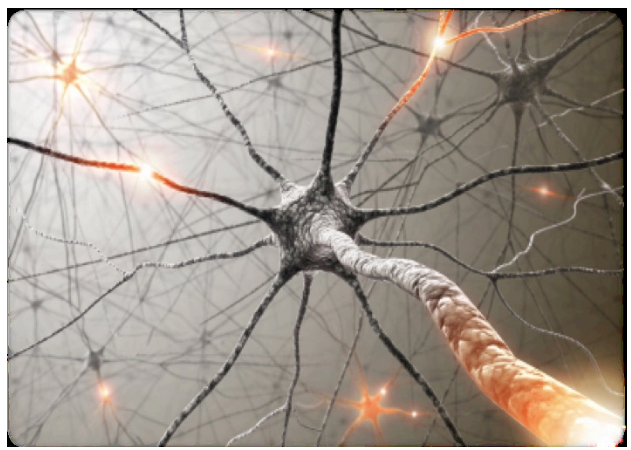
# 02 Linear Predictor



# Roadmap

- Linear predictors

- Loss minimization

- Stochastic gradient descent

# Application: spam classification

- Input: x = email message

```
From:  pliang@cs.stanford.edu
Date:  September 26, 2018
Subject:  CS221 announcement

Hello students,
 I've attached the answers to homework 1...
```

```
From:  a9k62n@hotmail.com
Date:  September 26, 2018
Subject:  URGENT

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

- Output: y $\in \{spam, non-spam\}$
- Objective: obtain a predictor f

$$x \longrightarrow \boxed{f} \longrightarrow y$$

# Types of prediction tasks

- Binary classification (e.g., email ) (spam/not spam):

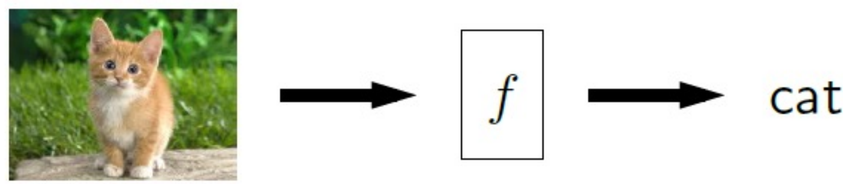
$$x \longrightarrow \boxed{f} \longrightarrow y \in \{+1, -1\}$$

- Regression (e.g., location, year ) (housing price):

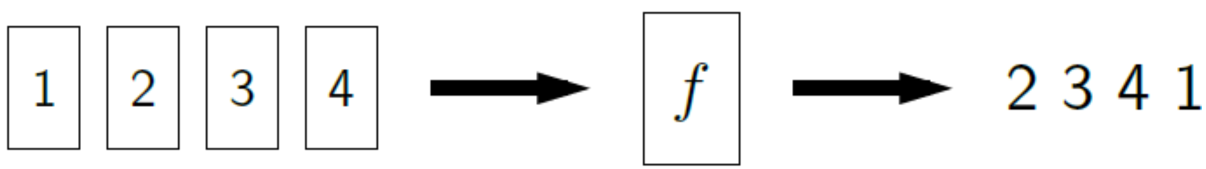
$$x \longrightarrow \boxed{f} \longrightarrow y \in \mathbb{R}$$

# Types of prediction tasks

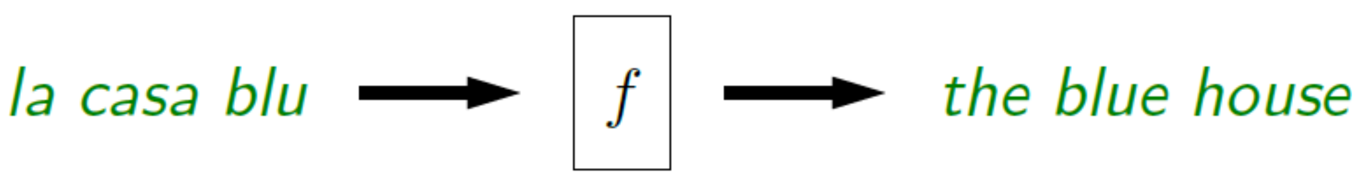
- Multiclass classification: y is a category




- Ranking: y is a permutation




- Structured prediction: y is an object which is built from parts



# Question

- Give an example of a prediction task (e.g., image, face/not face).

