上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 9: Linear Models

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

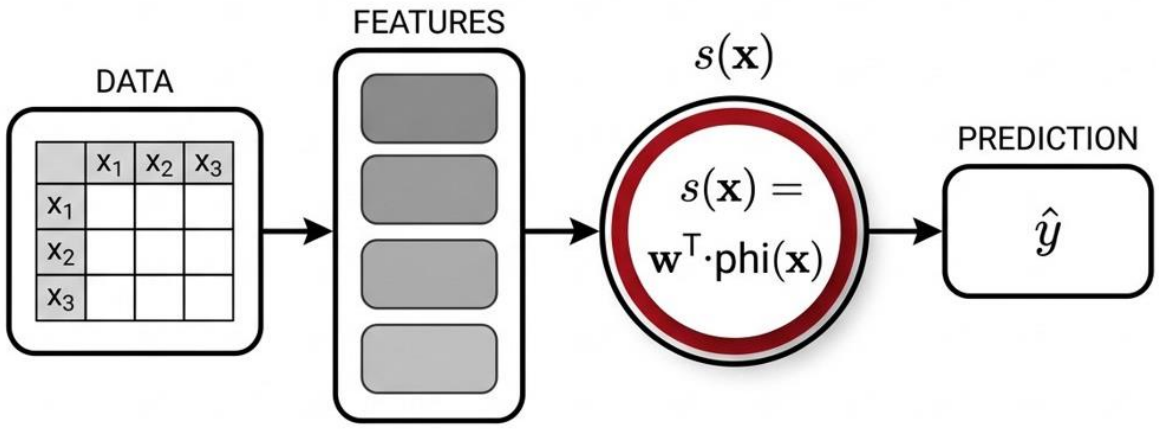
https://oc.sjtu.edu.cn/courses/89538

Part of lecture credits: CS221 - Stanford University

Recap: What Did We Build Last Time?

- Data → Features $\phi(x)$
- Score function: $s(x) = w \cdot \phi(x)$
- Learning = choose w

Question: How do we turn this score into predictions?

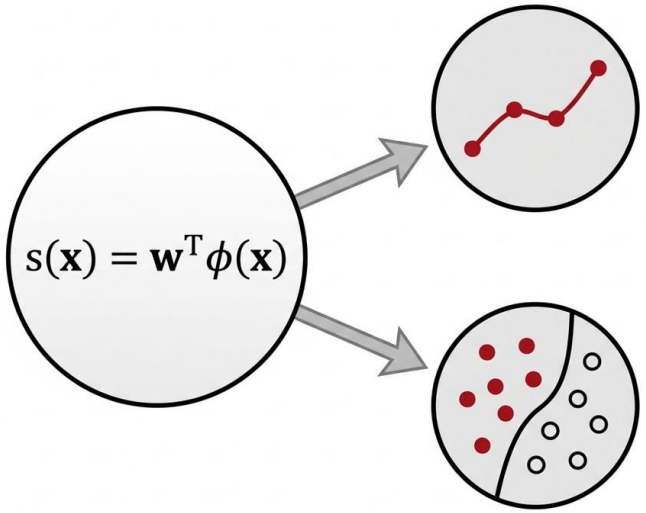


A Common Skeleton

$$s(x) = w \cdot \phi(x)$$

- Regression: output the score directly
- Classification: transform the score

Key Idea: The score is the common core



The Three Design Decisions

1. Hypothesis class: which predictors are possible?

   - Linear, curve, ...
2. Loss / objective: how good is a predictor?
3. Optimization: how do we compute the best predictor?

This lecture: focus on 1 + 2

Linear Regression

The Discovery of Ceres（谷神星）

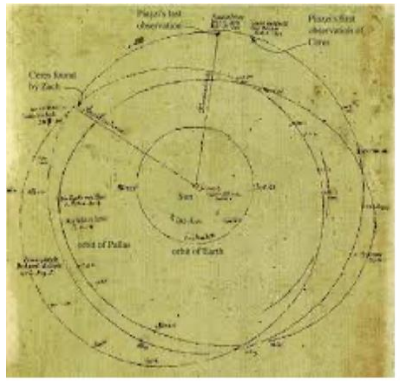
1801: astronomer Piazzi discovered Ceres, made 19 observations of location before it was obscured by the sun

Time	Right ascension	Declination
Jan 01, 20:43:17.8	50.91	15.24
Jan 02, 20:39:04.6	50.84	15.30
...	...	...
Feb 11, 18:11:58.2	53.51	18.43

When and where will Ceres be observed again?

Gauss's triumph

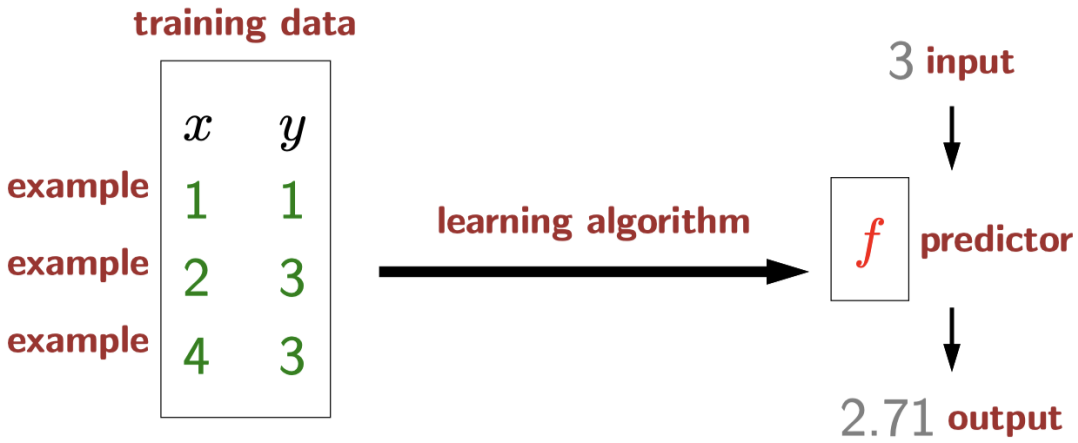
September 1801: Gauss took Piazzi's data and created a model of Ceres's orbit, makes prediction

