Machine Learning (CS 181):
2. Linear Regression and Foundations

Contents

Contents

Basic Notation

$\mathbf{A} \in \mathbb{R}^{m \times n}$	matrices are bold caps
$\mathbf{a} \in \mathbb{R}^{m(\times 1)}$	vectors are bold lower, always column
$a \in \mathbb{R}$	scalars are lower case, non bold
$\mathcal{A}$	sets are script case
$\{\mathbf{a}_i\}_1^n$	a sequence of $\mathbf{a}_1 \dots \mathbf{a}_n$

- We distinguish between $\mathbf{a}_i$ and b_i. The first is the $i'th$ vector of a sequence, the second is the i'th scalar in $\mathbf{b}$.

Basic Notation

$\mathbf{A} \in \mathbb{R}^{m \times n}$	matrices are bold caps
$\mathbf{a} \in \mathbb{R}^{m(\times 1)}$	vectors are bold lower, always column
$a \in \mathbb{R}$	scalars are lower case, non bold
$\mathcal{A}$	sets are script case
$\{\mathbf{a}_i\}_1^n$	a sequence of $\mathbf{a}_1 \dots \mathbf{a}_n$

- We distinguish between $\mathbf{a}_i$ and b_i. The first is the $i'th$ vector of a sequence, the second is the i'th scalar in $\mathbf{b}$.

Machine Learning Setup

- Inputs
  - Input space: $\mathcal{X} = \mathbb{R}^m$
  - features, covariants, predictors, etc.
- Outputs
  - Output space: $\mathcal{Y}$
  - many different types of predictions
- Goal: Learn a hypothesis/model $h : \mathcal{X} \mapsto \mathcal{Y}$
  - $\hat{y} = h(\mathbf{x})$; model prediction

Machine Learning Setup

- Inputs
  - Input space: $\mathcal{X} = \mathbb{R}^m$
  - features, covariants, predictors, etc.
- Outputs
  - Output space: $\mathcal{Y}$
  - many different types of predictions
- Goal: Learn a hypothesis/model $h : \mathcal{X} \mapsto \mathcal{Y}$
  - $\hat{y} = h(\mathbf{x})$; model prediction

Machine Learning Setup

- Inputs
  - Input space: $\mathcal{X} = \mathbb{R}^m$
  - features, covariants, predictors, etc.
- Outputs
  - Output space: $\mathcal{Y}$
  - many different types of predictions
- Goal: Learn a hypothesis/model $h : \mathcal{X} \mapsto \mathcal{Y}$
  - $\hat{y} = h(\mathbf{x})$; model prediction