# Data

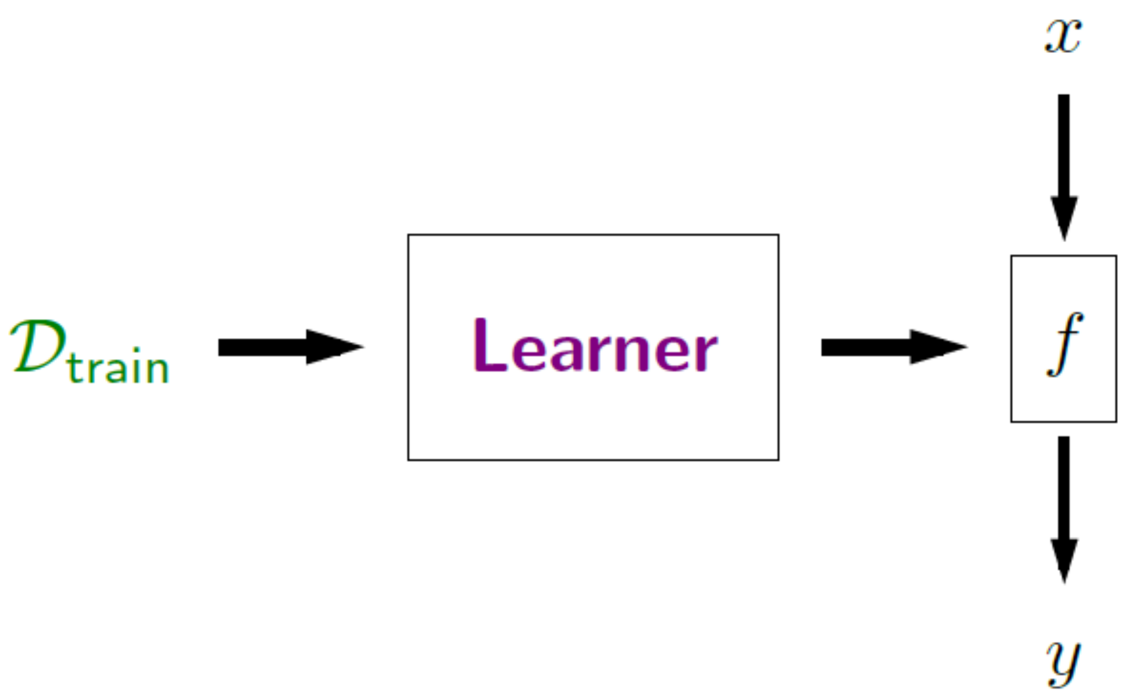
- Example: species that y is the ground-truth output for x

$$(x, y)$$

- Training data: list of examples

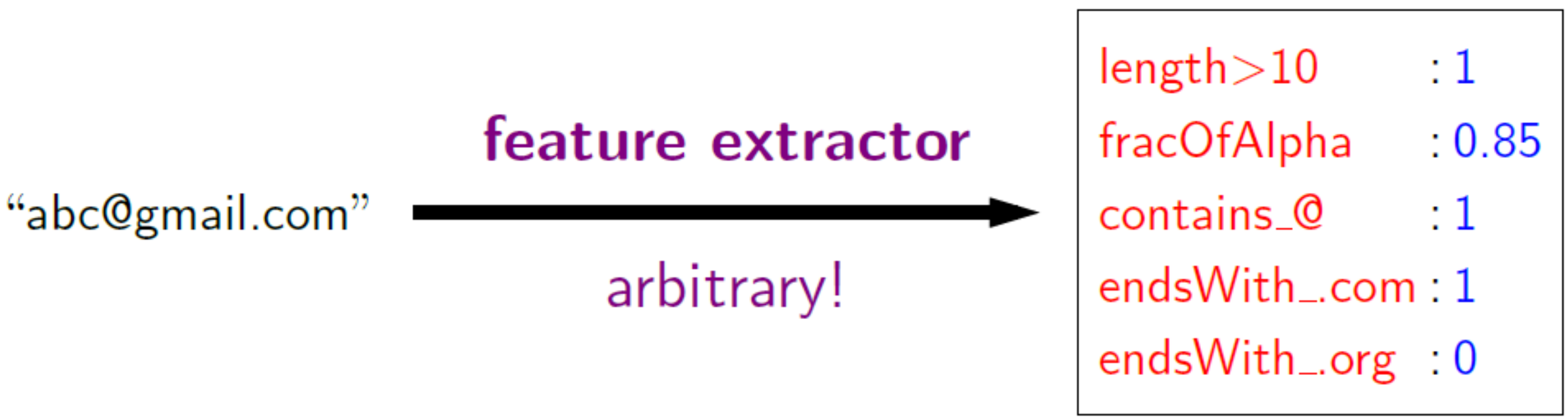
$$\begin{aligned}
\mathcal{D}_{\text{train}} = [ & \\
& (\text{"...10m dollars..."}, +1), \\
& (\text{"...CS221..."}, \quad -1), \\
]
\end{aligned}$$

# Framework



# Feature extraction

- Example task: predict y, whether a string x is an email address
- Question: what properties of x might be relevant for predicting y?