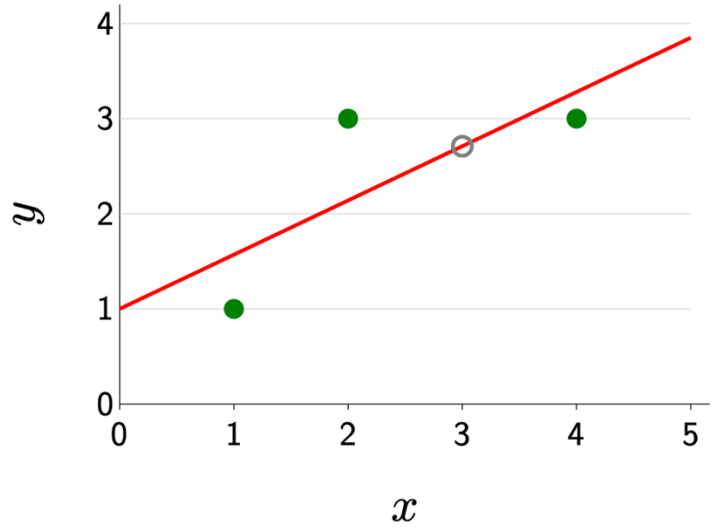


December 1801: Ceres located with in 1/2 degree of Gauss's prediction, much more accurate than other astronomers

Method: least squares linear regression (最小二乘法)

Linear Regression Framework





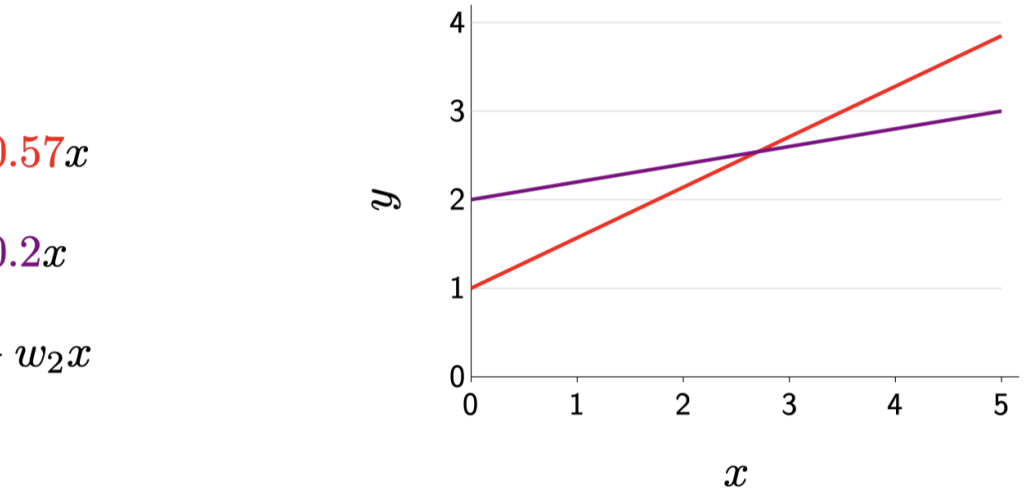
- Dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$
- Target: $y_i \in \mathbb{R}$
- Model: $f_w(x) = w^T \phi(x)$

Hypothesis Class: Which Predictors?

$$f(x) = 1 + 0.57x$$

$$f(x) = 2 + 0.2x$$

$$f(x) = w_1 + w_2 x$$



weight vector $\mathbf{w} = [w_1, w_2]$

Hypothesis class:

$$\mathcal{H} = f_w : w \in \mathbb{R}^2$$

Loss Function: How Good is a Predictor?

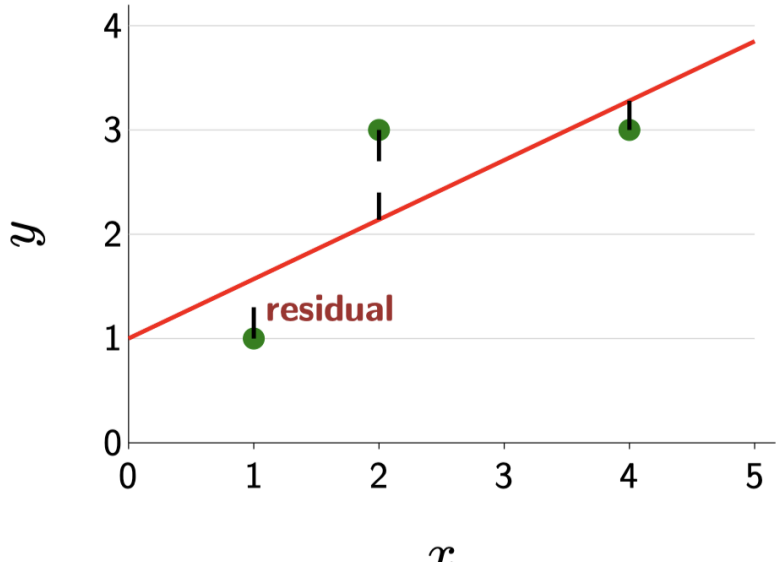
$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$\mathbf{w} = [1, 0.57]$

$\phi(x) = [1, x]$

training data $\mathcal{D}_{\text{train}}$

x	y
1	1
2	3
4	3



$$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$$ **squared loss**

$$\text{Loss}(1, 1, [1, 0.57]) = ([1, 0.57] \cdot [1, 1] - 1)^2 = 0.32$$

$$\text{Loss}(2, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 2] - 3)^2 = 0.74$$

$$\text{Loss}(4, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 4] - 3)^2 = 0.08$$

