Lecture 04
Machine Learning I

AncoraSIR.com

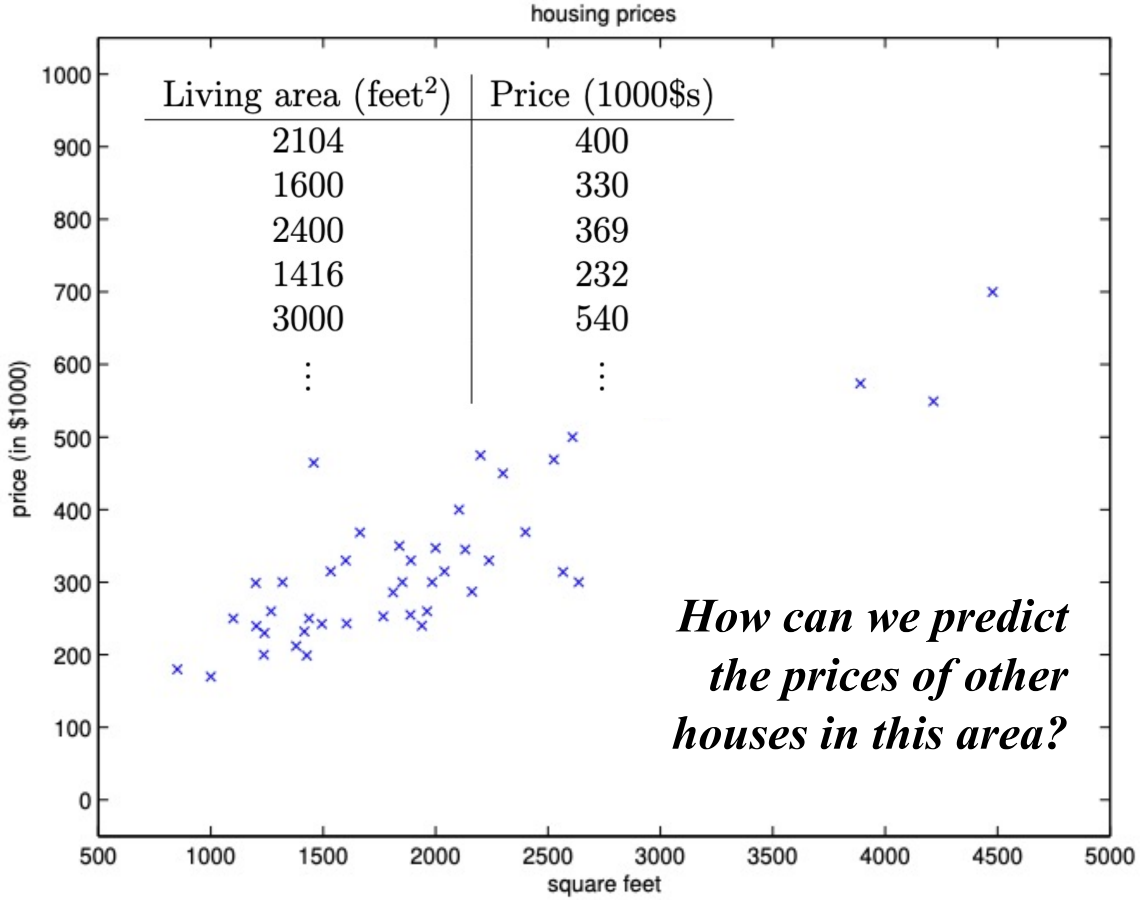
Machine Learning Basics

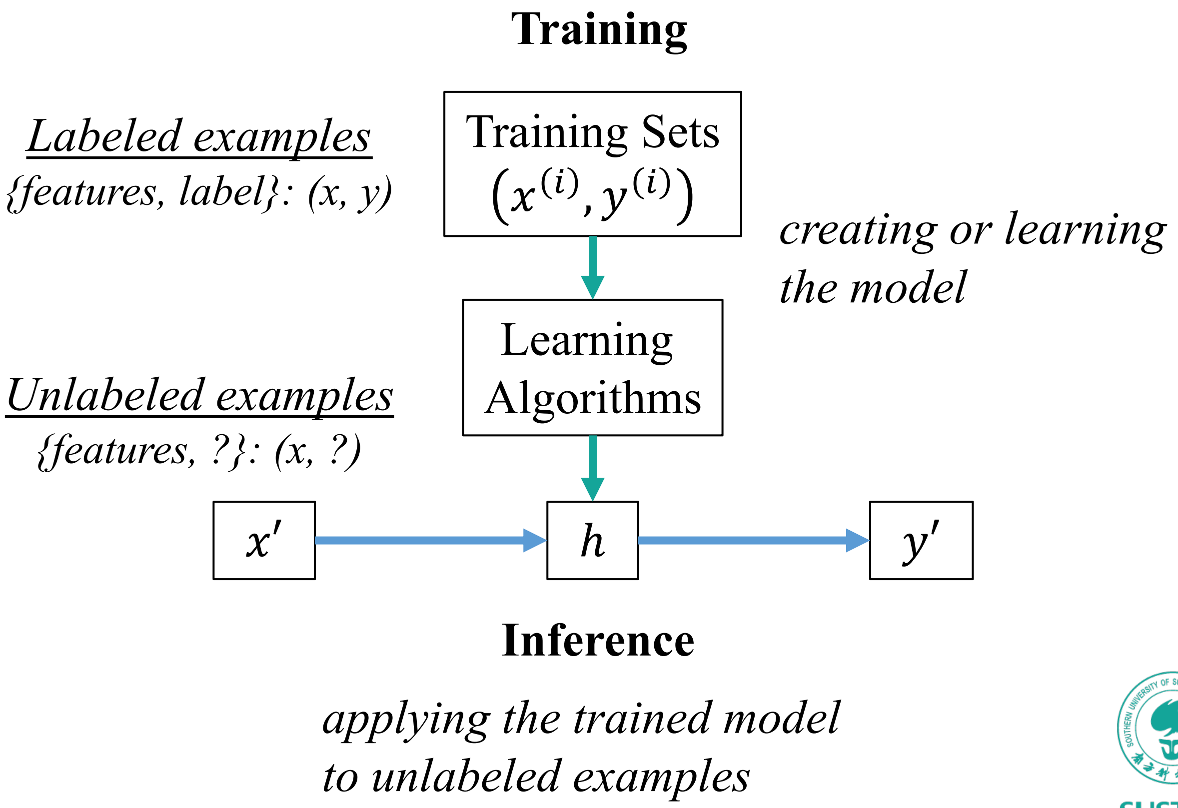
AncoraSIR.com

(Supervised) Machine Learning

The ability to teach a computer without explicitly programming it

- Design a **Model** that defines the relationship between **Features** (input x_i) and **Labels** (output y)