- Feature extractor: Given input x, output a set of (feature name, feature value) pairs.



# Feature vector notation

- Mathematically, feature vector doesn't need feature names:

| | |
|---|---|
| length>10 | : 1 |
| fracOfAlpha | : 0.85 |
| contains_@ | : 1 |
| endsWith_.com | : 1 |
| endsWith_.org | : 0 |

$$\rightarrow \begin{bmatrix} 1 \\ 0.85 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

**Definition: feature vector**

For an input x, its feature vector is:

$$\phi(x) = [\phi_1(x), \ldots, \phi_d(x)].$$

Think of $\phi(x) \in \mathbb{R}^d$ as a point in a high-dimensional space.

# Weight vector

- Weight vector: for each feature j, have real number $w_j$ representing contribution of feature to prediction

| | |
|---|---|
| length>10 | :-1.2 |
| fracOfAlpha | :0.6 |
| contains_@ | :3 |
| endsWith_.com | :2.2 |
| endsWith_.org | :1.4 |
| ... | |

# Linear predictors

Weight vector $\mathbf{w} \in \mathbb{R}^d$

| | |
|---|---|
| length>10 | :-1.2 |
| fracOfAlpha | :0.6 |
| contains_@ | :3 |
| endsWith_.com | :2.2 |
| endsWith_.org | :1.4 |

Feature vector $\phi(x) \in \mathbb{R}^d$

| | |
|---|---|
| length>10 | :1 |
| fracOfAlpha | :0.85 |
| contains_@ | :1 |
| endsWith_.com | :1 |
| endsWith_.org | :0 |

Score: weighted combination of features

$$\mathbf{w} \cdot \phi(x) = \sum_{j=1}^{d} w_j \phi(x)_j$$

Example: -1.2(1) + 0.6(0.85) + 3(1) + 2.2(1) + 1.4(0) = 4.51

# Linear predictors

- Weight vector $\mathbf{w} \in \mathbb{R}^d$
- Feature vector $\phi(x) \in \mathbb{R}^d$
- For binary classification:

**Definition: (binary) linear classifier**

$$f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \phi(x) > 0 \\ -1 & \text{if } \mathbf{w} \cdot \phi(x) < 0 \\ ? & \text{if } \mathbf{w} \cdot \phi(x) = 0 \end{cases}$$

# Geometric intuition

- A binary classier $f_{\mathbf{w}}$ defines a hyperplane with normal vector $\mathbf{w}$.

