{w}} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} \\ \bar{\mathbf{x}}_3^T \bar{\mathbf{w}} \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} \end{bmatrix} \qquad \bar{\mathbf{X}}\bar{\mathbf{w}} - \mathbf{y} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \bar{\mathbf{x}}_3^T \bar{\mathbf{w}} - y_3 \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix}$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\left(l_2 \text{ norm} \right)^2$$

$$\left|\left| \bar{\mathbf{X}} \bar{\mathbf{w}} - \mathbf{y} \right|\right|_2^2 = \left|\left| \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \bar{\mathbf{x}}_3^T \bar{\mathbf{w}} - y_3 \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix} \right|\right|_2^2$$

← Square

← l_2 norm

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\left|\left| \bar{\mathbf{X}} \bar{\mathbf{w}} - \mathbf{y} \right|\right|_2^2 = \left|\left| \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \bar{\mathbf{x}}_3^T \bar{\mathbf{w}} - y_3 \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix} \right|\right|_2^2 = \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2.$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

Matrix form:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left| \bar{\mathbf{X}} \bar{\mathbf{w}} - \mathbf{y} \right|\right|_2^2$$

L_2 form

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Tasks

Linear Regression

Methods

Least Squares Regression

LASSO

Least Absolute Deviations

Algorithms

?

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Tasks	Methods	Algorithms
Linear Regression	Least Squares Regression	Analytical Solution
	LASSO	Gradient Descent (GD)
	Least Absolute Deviations	

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $\frac{\partial \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial\ \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \bar{\mathbf{X}}^T\mathbf{y})$

Algorithms

Analytical Solution

Gradient Descent (GD)

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $\dfrac{\partial \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial\ \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \bar{\mathbf{X}}^T\mathbf{y}) = \mathbf{0}$

Step 1

Solve

1st-order optimality condition

Algorithms

Analytical Solution

Gradient Descent (GD)

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $\frac{\partial \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial\ \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \bar{\mathbf{X}}^T\mathbf{y}) = \mathbf{0}$

Normal equation: $\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}}^\star = \bar{\mathbf{X}}^T\mathbf{y}$

Assume $\bar{\mathbf{X}}^T\bar{\mathbf{X}}$ is full rank.

itr means that an inverse exists

Analytical solution: $\bar{\mathbf{w}}^\star = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\,\bar{\mathbf{X}}^T\mathbf{y}$

Algorithms

- Analytical Solution
- Gradient Descent (GD)

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

iterative process

Gradient descent repeats:

1. Compute gradient: $\mathbf{g}_t = \bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}}_t - \bar{\mathbf{X}}^T\mathbf{y}$
2. Update: $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \alpha\,\mathbf{g}_t$

Algorithms

Analytical Solution

Gradient Descent (GD)

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Tasks	Methods	Algorithms
Linear Regression	Least Squares Regression	Analytical Solution
	LASSO	Gradient Descent (GD)
	Least Absolute Deviations	

Solve Least Squares in Python

1. Load Data

```
from keras.datasets import boston_housing

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

print('shape of x_train: ' + str(x_train.shape))
print('shape of x_test: ' + str(x_test.shape))
print('shape of y_train: ' + str(y_train.shape))
print('shape of y_test: ' + str(y_test.shape))
```

```
shape of x_train: (404, 13)
shape of x_test: (102, 13)
shape of y_train: (404,)
shape of y_test: (102,)
```

400 houses with 13 features for training
102 houses with 13 features for testing
404 target values for training data
102 target values for testing data

2. Add A Feature

```
import numpy

n, d = x_train.shape
xbar_train = numpy.concatenate((x_train, numpy.ones((n, 1))),
                               axis=1)

print('shape of x_train: ' + str(x_train.shape))
print('shape of xbar_train: ' + str(xbar_train.shape))
```

augmentation

```
shape of x_train: (404, 13)
shape of xbar_train: (404, 14)
```

3. Solve the Least Squares

Analytical solution: $\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T \bar{\mathbf{X}})^{-1} \bar{\mathbf{X}}^T \mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Squares

Analytical solution: $\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Squares