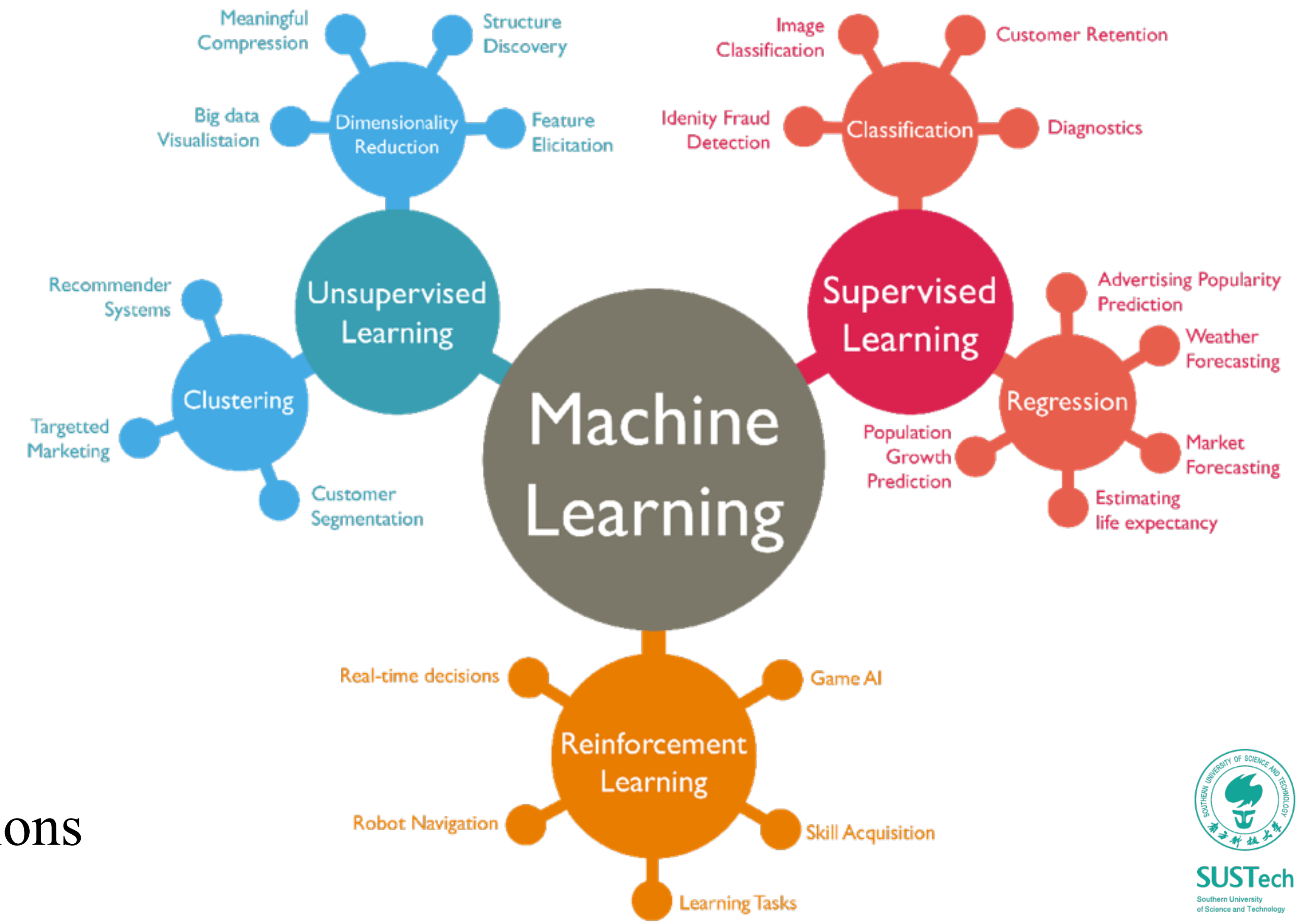




The Landscape of Machine Learning

Differences between different learning problems

- **Supervised Learning**
  - Training data is labeled
  - Goal is correctly label new data
- **Reinforcement Learning**
  - Training data is unlabeled
  - Receives feedback for its actions
  - Goal is to perform better actions
- **Unsupervised Learning**
  - Training data is unlabeled
  - Goal is to categorize the observations



AncoraSIR.com

Features in Machine Learning

The observations (input variable x_i) that are used to form predictions

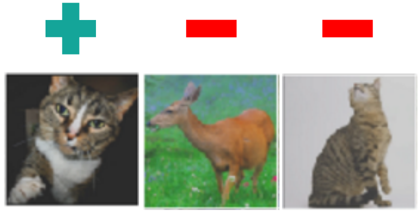
- **Image Classification**
  - Label images with appropriate categories
  - *The pixels are the features*
- **Autonomous Driving**
  - Enable cars to drive
  - *Data from the cameras, range sensors, and GPS are features*
- **Speech Recognition**
  - Convert voice snippets to text (e.g. Siri)
  - *The pitch and volume of the sound samples are the features*
- **Extracting relevant features is important for building a model**
  - *Time of day* is an irrelevant feature when classifying images
  - *Time of day* is relevant when classifying emails because SPAM often occurs at night
- **Common Types of Features in Robotics**
  - Pixels (RGB data)
  - Depth data (sonar, laser rangefinders)
  - Movement (encoder values)
  - Orientation or Acceleration (Gyroscope, Accelerometer, Compass)

Measuring Success for Classification

A confusion matrix that allows visualization of the performance of an algorithm

Prediction:

Image:



True Positive

True Negative

y

		Actual Value (as confirmed by experiment)		
		True	False	
y' **Predictive Value** (predicted by the test)	Positive	**True Positive (TP)** *Correctly identified as relevant*	**False Positive (FP)** *Incorrectly labeled as relevant* Type I Error	Precision $\frac{TP}{(TP+FP)}$
	Negative	**True Negative (TN)** *Correctly identified as not relevant* Type II Error	**False Negative (FN)** *Incorrectly labeled as not relevant*	Negative Predictive Value $\frac{TN}{(TN+FN)}$
		Sensitivity $\frac{TP}{(TP+TN)}$	Precision $\frac{FP}{(FP+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Regression uses other measurements

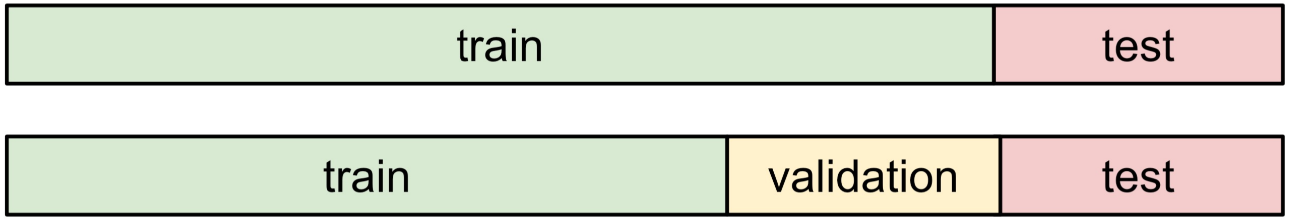
Can be used for both ***single-class*** and ***multi-class*** classification problems

AncoraSIR.com

Training and Test Data, Bias and Variance

Characteristics of Data

- **Training Data**
  - Data used to learn a model
- **Test Data**
  - Data used to assess the accuracy of model



Overfitting

- Model performs well on training data but poorly on test data

- **Bias**
  - Expected difference between model's prediction and truth
- **Variance**
  - How much the model differs among training sets