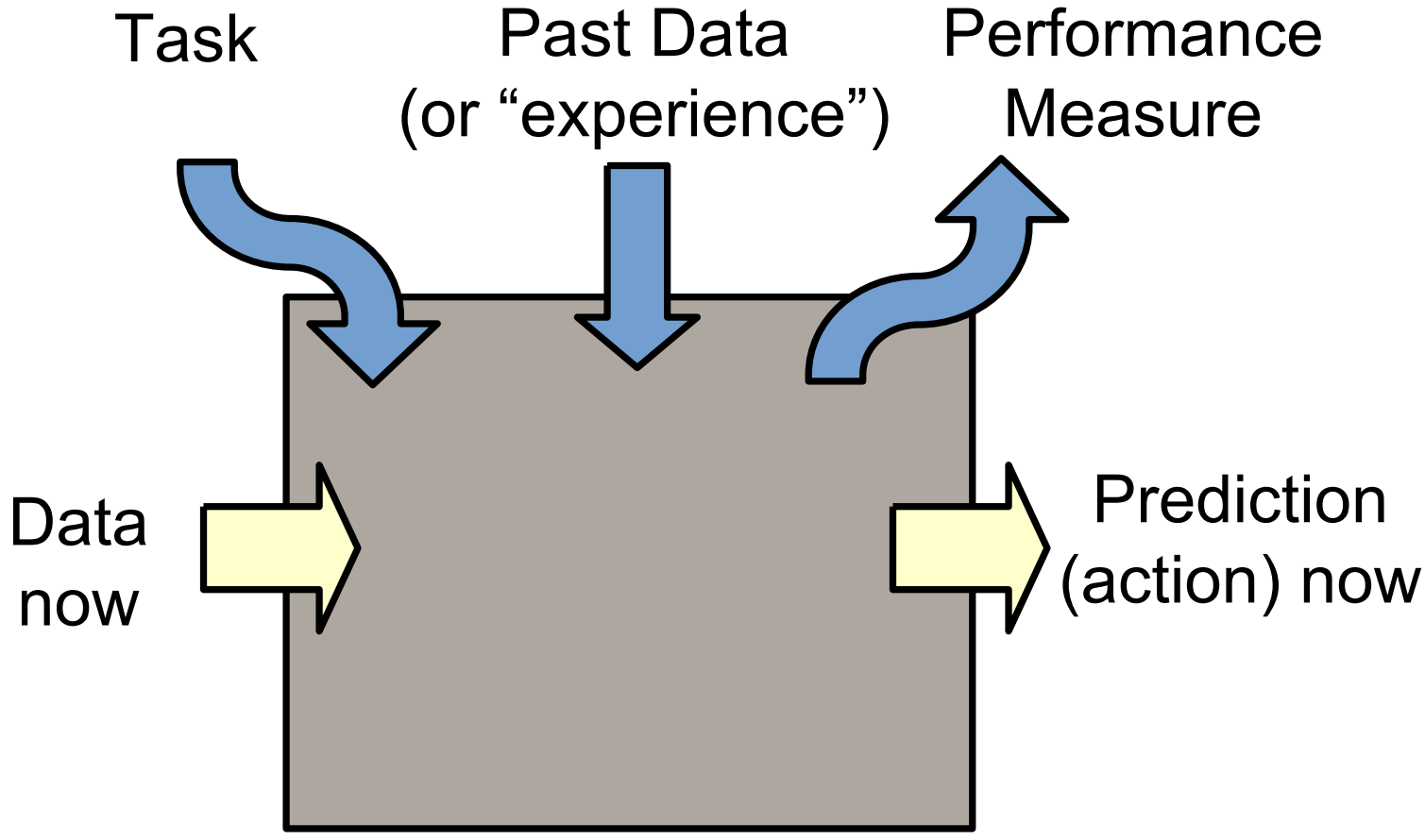
Supervised Learning

- Given set of input, output pairs

$$D = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

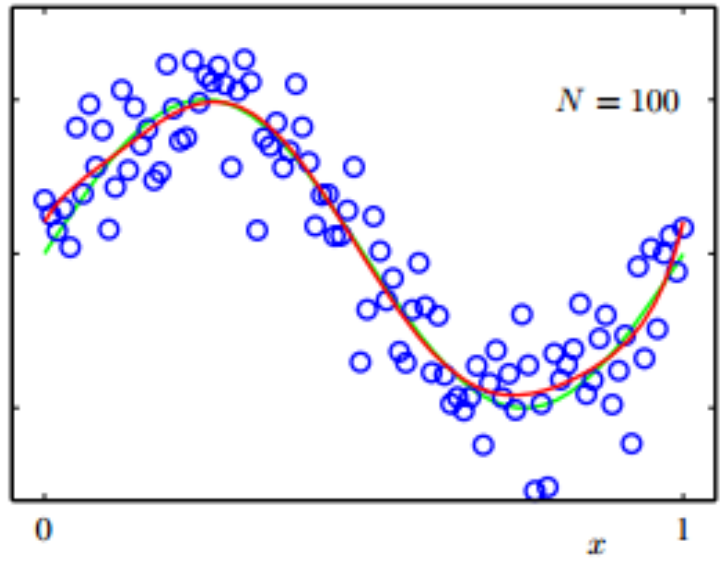
- Learn the "best" $h(\mathbf{x})$ based on D
- Predict $\hat{y}$ for unseen $\mathbf{x}$ based on $h(\mathbf{x})$

What is machine learning?