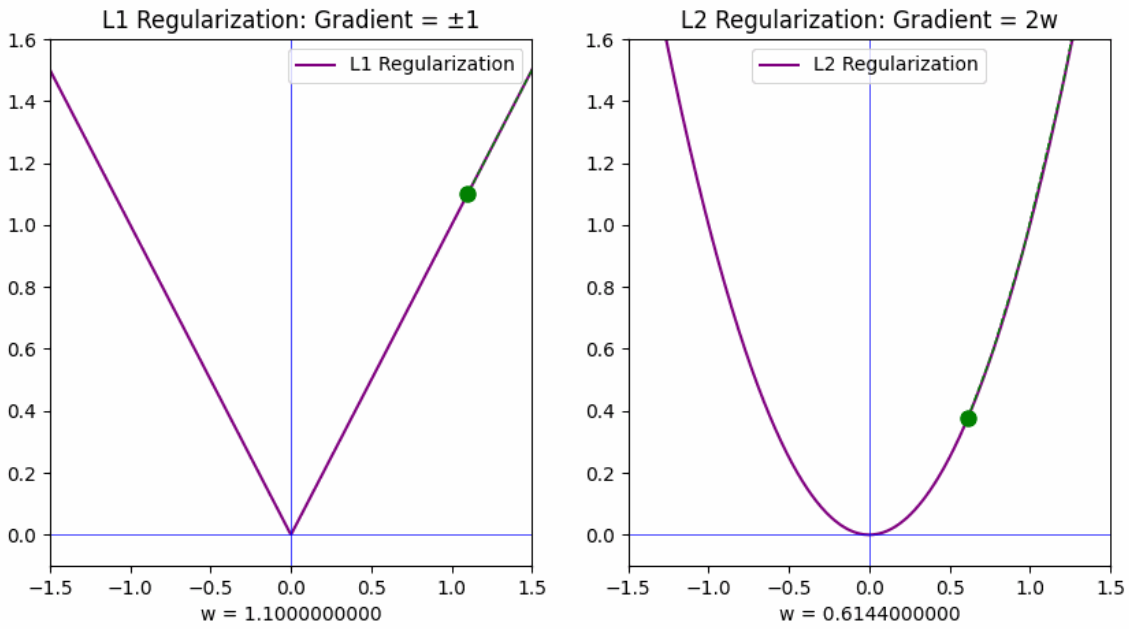
$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w})$$

Why Squared Loss?

- Nonnegative
- Penalizes large errors
- Smooth (differentiable)

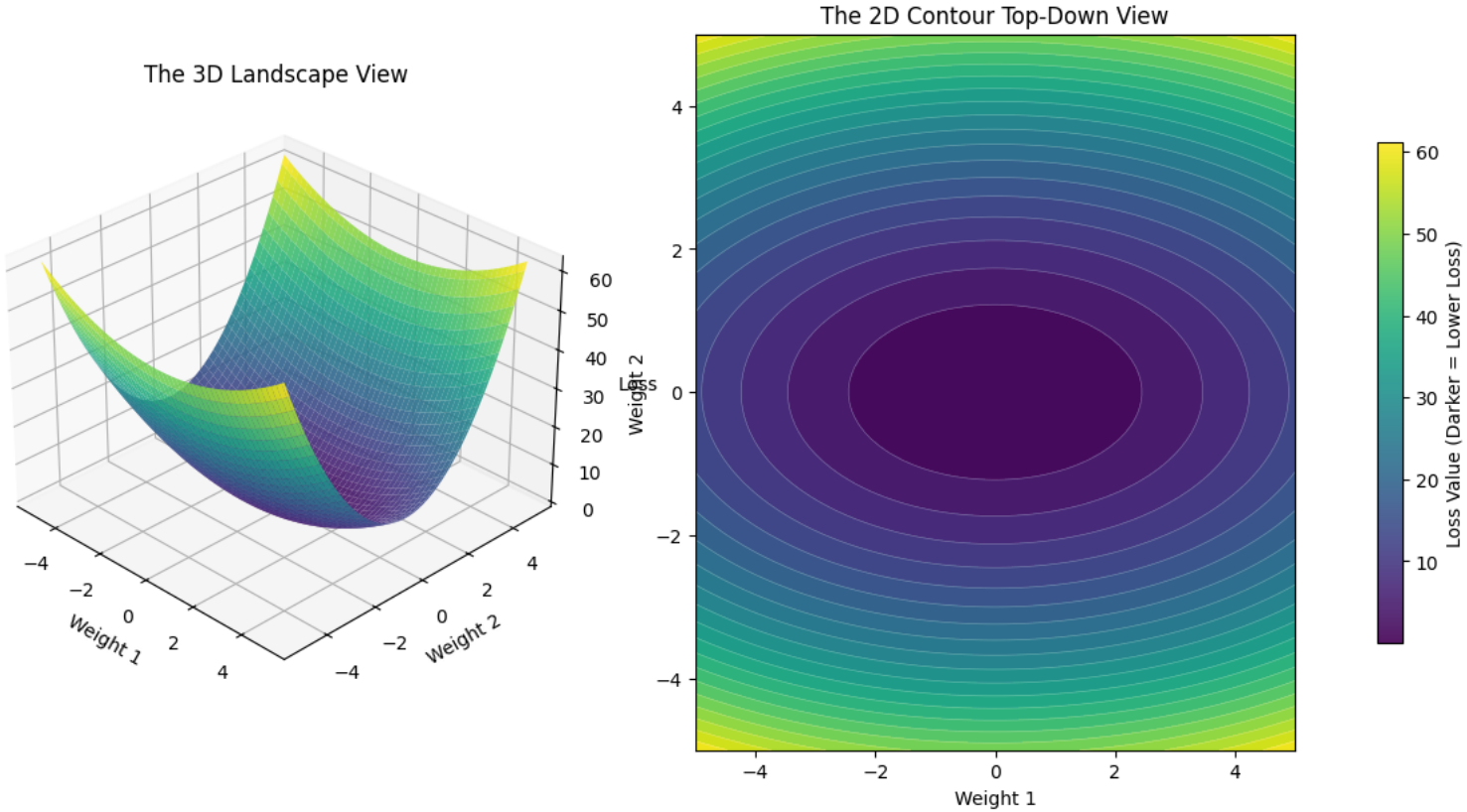
Discussion: What changes if we use absolute error instead?

$$\text{Loss}(x, y, w) = |f_w(x) - y|$$



Squared Loss: Visualizations

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (f_{\mathbf{w}}(x) - y)^2$$

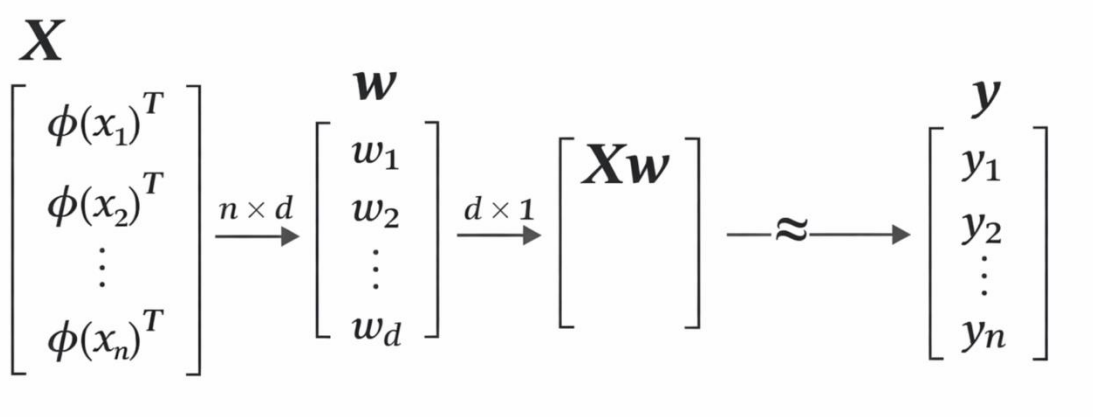


Matrix Form of Linear Regression

- Let

$$X = \begin{bmatrix} \phi(x_1)^\top \\ \phi(x_2)^\top \\ \vdots \\ \phi(x_n)^\top \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Then



$$\hat{R}(w) \propto \|Xw - y\|_2^2$$

Closed-Form Solution

Find the closed-form solution of w that minimizes $\mathcal{L}(w)$ using least squared.