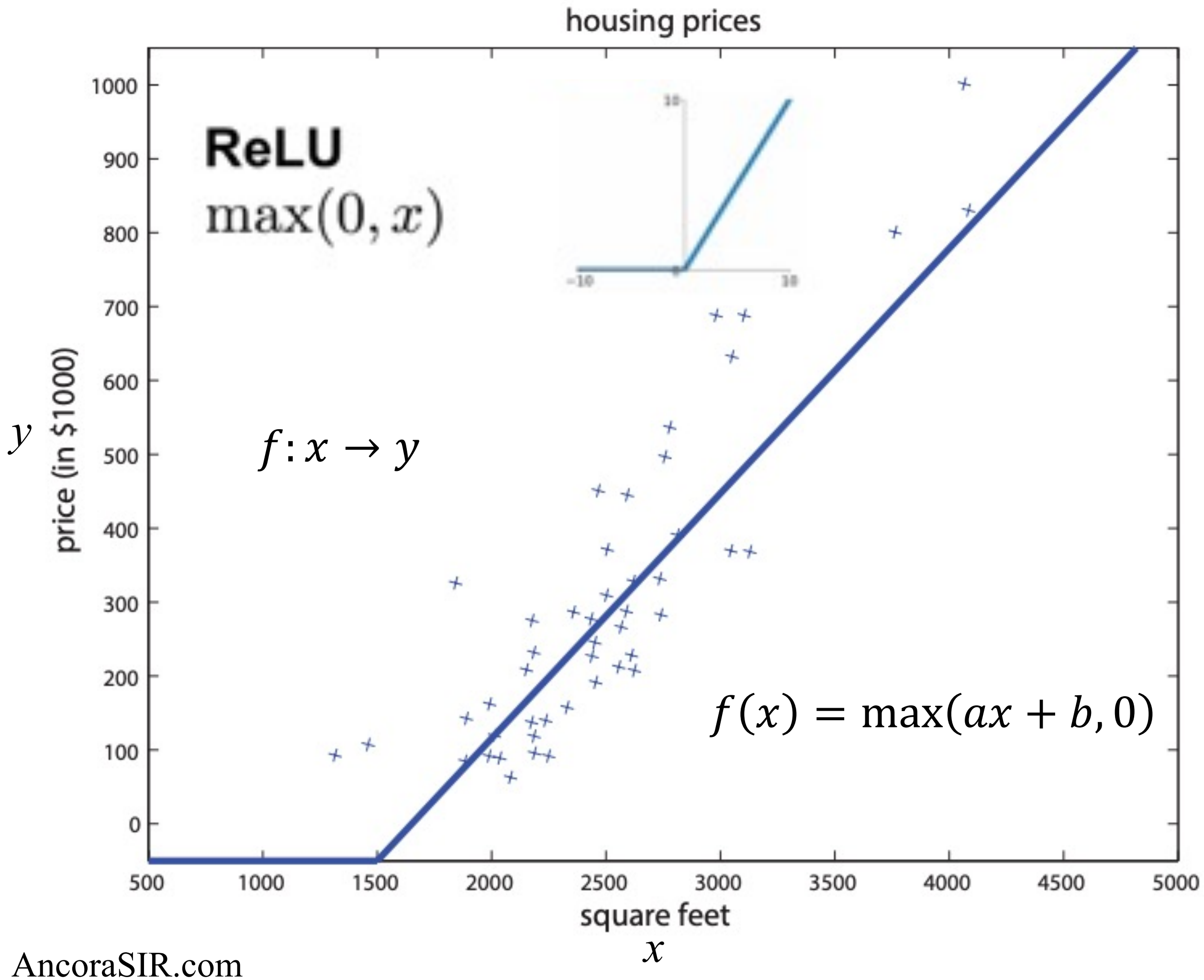
$$\begin{aligned}\text{MSE} &= \mathbb{E}[(\hat{\theta}_m - \theta)^2] \\ &= \text{Bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)\end{aligned}$$

Model Scenarios

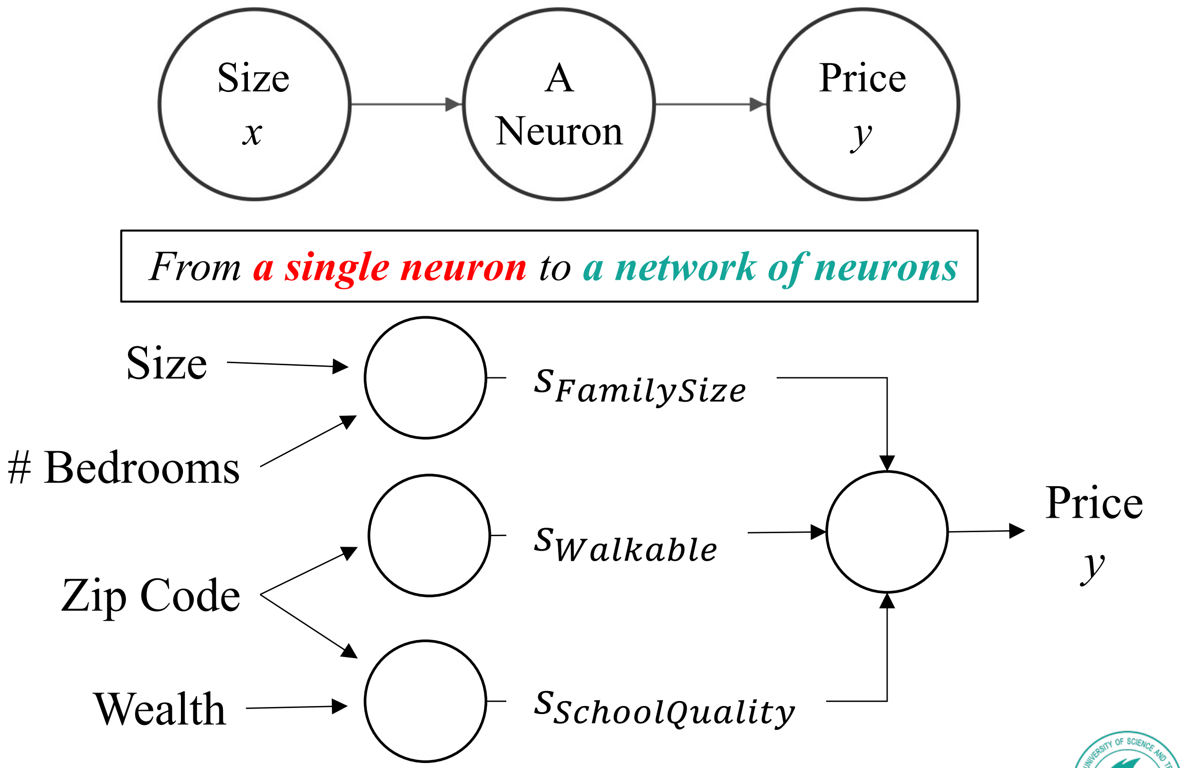
- *High Bias*: Model makes inaccurate predictions on training data
- *High Variance*: Model does not generalize to new datasets
- *Low Bias*: Model makes accurate predictions on training data
- *Low Variance*: Model generalizes to new datasets

A Further Look into Housing Price Prediction

Building a neural network with Weighted-Sum Scores



$y = f_{WeightedSum}(x) = wx + b$



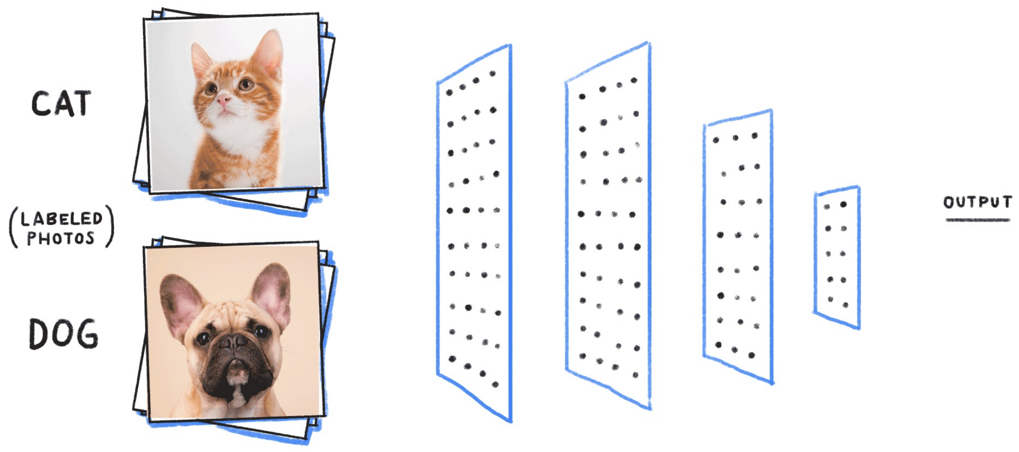
$\boldsymbol{s} = f_{WeightedSum1}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$