Analytical solution: $\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

optimal model weights or w* is saved in w

3. Solve the Least Squares

Training Mean Squared Error (MSE): $\frac{1}{n}\left|\left|\mathbf{y}-\bar{\mathbf{X}}\bar{\mathbf{w}}\right|\right|_2^2$

used for evaluation

3. Solve the Least Squares

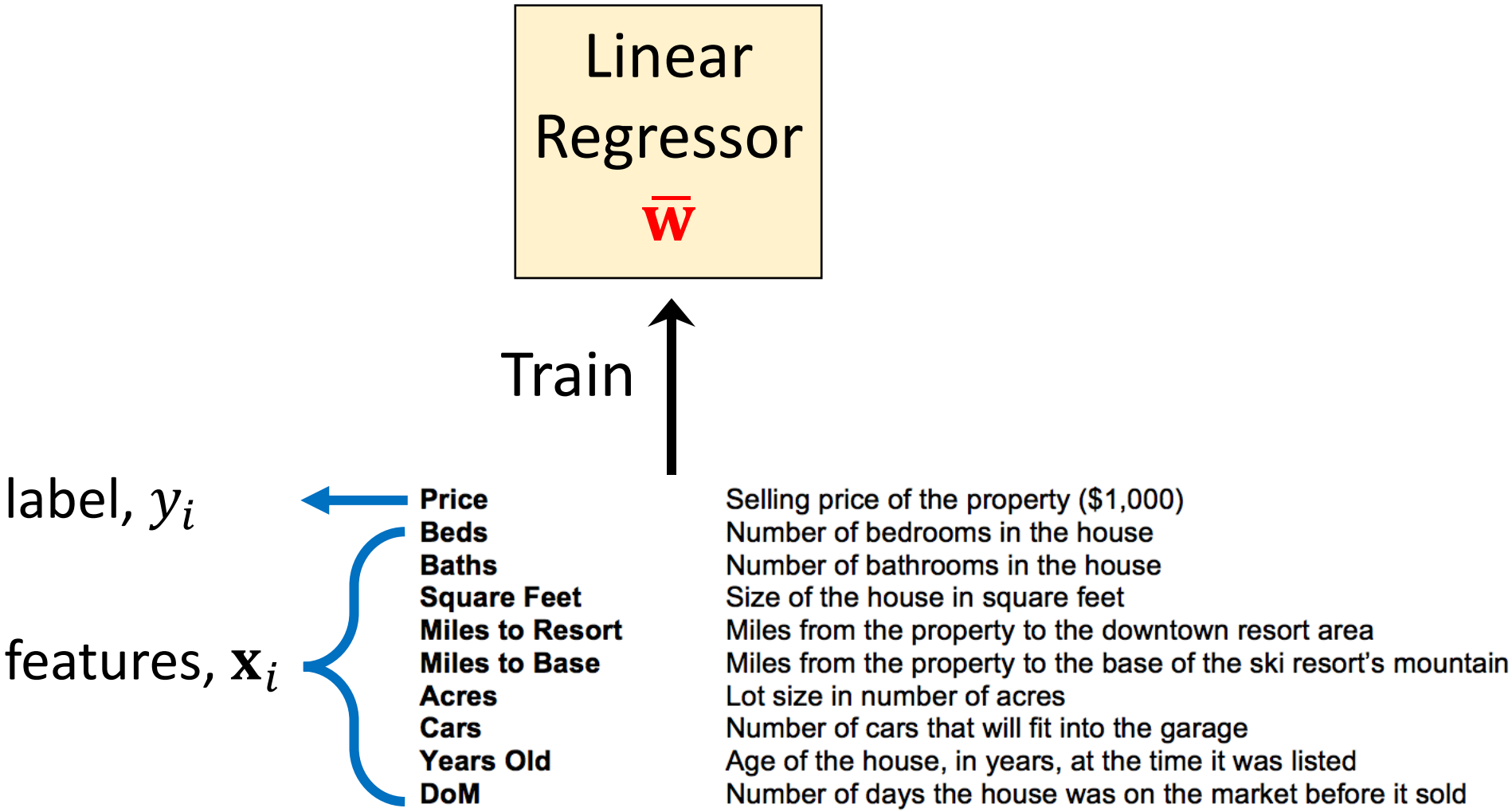
Training Mean Squared Error (MSE): $\frac{1}{n}\left|\left|\mathbf{y} - \bar{\mathbf{X}}\bar{\mathbf{w}}\right|\right|_2^2$

```
y_lsr = numpy.dot(xbar_train, w)
diff = y_lsr - y_train
mse = numpy.mean(diff * diff)
print('Train MSE: ' + str(mse))
```

y_lsr is model prediction and y_train is the actual.
so we get the diff to see the accuracy of the model

```
Train MSE: 22.004800838348 14
```