$$\begin{aligned} L(w) &= \|Xw - y\|_2^2 \\ &= (Xw - y)^T (Xw - y) \\ &= w^T X^T Xw - 2y^T Xw + y^T y \end{aligned}$$

Compute the gradient w.r.t. w:

$$\nabla_w L(w) = 2X^T Xw - 2X^T y$$

At optimum:

$$\nabla_w L(w) = 0$$

So:

$$2X^T Xw - 2X^T y = 0$$

$$X^T Xw = X^T y$$

If $X^T X$ is invertible:

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

Complexity: $O(nd^2 + d^3)$ n: number of samples d: number of features

[1] Detailed derivations of solving LR with least squared: https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

Probabilistic View of Regression

Why squared loss:

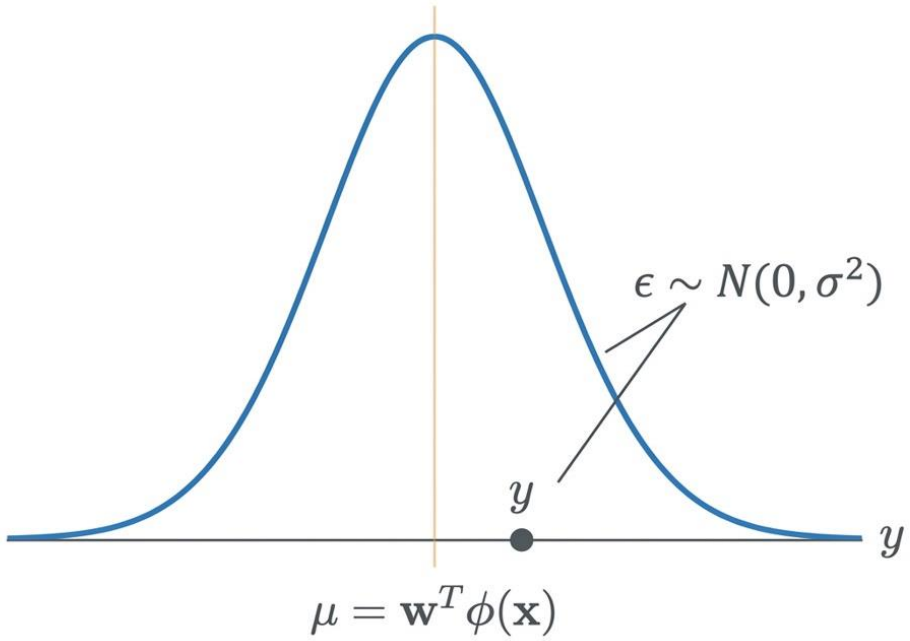
- Assume generative noise model:

$$y = w^\top \phi(x) + \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

- Then conditional distribution:

$$y \mid x \sim \mathcal{N}\left(w^\top \phi(x), \sigma^2\right)$$



Maximum Likelihood for Linear Regression

- Likelihood of the dataset:

$$L(w, \sigma^2) = \prod_{i=1}^{n} p(y_i \mid x_i; w, \sigma^2)$$

$$= (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^\top \phi(x_i))^2 \right)$$

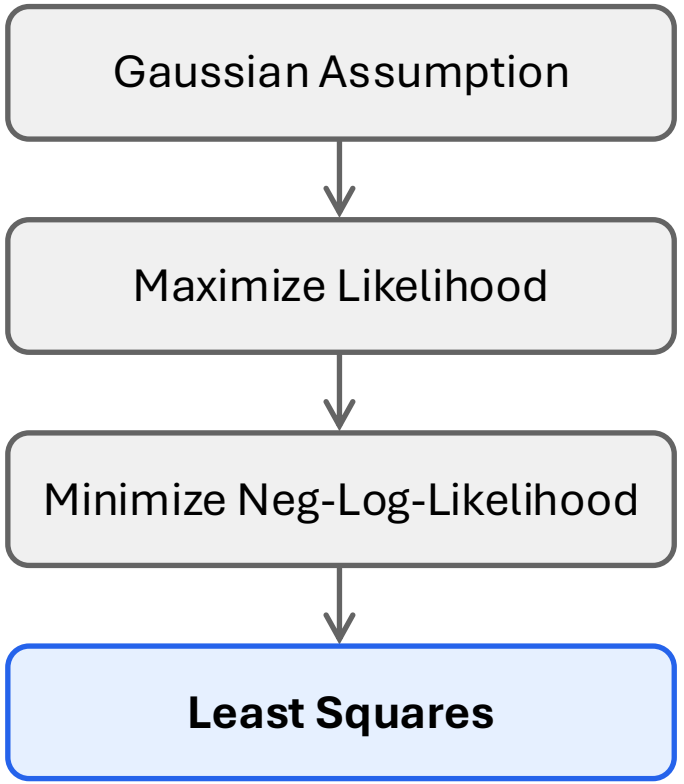
- Maximize the likelihood $\max_w L(w, \sigma^2)$