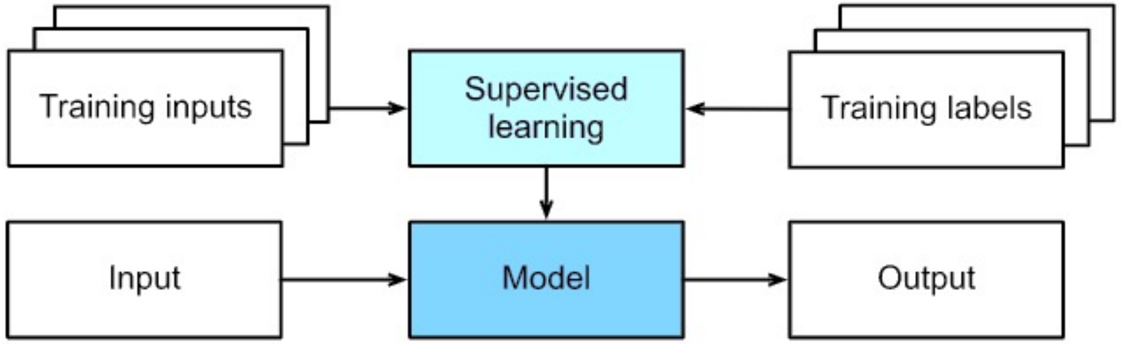
$y = f_{WeightedSum2}(\boldsymbol{s})$

AncoraSIR.com

Supervised Learning with Neural Networks

Structured Data vs. Unstructured Data

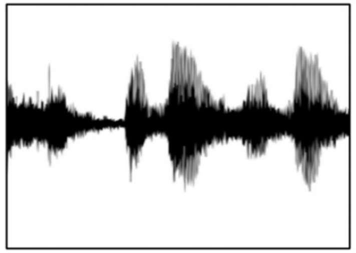




Structured Data

Size	#bedrooms	···	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

Unstructured Data



Audio

Image

Four scores and seven years ago…

Text

AncoraSIR.com

A Roadmap of Supervised Machine Learning

$$\hat{y} = g_{Activation}\left[f_{WeightedSum}(\mathbf{x})\right]$$

- Linear Regression
  - (Argurably) the simplest ML model
  - Basic concepts applicable to all ML problems
  - $\hat{y} = f_{WeightedSum}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
- Logistic Regression
  - Binary LC using sigmoid activation
  - Binary output with a probability
  - $\hat{y} = g_{Activation}(s) = \text{sigmoid}(s)$
- Softmax Regressions
  - Multi-class LC using softmax activation
  - Multi-class output with a probability distribution
  - $\mathbf{s} = f_{WeightedSum}(\mathbf{x}) = \mathbf{Wx} + \mathbf{b}$
  - $\hat{\mathbf{y}} = g_{Activation}(\mathbf{s}) = \text{softmax}(\mathbf{s})$
- Linear Classification
  - Vectorized weights for multiple classes
  - $\mathbf{s} = f_{WeightedSum}(\mathbf{x}) = \mathbf{Wx} + \mathbf{b}$
  - $\hat{\mathbf{y}} = g_{Activation}(\mathbf{s}) = ?$
- Single-neuron Perceptron
  - Binary LC using step activation
  - $s = f_{WeightedSum}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
  - $\hat{y} = g_{Activation}(s) = \text{step}(s, 0)$
- Multi-layer Perceptron
  - Neural network featuring hidden units
  - $\hat{\mathbf{y}}_N = g_{A_N}\left[f_{W_N}(\hat{\mathbf{y}}_{N-1})\right] \dots \hat{\mathbf{y}}_1 = g_{A_1}\left[f_{W_1}(\mathbf{x})\right]$

Regression vs. Classification

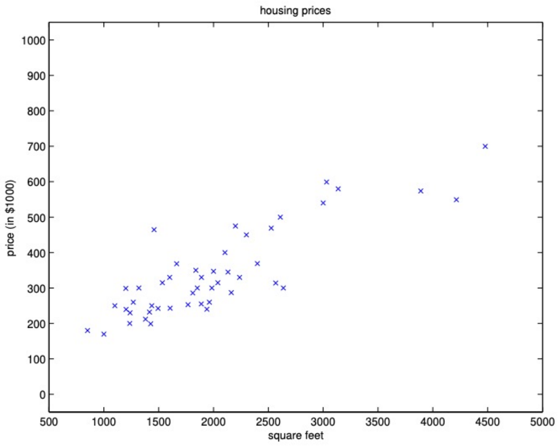
仿生设计
与学习
实验室
Bionic Design
& Learning Lab

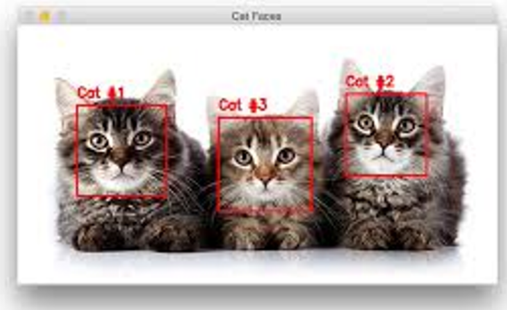
AncoraSIR.com

Classification vs. Regression

Continuous or Discrete Values

- Design a **Model** that defines the relationship between **Features** (input x_i) and **Labels** (output y)