( $\mathbb{R}^2 \implies$ hyperplane is a line; $\mathbb{R}^3 \implies$ hyperplane is a plane)

Example:

$$\mathbf{w} = [2, -1]$$

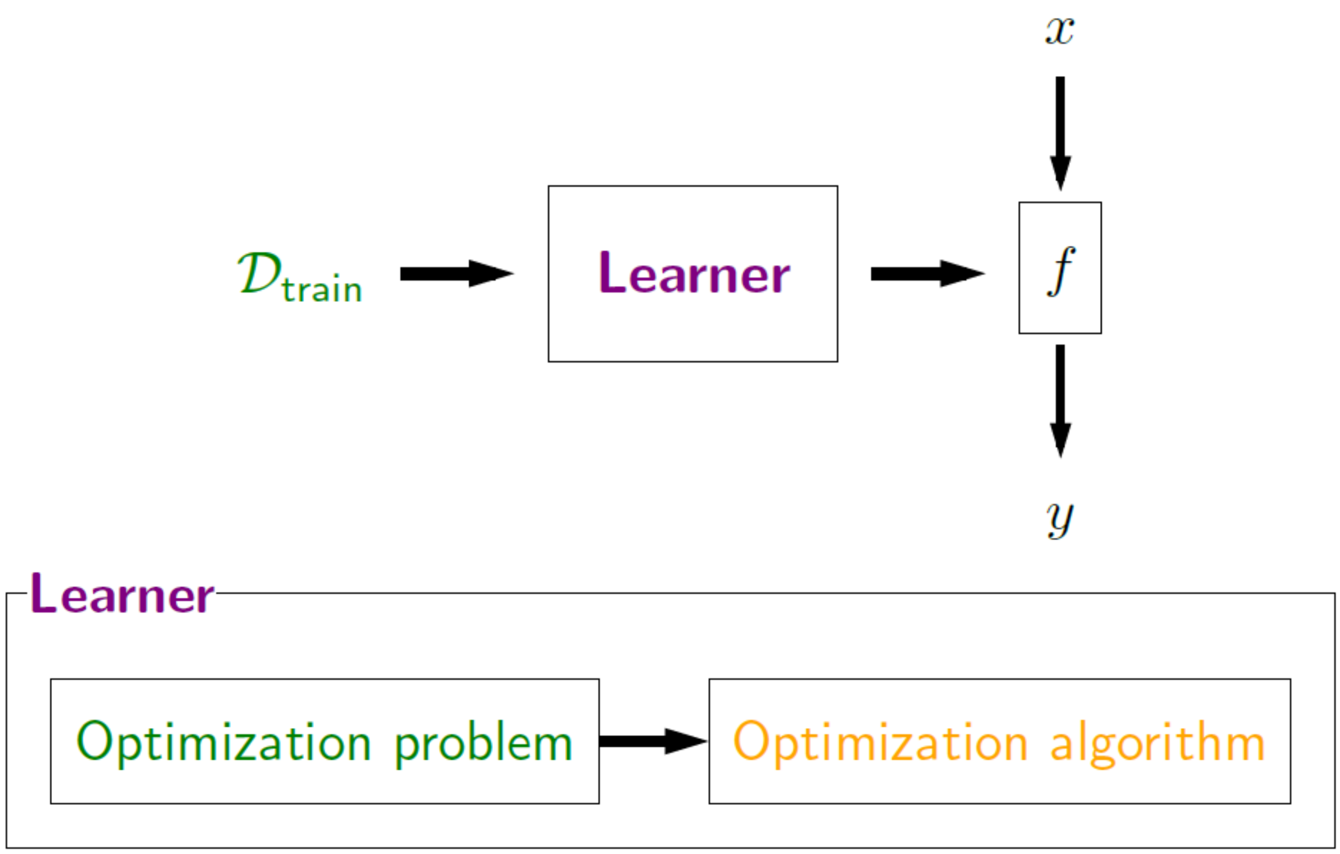
$$\phi(x) \in \{[2,0],[0,2],[2,4]\}$$

[blackboard]

# Roadmap

- Linear predictors
- Loss minimization
- Stochastic gradient descent

# Framework



$x$
$\mathcal{D}_{\text{train}}$
Learner
$f$
$y$
Learner
Optimization problem
Optimization algorithm

# Loss functions

**Definition: loss function**

A loss function Loss $(x, y, \mathbf{w})$ quantifies how unhappy you would be if you used w to make a prediction on x when the correct output is y. It is the objective we want to minimize.

# Score and margin

- Correct label: $y$
- Predicted label: $y' = f_{\mathbf{w}}(x) = \mathrm{sign}(\mathbf{w} \cdot \phi(x))$
- Example: $\mathrm{w} = [2, -1], \emptyset(x) = [2,0], y = 1$

**Definition: score**

The score on an example $(x, y)$ is $\mathbf{w} \cdot \phi(x)$, how confident we are in predicting +1

**Definition: margin**

The margin on an example $(x, y)$ is $\mathbf{w} \cdot \phi(x)y$, how correct we are

# Question

- When does a binary classifier predict an error on an example?