=> Minimize the negative log-likelihood (NLL)

maximize $\log L(w, \sigma^2) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^\top \phi(x_i))^2$

minimize $\sum_{i=1}^{n} (y_i - w^\top \phi(x_i))^2$

squared loss

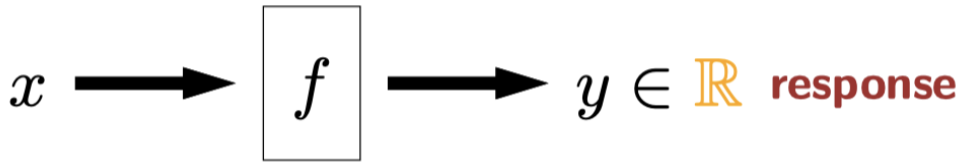


From Regression to Classification

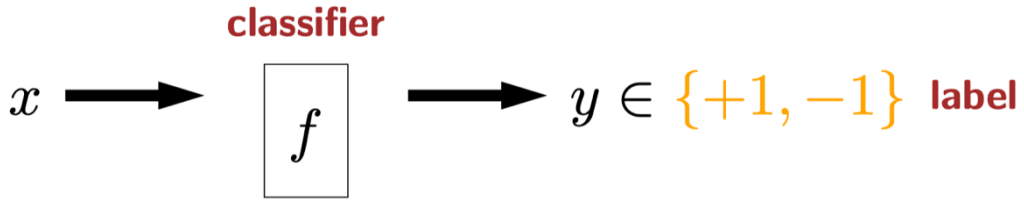
What if $y \in \{+1,-1\}$?

Transition: What Changes in Classification

Regression



Classification



Can we still use $f_w(x) = w^\top \phi(x)$?

Problem:

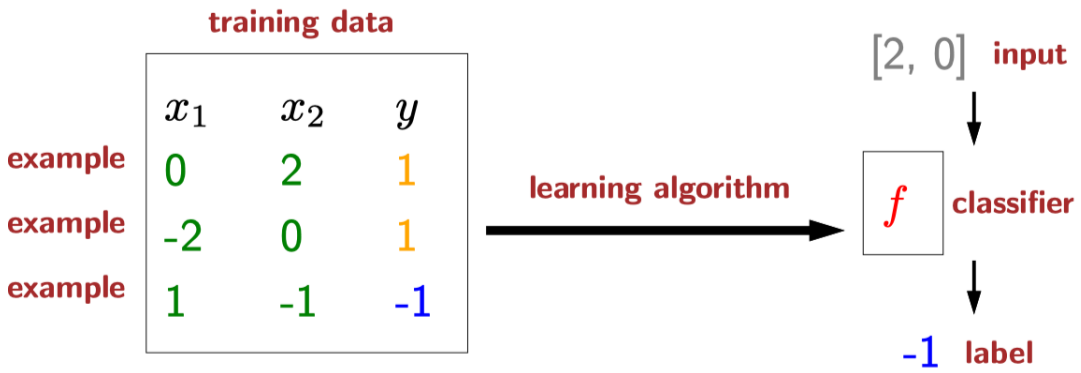
- score can be any real number
- but labels are discrete
- we also want probabilities

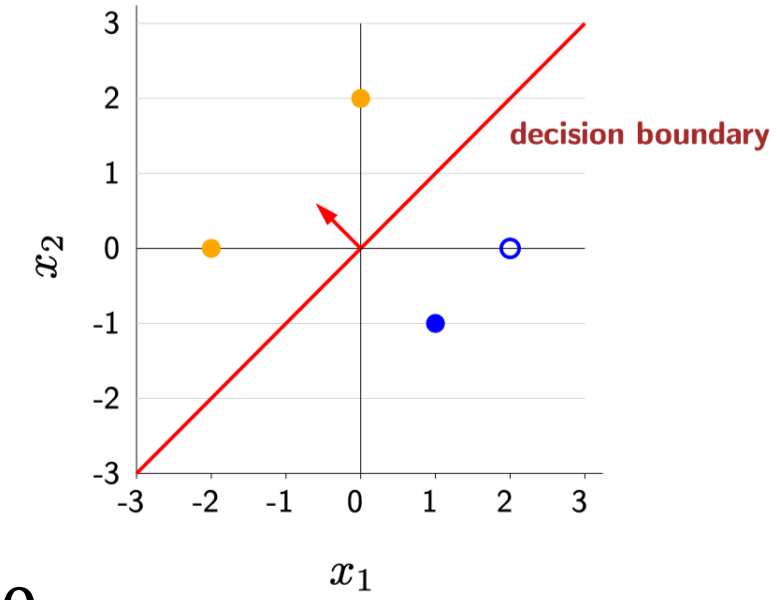
A Naïve Linear Classifier

A hard decision rule would be:

$$f(x) = \text{sign}\big(w \cdot \phi(x)\big)$$

$$\text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$





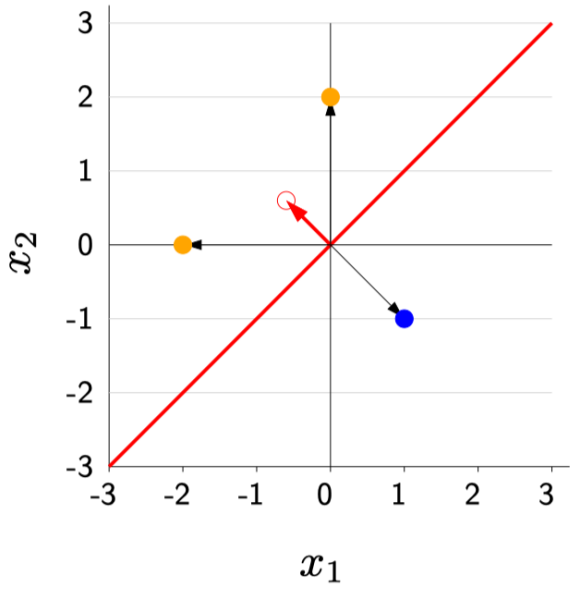
Decision boundary: x such that $w \cdot \phi(x) = 0$

Linear Classifier: Example

$$f(x) = \text{sign}(\overbrace{[-0.6, 0.6]}^{\mathbf{w}} \cdot \overbrace{[x_1, x_2]}^{\phi(x)})$$

$$\text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$

x_1	x_2	$f(x)$
0	2	1
-2	0	1
1	-1	-1



$$f([0, 2]) = \text{sign}([-0.6, 0.6] \cdot [0, 2]) = \text{sign}(1.2) = 1$$

$$f([-2, 0]) = \text{sign}([-0.6, 0.6] \cdot [-2, 0]) = \text{sign}(1.2) = 1$$

$$f([1, -1]) = \text{sign}([-0.6, 0.6] \cdot [1, -1]) = \text{sign}(-1.2) = -1$$

Decision boundary: $x_1 = x_2$

Loss Function: How Good is a Classifier?