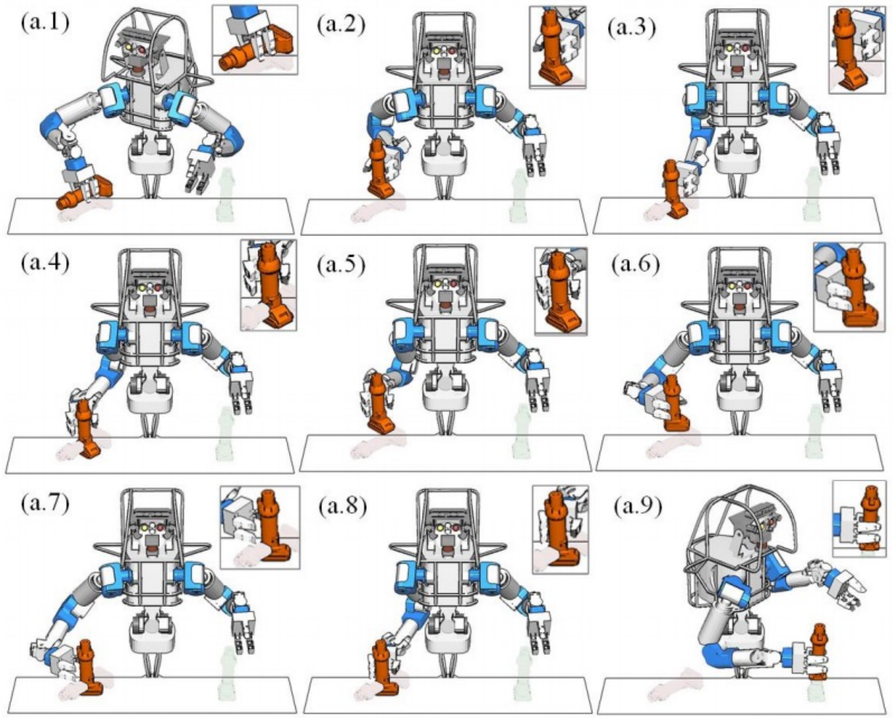


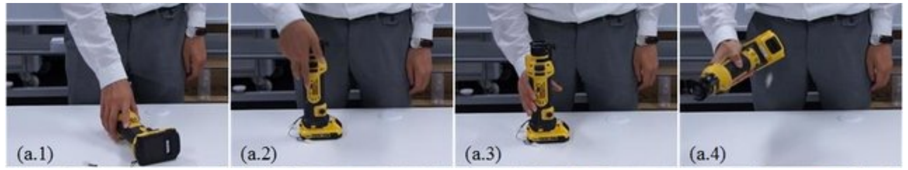


AncoraSIR.com

- **Regression**: usually predicts continuous values.
    - *What is the value of a house in Shenzhen?*
    - *What is the probability that a user will click on this ad?*
- **Classification**: usually predicts discrete values.
    - *Is a given email spam or not spam?*
    - *Is this an image of a dog, a cat, or a hamster?*
- **Regression as classification**
    - *Scores higher than 60 gets a pass?*
    - *What's the probability of getting a pass?*
    - *How likely the robot's motion is similar to the human's motion?*

https://arxiv.org/pdf/1812.03274.pdf





Linear Regression

Arguably the simplest and most popular among the standard tools

- Linear Regression Assumption
    1. The relationship between the feature x and target y is linear

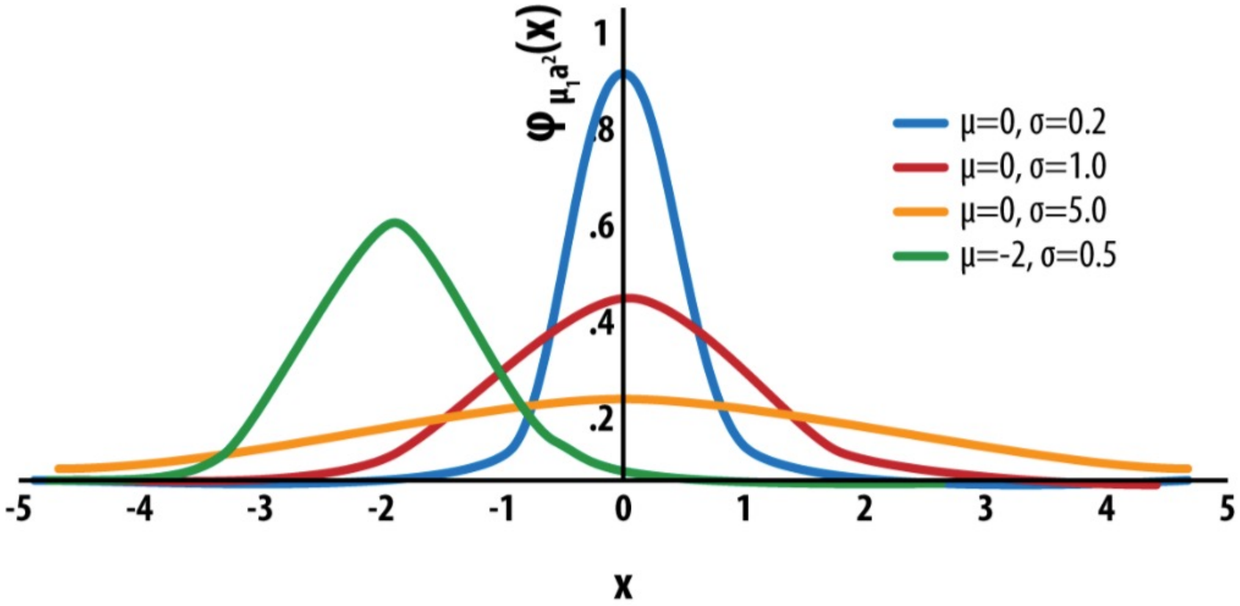
$$y = wx + b$$

learnable parameters that must be estimated from data

    2. Any noise is well-balanced, i.e. follows a Gaussian distribution

$$y = wx + b + N(0, \epsilon)$$

standard deviation of the noise term



$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(z-\mu)^2\right)$$

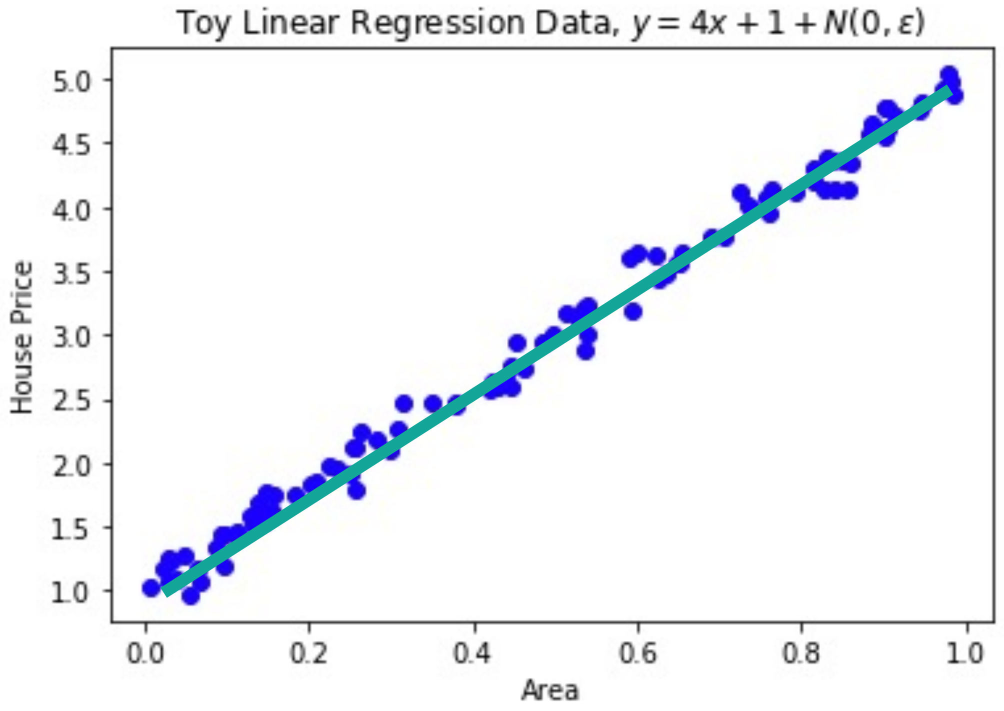
AncoraSIR.com

An Example with a Toy Dataset

Linear Regression

```python
import numpy as np
# Generate synthetic data
N=100
w_true = 4
b_true = 1
noise_scale = .1
x_np = np.random.rand(N, 1)
# Convert shape of y_np to (N,)
noise = np.random.normal(scale=noise_scale, size=(N, 1))
y_np = np.reshape(w_true * x_np + b_true + noise, (-1))
```

```python
import matplotlib.pyplot as plt
plt.plot(x_np, y_np, 'bo')
plt.xlabel('Area')
plt.ylabel('House Price')
plt.title('Toy Linear Regression Data, $y=4x+1+N(0, \epsilon)$')
plt.show()
```



Common examples include

- predicting prices (of homes, stocks, etc.),
- predicting length of stay (for patients in the hospital),
- demand forecasting (for retail sales)

- **Training data**: the toy dataset
- **An instance**: a set of *x* & *y*
- **Target/Label**: the house price
- **Feature/Covariate**: house area