    - margin less than 0
    - margin greater than 0
    - score less than 0
    - score greater than 0

# Binary classification

Example: $\mathrm{w} = [2, -1], \emptyset(x) = [2,0], y = 1$

Recall the binary classifier:

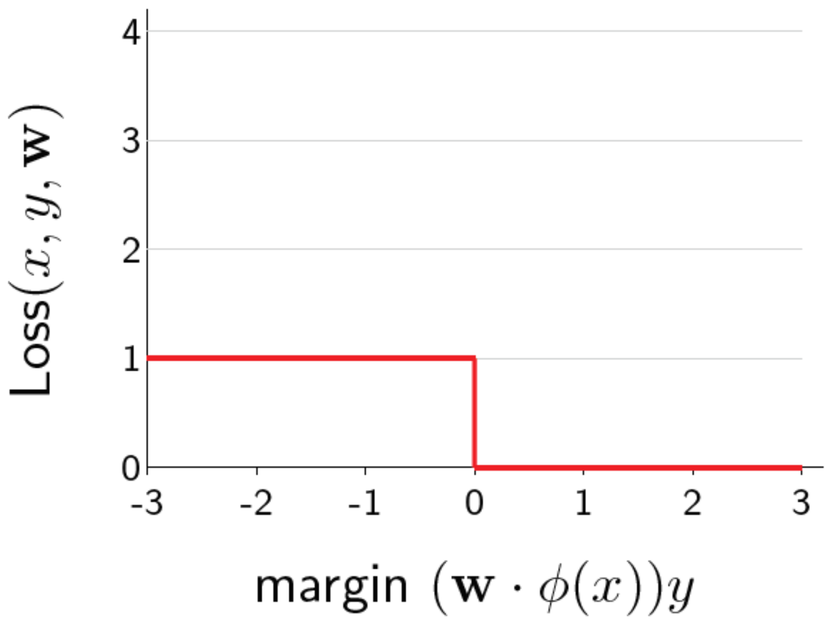
$$f_{\mathbf{w}}(x) = \mathrm{sign}(\mathbf{w} \cdot \phi(x))$$

**Definition: zero-one loss**

$$\mathrm{Loss}_{0\text{-}1}(x, y, \mathbf{w}) = \mathbf{1}[f_{\mathbf{w}}(x) \neq y]$$

$$= \mathbf{1}[\underbrace{(\mathbf{w} \cdot \phi(x))y}_{\text{margin}} \leq 0]$$

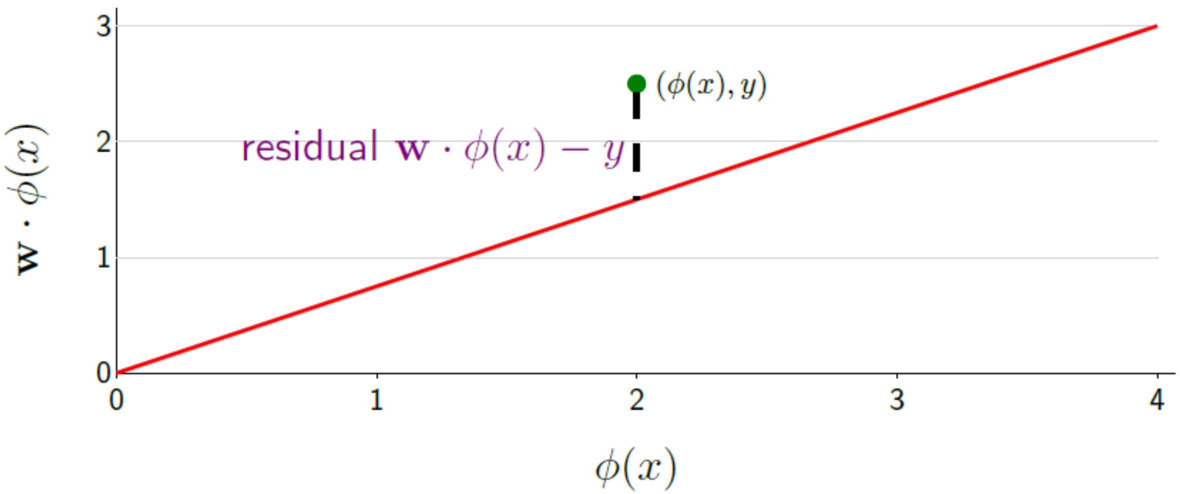
# Binary classification




$$\mathrm{Loss}_{0\text{-}1}(x, y, \mathbf{w}) = \mathbf{1}[(\mathbf{w} \cdot \phi(x))y \leq 0]$$

# Linear regression

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




**Definition: residual**

The residual is $(\mathbf{w} \cdot \phi(x)) - y$, the amount by which prediction $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$ overshoots the target y.