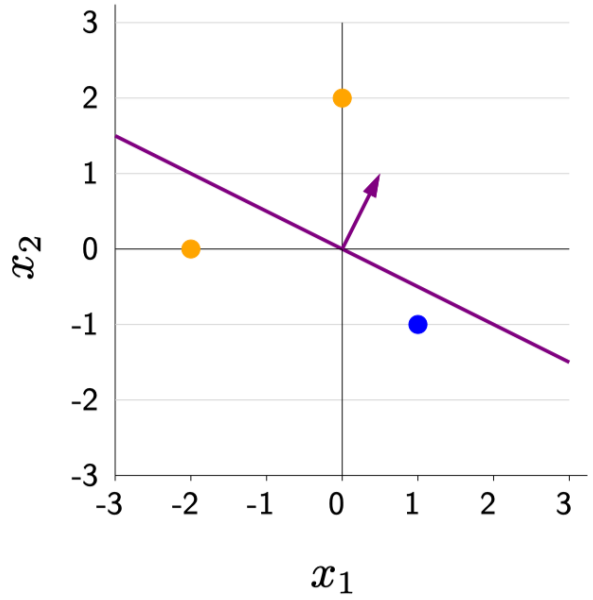
$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$$\mathbf{w} = [0.5, 1]$$

$$\phi(x) = [x_1, x_2]$$

training data $\mathcal{D}_{\text{train}}$

x_1	x_2	y
0	2	1
-2	0	1
1	-1	-1



- Zero-one loss: $\text{Loss}_{0-1}(x, y, w) = \mathbf{1}[f_w(x) \neq y]$

$$\text{Loss}([0, 2], 1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [0, 2]) \neq 1] = 0$$

$$\text{Loss}([-2, 0], 1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [-2, 0]) \neq 1] = 1$$

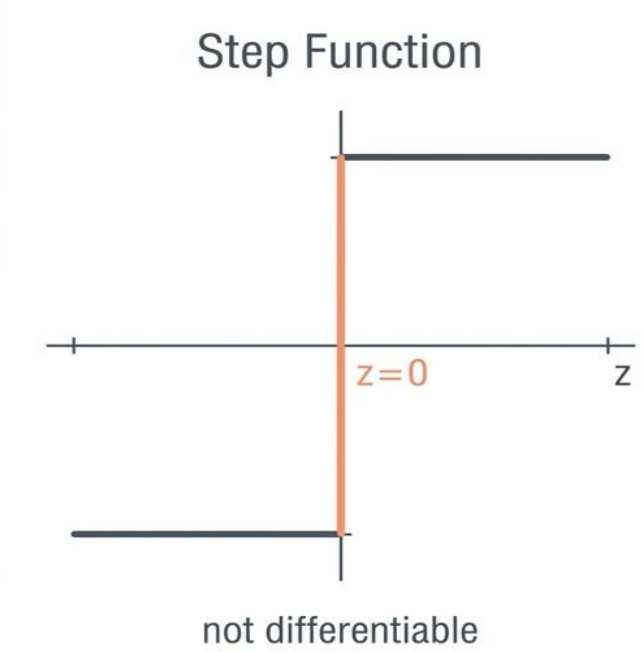
$$\text{Loss}([1, -1], -1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [1, -1]) \neq -1] = 0$$

$$\text{TrainLoss}([0.5, 1]) = 0.33$$

Why This Fails

- Problems:
  - **Not differentiable**: gradient is 0 almost everywhere
  - No notion of confidence
  - No probability output
- Question:
  Can we learn from this?
- Answer:
  Not effectively with gradient methods.

$$\text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$



What Do We Want?

We need a link function g such that:

correct Smooth and differentiable

correct Outputs in range [0, 1]

correct Interpretable as probability

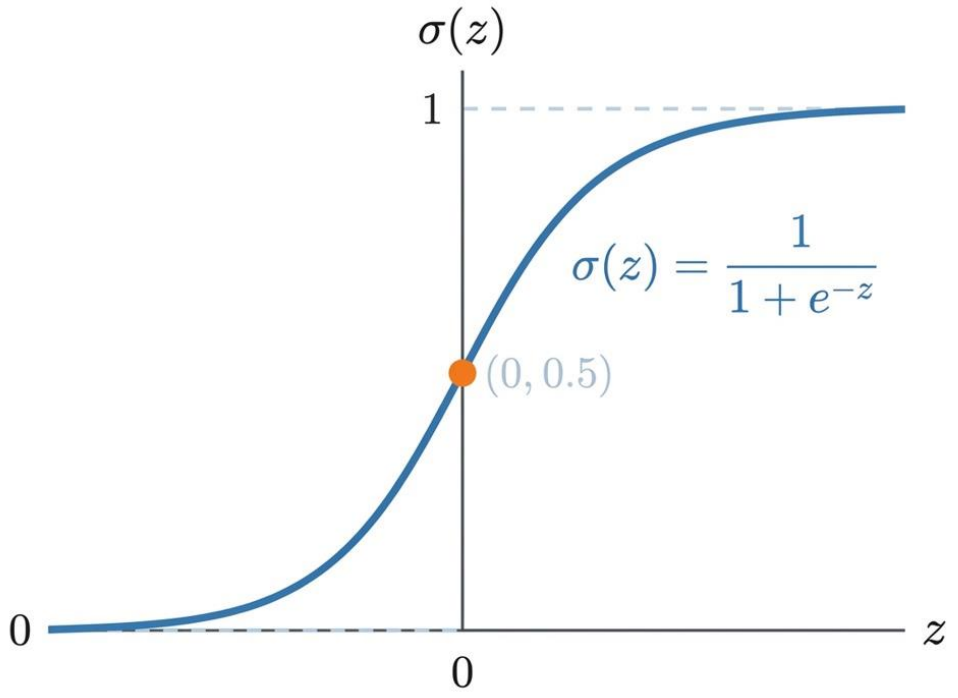
$$p(y = 1 \mid x) = g(w \cdot \phi(x))$$

Logistic Function

- Define Logistic (Sigmoid):

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

- Properties:
  - Range: (0, 1)
  - Monotonic increasing
  - Symmetric: $\sigma(0) = 0.5$
  - Smooth & differentiable