AncoraSIR.com

Linear Model

The goal of linear regression

- **w**
  - The ***weight*** determines the influence of each feature on our prediction, usually a vector form with w_i
- b
  - The ***bias*** says what value the predicted price should take when all features take 0
- Given a dataset, **our goal** is
  - To choose the weights **w** and bias b such that on average, the predictions made based on our model best fit the true prices observed in the data.

$$\hat{y} = w_1 \cdot x_1 + \ldots + w_d \cdot x_d + b \longrightarrow \hat{y} = \mathbf{w}^T \mathbf{x} + b.$$

$$\hat{y}^i = w_1 x_1^i + w_2 x_2^i + \cdots + w_d x_d^i + b$$

index label data point

i y^i [x_1^i x_2^i $x_{\ldots}^i$ x_d^i]

City	Number of weekly riders	Price per week ($)	Population of city	Monthly income of riders ($)	Average parking rates per month ($)
1	192000	15	1800000	5800	50
2	190400	15	1790000	6200	50
3	191200	15	1780000	6400	60
4	177600	25	1778000	6500	60
5	176800	25	1750000	6550	60
6	178400	25	1740000	6580	70
7	180800	25	1725000	8200	75
8	175200	30	1725000	8600	75
9	174400	30	1720000	8800	75
10	173920	30	1705000	9200	80
11	172800	30	1710000	9630	80
12	163200	40	1700000	10570	80
13	161600	40	1695000	11330	85
14	161600	40	1695000	11600	100
15	160800	40	1690000	11800	105
16	159200	40	1630000	11830	105
17	148800	65	1640000	12650	105
18	115696	102	1635000	13000	110
19	147200	75	1630000	13224	125
20	150400	75	1620000	13766	130
21	152000	75	1615000	14010	150
22	136000	80	1605000	14468	155
23	126240	86	1590000	15000	165
24	123888	98	1595000	15200	175
25	126080	87	1590000	15600	175
26	151680	77	1600000	16000	190
27	152800	63	1610000	16200	200

AncoraSIR.com

Vectorization of a Linear Model

The goal of linear regression

$$\hat{y} = w_1 \cdot x_1 + \ldots + w_d \cdot x_d + b \;\rightarrow\; \hat{y} = \mathbf{w}^T\mathbf{x} + b \;\rightarrow\; \hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$$

- Vectorization
  - All features into a vector **x** for a single data point
  - All weights into a vector **w**
  - Our entire dataset as the *design matrix* **X,** including one row for every example and one column for every feature

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(i)} & \cdots & x_d^{(i)} \end{bmatrix}$$

one row for every example

one column for every feature

$$\hat{y}^i = w_1 x_1^i + w_2 x_2^i + \cdots + w_d x_d^i + b$$

index label data point

i y^i $[\, x_1^i \quad x_2^i \quad x_{\ldots}^i \quad x_d^i \,]$

City	Number of weekly riders	Price per week ($)	Population of city	Monthly income of riders ($)	Average parking rates per month ($)
1	192000	15	1800000	5800	50
2	190400	15	1790000	6200	50
3	191200	15	1780000	6400	60
4	177600	25	1778000	6500	60
5	176800	25	1750000	6550	60
6	178400	25	1740000	6580	70
7	180800	25	1725000	8200	75
8	175200	30	1725000	8600	75
9	174400	30	1720000	8800	75
10	173920	30	1705000	9200	80
11	172800	30	1710000	9630	80
12	163200	40	1700000	10570	80
13	161600	40	1695000	11330	85
14	161600	40	1695000	11600	100
15	160800	40	1690000	11800	105
16	159200	40	1630000	11830	105
17	148800	65	1640000	12650	105
18	115696	102	1635000	13000	110
19	147200	75	1630000	13224	125
20	150400	75	1620000	13766	130
21	152000	75	1615000	14010	150
22	136000	80	1605000	14468	155
23	126240	86	1590000	15000	165
24	123888	98	1595000	15200	175
25	126080	87	1590000	15600	175
26	151680	77	1600000	16000	190
27	152800	63	1610000	16200	200

AncoraSIR.com

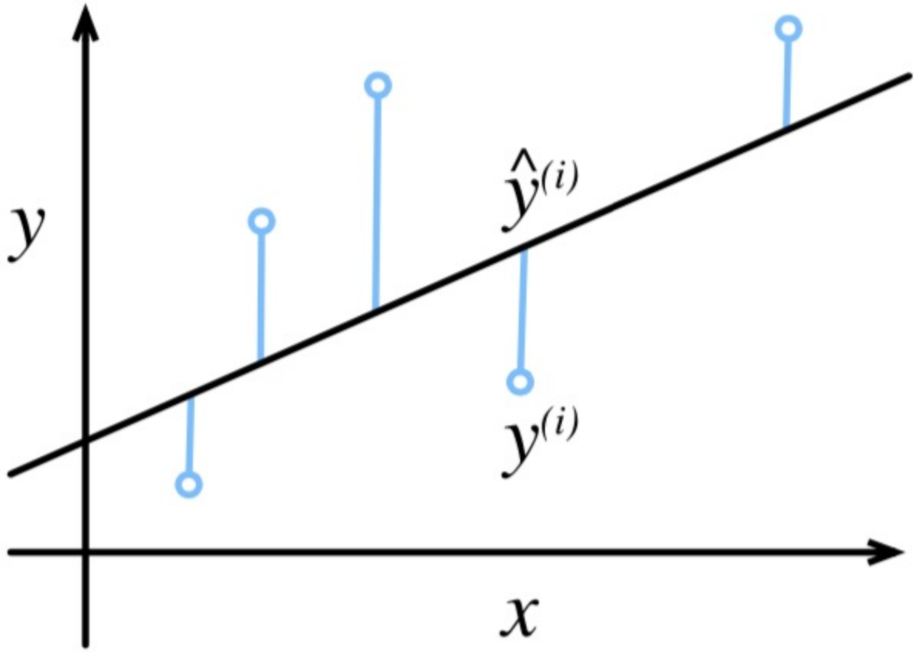
Loss Function

A quality measure for some given model

- To quantify the distance between the ***predicted*** and ***real*** value of the target.
  - usually be a non-negative number where smaller values are better
  - perfect predictions incur a loss of 0
- The Sum of Squared Errors $l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$
  - the empirical error is only
    a function of the model parameters
- Loss Function as an averaged SSE

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2$$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\ L(\mathbf{w}, b)$$

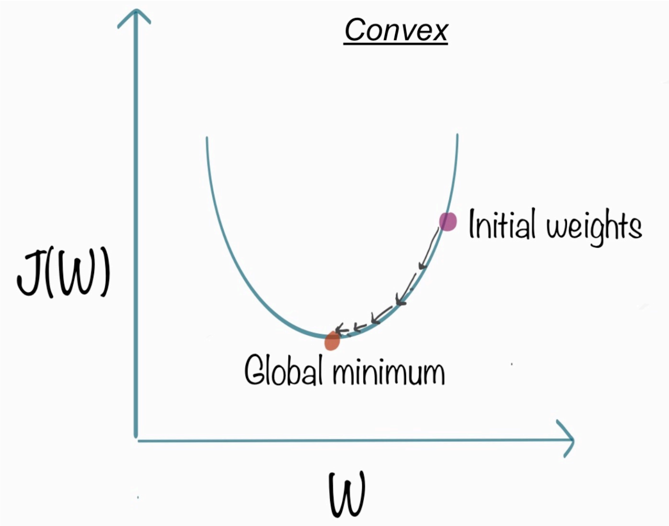


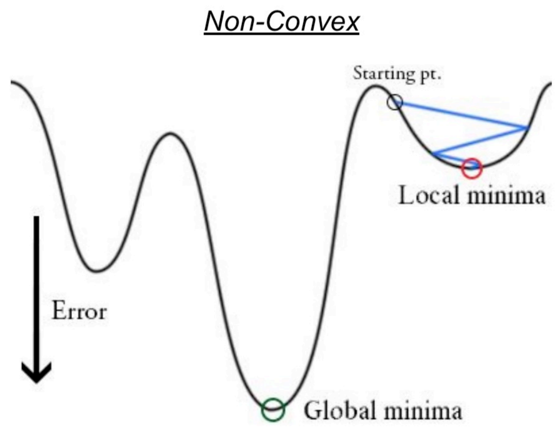
AncoraSIR.com

Gradient Descent

A procedure for updating the model parameters to improve its quality

- **Iteratively reducing** the error by updating the parameters in the direction that incrementally lowers the loss function, or Gradient Descent
  - On *convex* loss surfaces, it will eventually converge to a global minimum
  - For *nonconvex* surfaces, it will at least lead towards a (hopefully good) local minimum.