# Linear regression

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

**Definition: squared loss**

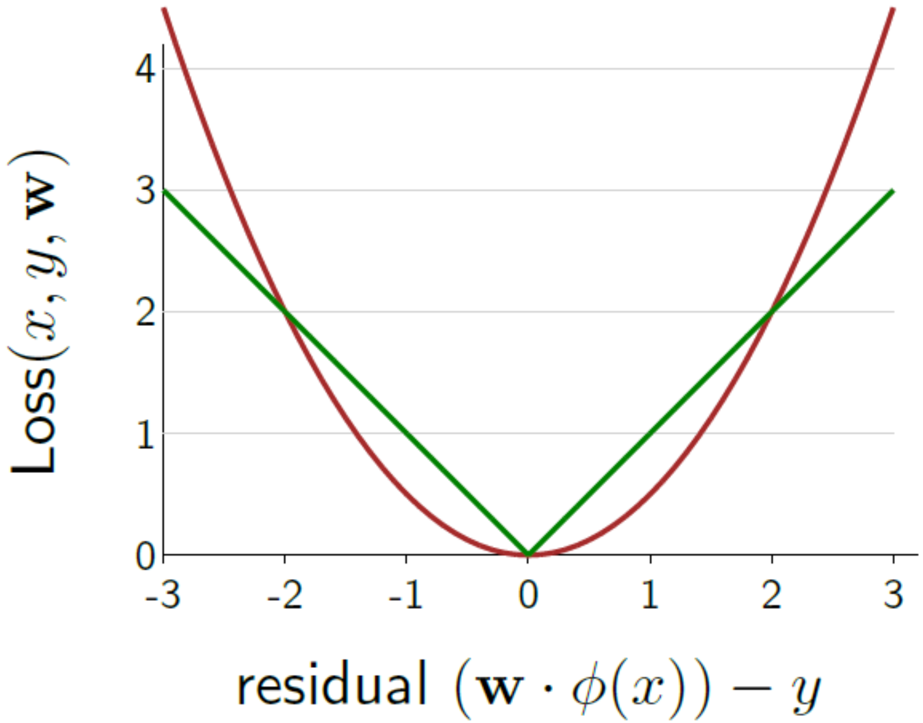
$$\text{Loss}_{\text{squared}}(x, y, \mathbf{w}) = (\underbrace{f_{\mathbf{w}}(x) - y}_{\text{residual}})^2$$

Example：

$$\mathbf{w} = [2, -1], \phi(x) = [2, 0], y = -1$$

$$\text{Loss}_{\text{squared}}(x, y, \mathbf{w}) = 25$$

# Regression loss functions




$$\text{Loss}_{\text{squared}}(x, y, \mathbf{w}) = (\mathbf{w} \cdot \phi(x) - y)^2$$

$$\text{Loss}_{\text{absdev}}(x, y, \mathbf{w}) = |\mathbf{w} \cdot \phi(x) - y|$$

# Loss minimization framework

So far: one example, Loss(x, y, w) is easy to minimize.

**Key idea: minimize training loss**

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

$$\min_{\mathbf{w}\in\mathbb{R}^d} \text{TrainLoss}(\mathbf{w})$$

Key: need to set w to make global tradeoffs—not every example can be happy.

# Which regression loss to use?

Example ： $\mathcal{D}_{\text{train}} = \{(1,0),(1,2),(1,1000)\}$

For least squares ($L_2$) regression ：

$$\text{Loss}_{\text{squared}}(x,y,\mathbf{w}) = (\mathbf{w}\cdot\phi(x) - y)^2$$

- w that minimizes training loss is mean y
- Mean: tries to accommodate every example, popular

For least absolute deviation ($L_1$) regression ：

$$\text{Loss}_{\text{squared}}(x,y,\mathbf{w}) = |\mathbf{w}\cdot\phi(x) - y|$$

- w that minimizes training loss is median y
- Median: more robust to outliers