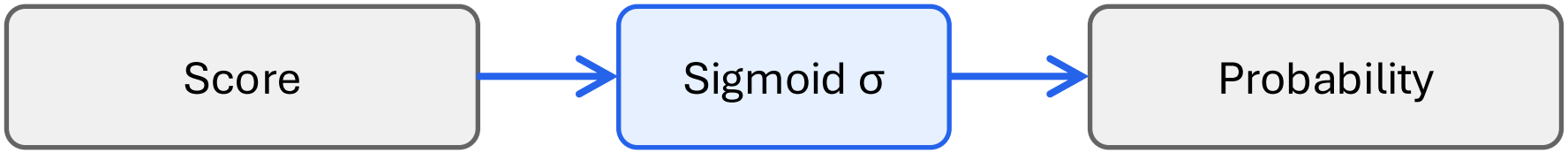


Sigmoid maps real numbers to (0,1)

Logistic Model

Logistic Regression defines:

$$p(y = 1|x) = \sigma(w^T \phi(x))$$
$$p(y = 0|x) = 1 - \sigma(w^T \phi(x))$$



Decision Rule

- Prediction

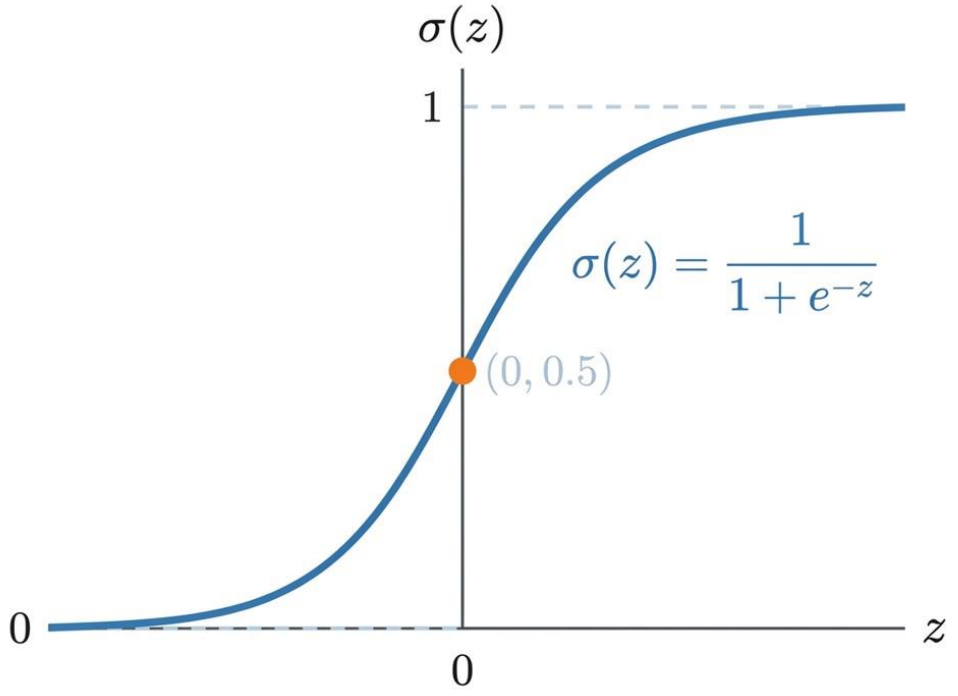
  $\hat{y} = 1$ if $\sigma\left(w^T \cdot \phi(x)\right) > 0.5$

  $\hat{y} = 0$ otherwise

- Equivalent to

  $w^T \cdot \phi(x) > 0$

Where does logistic (σ) come from?



Sigmoid maps real numbers to (0,1)

Bernoulli Model

- Model the label as a coin flip:

$$y|x \sim Bernoulli(p)$$

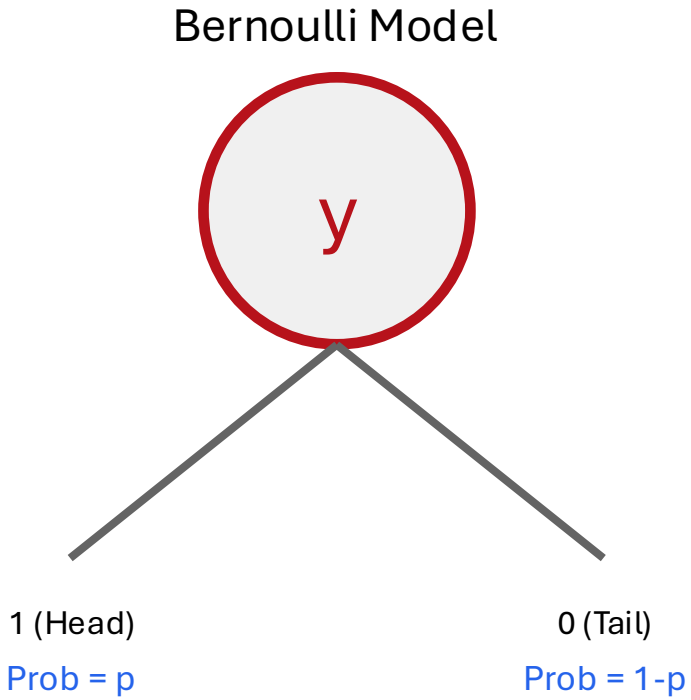
- Link to score:

$$p = \sigma\left(w^T \phi(x)\right)$$

- Likelihood:

$$p(y|x) = p^y \, (1-p)^{(1-y)}$$

This unified the two cases y=0 and y=1



$$p(y \mid x; w) = \sigma(w^\top \phi(x))^y (1 - \sigma(w^\top \phi(x)))^{1-y}$$

Maximum Likelihood

- We want to maximize probability of observing the data:

$$L(w) = \prod_{i=1}^{n} p(y_i \mid x_i; w)$$

$$= \prod_{i=1}^{n} \sigma(w^\top \phi(x_i))^{y_i} (1 - \sigma(w^\top \phi(x_i)))^{1-y_i}$$

- Log-likelihood:

Maximize $$\log L(w) = \sum_{i=1}^{n} \left[y_i \log \sigma(w^\top \phi(x_i)) + (1 - y_i) \log(1 - \sigma(w^\top \phi(x_i))) \right]$$

Minimize $$\mathcal{L}(w) = -\sum_{i=1}^{n} \left[y_i \log \sigma(w^\top \phi(x_i)) + (1 - y_i) \log(1 - \sigma(w^\top \phi(x_i))) \right]$$