- The key technique for optimizing *nearly any* deep learning model

AncoraSIR.com

Stochastic Gradient Descent

a more efficient practice

- Sampling a random minibatch of examples every time we need to computer the update
  - Initialize model parameters at random;
  - Iteratively sample random batches to update the parameters in the direction of the negative gradient

learning rate

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b)$$

batch size: the number of examples in each minibatch

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = w - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right),$$

$$b \leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) = b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right).$$

- Hyperparameters
  - The values of the batch size and learning rate are manually pre-specified and not typically learned through model training.
  - Tunable but not updated in the training loop.

AncoraSIR.com

Maximum Likelihood Estimation

Assume that observations arise from normally distributed noisy observations

- The best values of *b* and *w* are those that maximize the likelihood of the entire dataset

$$y = \mathbf{w}^\top \mathbf{x} + b + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$P(Y \mid X) = \prod_{i=1}^{n} p(y^{(i)}|\mathbf{x}^{(i)})$$

- Maximizing the product of many exponential functions is *difficult*

- The likelihood of seeing a particular *y* for a given *x*

$$p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top \mathbf{x} - b)^2\right)$$

$$-\log p(\mathbf{y}|\mathbf{X}) = \sum_{i=1}^{n} \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} - b\right)^2$$

Negative Log-Likelihood (NLL)

If a constant

SSE

*Why **minimizing squared error** is equivalent to **maximum likelihood estimation** of a linear model under the assumption of additive Gaussian noise?*

AncoraSIR.com

Linear Classification

How to scientifically calculate a decision

- Hypothesis
  - Acceptance depending on Test and Grade
- Data
  - i sets of example data $\left(x^{(i)}, y^{(i)}\right)$
- Input
  - $x_1^{(i)}$ as test scores and $x_2^{(i)}$ as test scores
- Output
  - $\hat{y}^{(i)}$ as a threshold decision of Accept or Reject
- Model
  - A linear boundary line to separate the data
    - $w_1 x_1 + w_2 x_2 + b = 0$
  - A threshold to activate a decision against the line
    - > 0: Accept; < 0: Reject
- Learning
  - Obtain a set of w_i and b with small enough $y^{(i)} - \hat{y}^{(i)}$

<u>*An example of acceptance at a University*</u>

A Linear Boundary Line of $2x_1 + x_2 - 18 = 0$ as a decision criteria from regression to classification

AncoraSIR.com

An Example of Linear Classification with Images

A data-driven approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

A General Problem Statement

- Given
  - A **score function** that maps the raw data to class scores,
  - A **loss function** that quantities the agreement between the predicted scores and the ground truth labels.
- Goal
  - As an **Optimization Problem** in which we will minimize the loss function with respect to the parameters of the score function.

AncoraSIR.com

An Example of Linear Classifier for Images

A data-driven approach for linear classification

- Data
  - i sets of labelled image data $\left(x^{(i)}, y^{(i)}\right)$
- Hypothesis
  - Image features provides the data for classification
- Input
  - $x^{(i)}$ of image pixels,
  - i.e., arrays of 32x32x3 numbers
- Output
  - $\hat{y}^{(i)}$ as predicted classification of the image
  - i.e., a 10x1 vector with scores for each entry
- Model
  - A score function of weighted-sum
    - $f(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}\,\boldsymbol{x} + b$
- Learning
  - An optimization algorithm that updates the the weight **W** (10x3072) and bias *b* (10x1) by minimizing a loss function

3072x1

$f(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}\,\boldsymbol{x} + b$ 10x1

10x1 10x3072

Array of **32x32x3** numbers (3072 numbers total)

→ $f(\boldsymbol{x}, \mathbf{W})$ → **10** numbers giving class scores

↑

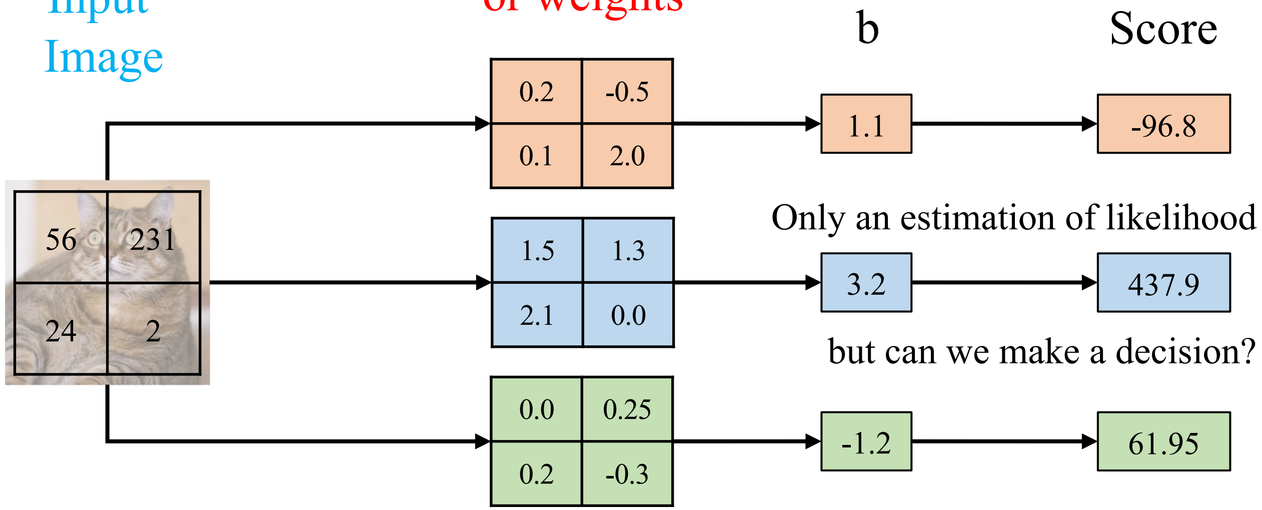
W

Parameters or weights

A (linear) score function

- *How can we tell whether this W is good or bad?*

Input Image



Only an estimation of likelihood

but can we make a decision?