Linear Regression for Housing Price



Linear Regression for Housing Price

Features of a House, $\mathbf{x}'$

➔ Extend it to $\bar{\mathbf{x}}'$

Linear Regressor $\bar{\mathbf{w}}$

Predict

Price:
$\bar{\mathbf{w}}^T \bar{\mathbf{x}}' = \$500K$

now testing data

4. Make Prediction for Test Samples

- Add a feature to the test feature matrix: $\mathbf{X}_{\text{test}} \rightarrow \bar{\mathbf{X}}_{\text{test}}$.
- Make prediction by: $\mathbf{y}_{\text{pred}} = \bar{\mathbf{X}}_{\text{test}} \bar{\mathbf{w}}$.

augment

```
n_test, _ = x_test.shape
xbar_test = numpy.concatenate((x_test, numpy.ones((n_test, 1))), axis=1)
y_pred = numpy.dot(xbar_test, w)
```

4. Make Prediction for Test Samples

- Add a feature to the test feature matrix: $\mathbf{X}_{\text{test}} \rightarrow \bar{\mathbf{X}}_{\text{test}}$.
- Make prediction by: $\mathbf{y}_{\text{pred}} = \bar{\mathbf{X}}_{\text{test}} \bar{\mathbf{w}}$.
- MSE (test): $\frac{1}{n_{\text{test}}} \left|\left| \mathbf{y}_{\text{pred}} - \mathbf{y}_{\text{test}} \right|\right|_2^2$

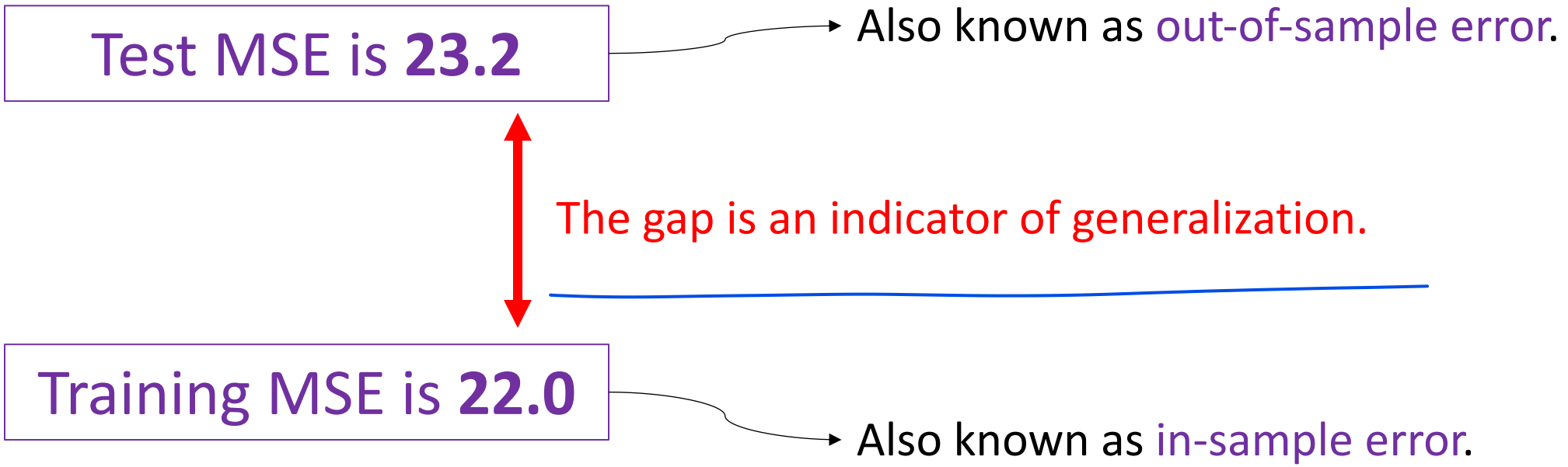
```
# mean squared error (testing)

diff = y_pred - y_test
mse = numpy.mean(diff * diff)
print('Test MSE: ' + str(mse))
```

```
Test MSE: 23.195599256409857
```

Training MSE is **22.0**

4. Make Prediction for Test Samples



small gap means model is robust and for unseen data it can predict wel!

5. Compare with Baseline

Trivial baseline:

- whatever the features are, the prediction is $\text{mean}(\mathbf{y})$.

```
y_mean = numpy.mean(y_train)

diff = y_pred - y_mean
mse = numpy.mean(diff * diff)
print('Test MSE: ' + str(mse))
```

```
Test MSE: 57.382976385300 44
```

Test MSE of least squares is **23.19**

Summary

- Linear regression problem.
- Least squares model.
- 2 algorithms for solving the model.
- Make predictions for never-seen-before test data.
- Evaluation of the model (training MSE and test MSE).
- Compare with baselines.