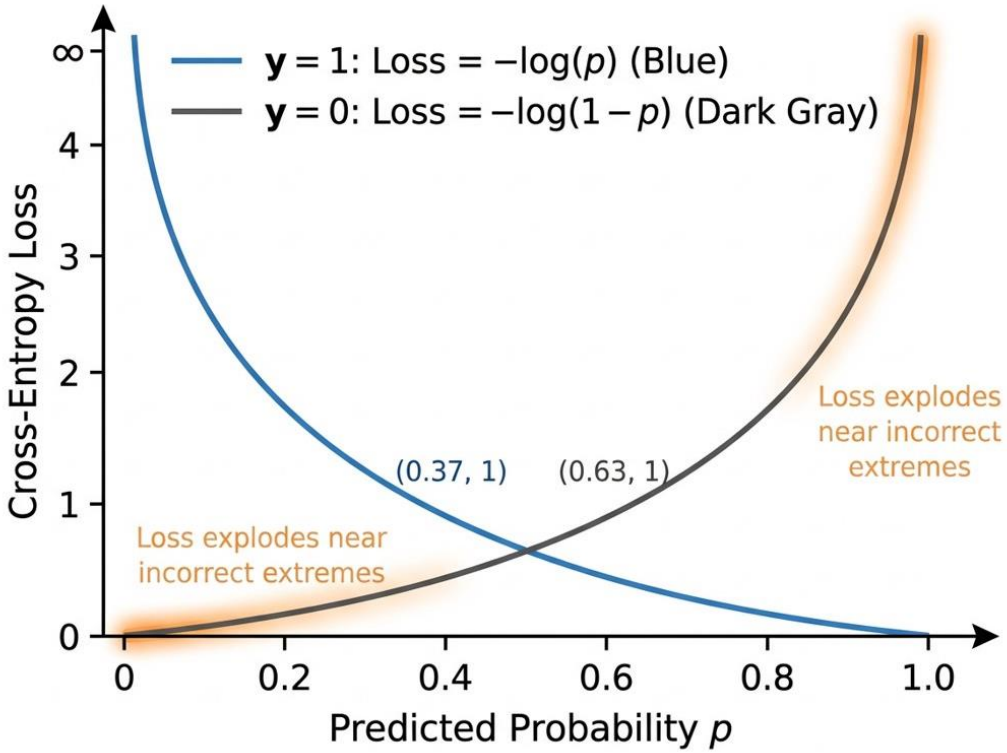
Cros-entropy loss

Cross-Entropy Loss

- Negative Log-Likelihood:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]$$

- This is Cross-Entropy Loss.
- Key Properties:
  - Convex in w (global minimum)
  - Heavy penalty for confident errors
  - Smooth gradients



Loss explodes when confident prediction is wrong

Regression vs Classification

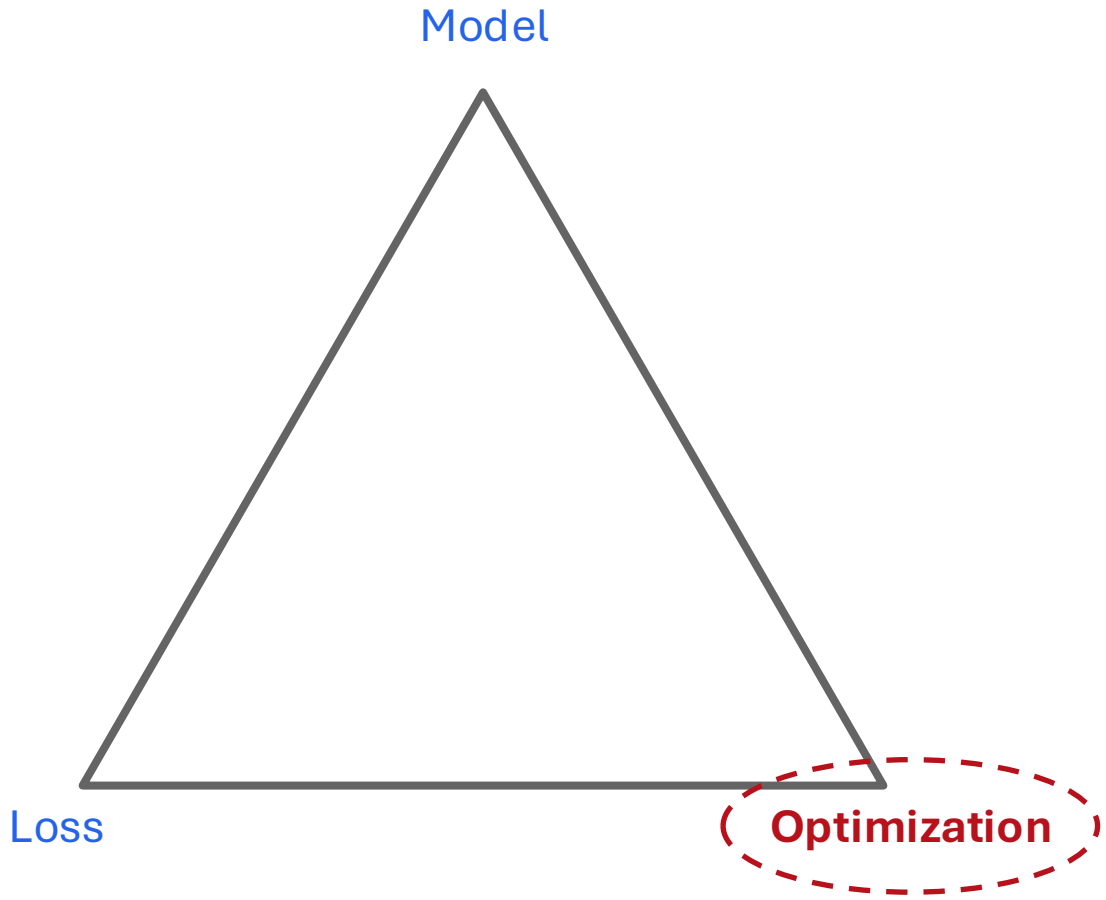
Component	Regression	Classification
Assumption	Gaussian	Bernoulli
Output Space	Real ($\mathbb{R}$)	Binary {0,1} / Prob [0,1]
Loss Function	Squared Error	Cross-Entropy

What's Missing?

- We defined:
  - Model (Linear)
  - Loss (Squared / Cross-Entropy)
- **But how do we solve it?**

$$w^* = \arg\min_w \mathcal{L}(x, y, w)$$

- Practical Questions:
  - Efficient gradients?
  - Convexity guarantees?
  - Scalability to big data?



Next Lecture: Optimization

- Topics:
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
  - Convexity & Convergence