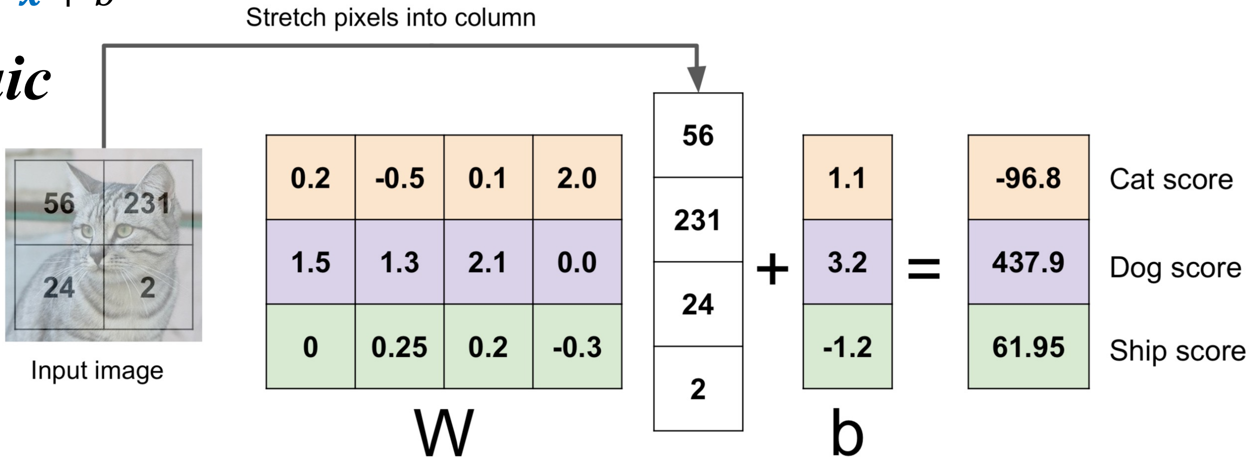
AncoraSIR.com

Three Viewpoints of Image Classification

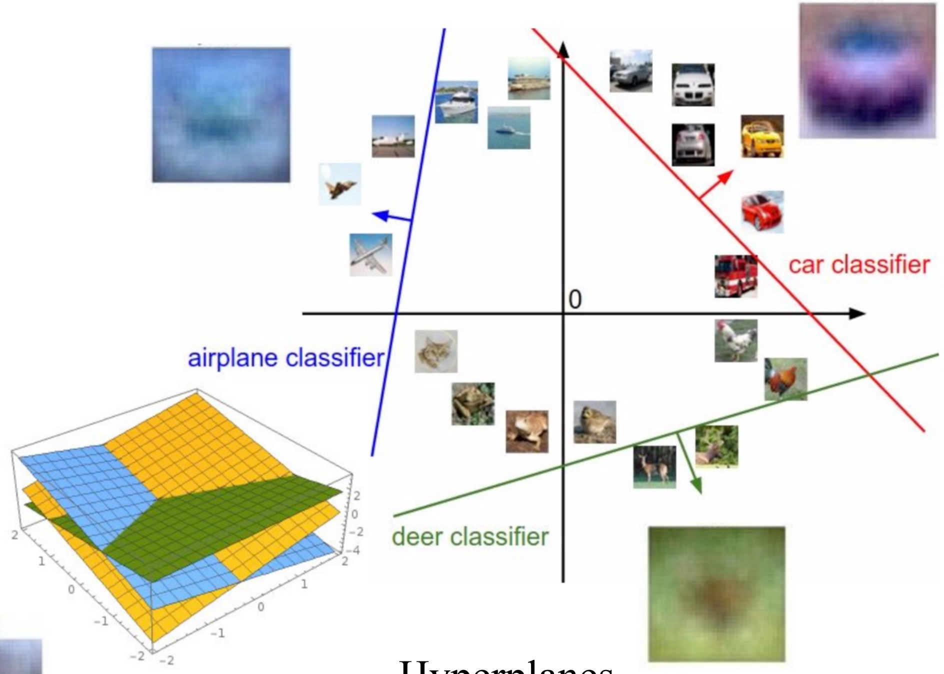
Strategies for making a decision based on weighted sum of the image features

$f(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}\,\boldsymbol{x} + b$

Algebraic



Geometric



Hyperplanes cutting up space

Visual

One template per class

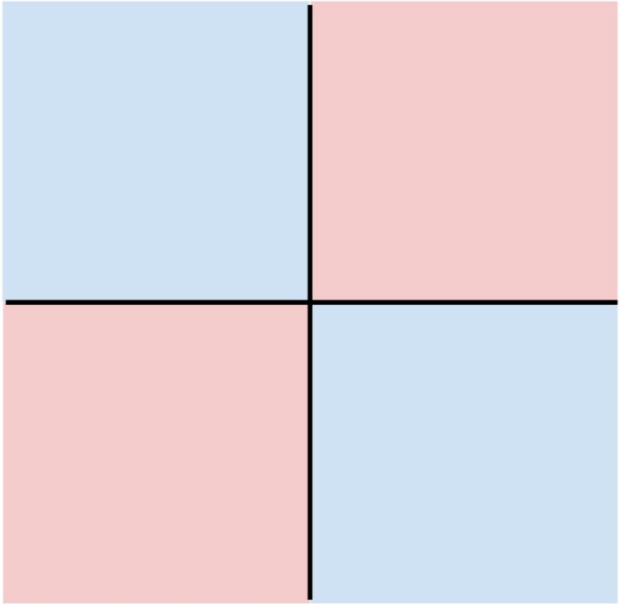
AncoraSIR.com

Hard Cases for a Linear Classifier

Simple linear classifiers are not enough to make a complex decision

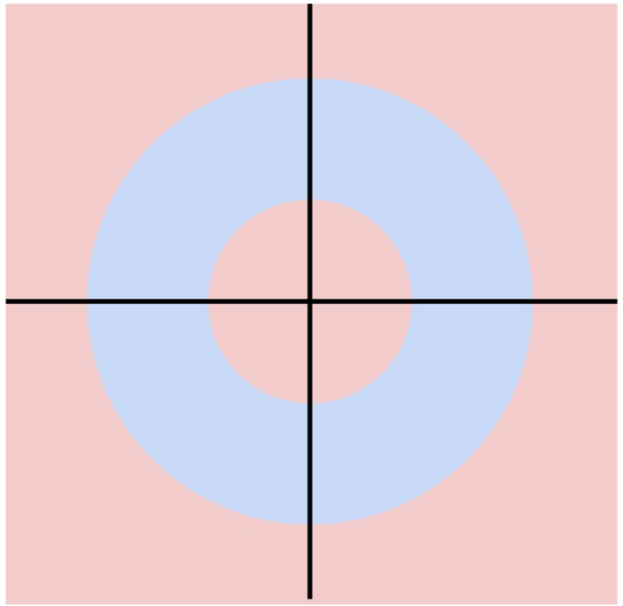
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



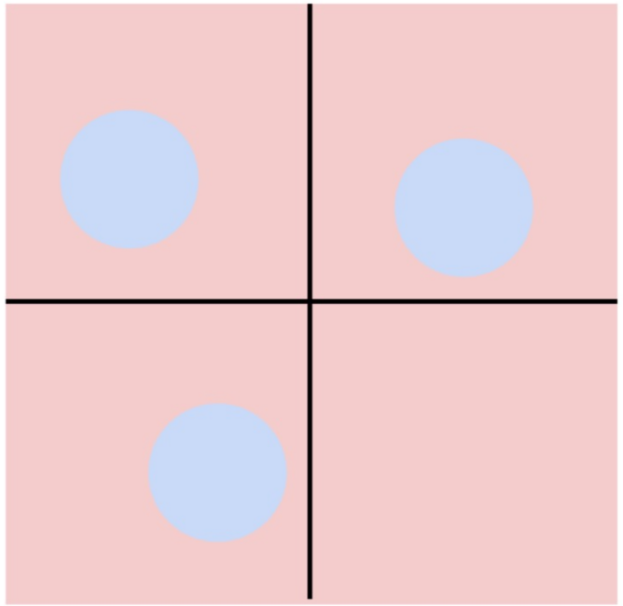
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



AncoraSIR.com

Thank you~

songcy@sustech.edu.cn