(1) Regression

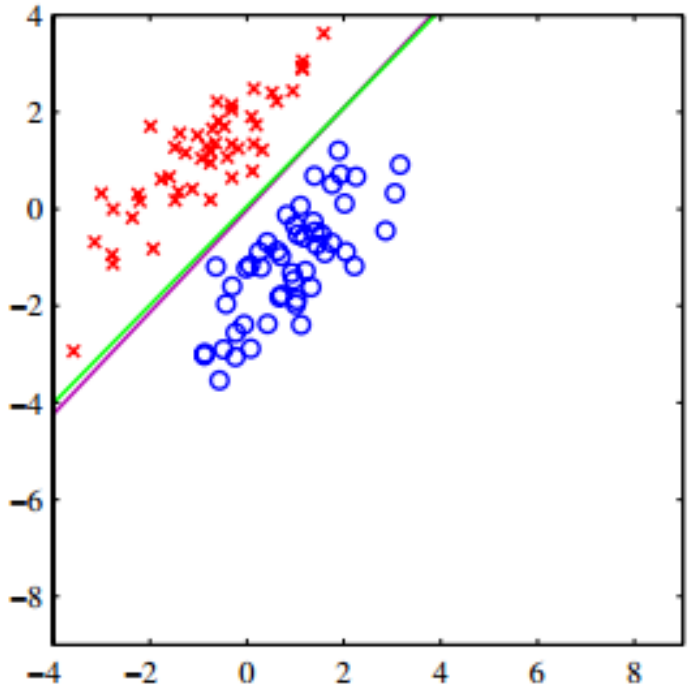
- Output space $\mathcal{Y}$ is real-valued
- Simplest case $\mathcal{Y} = \mathbb{R}$



Polynomial Regression

(2) Classification

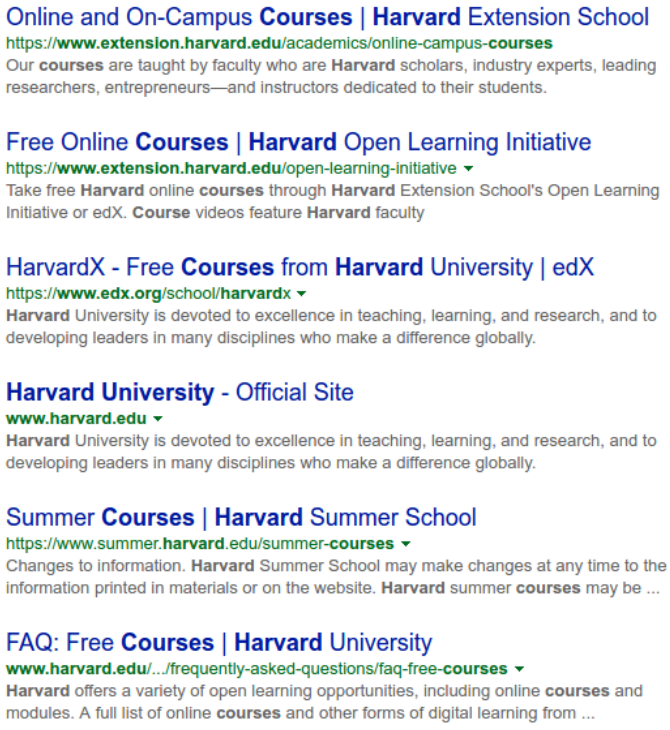
- Output space $\mathcal{Y}$ is a fixed set of classes.
- Simplest case $\mathcal{Y} = \{-1, 1\}$ (red/blue)



Binary Classification

(3) Ordinal Regression / Ranking

- Output space $\mathcal{Y}$ is a ranking of choices.



Search Engine Ranking of Results

(4) Structured Prediction

- Output space $\mathcal{Y}$ is a structure.

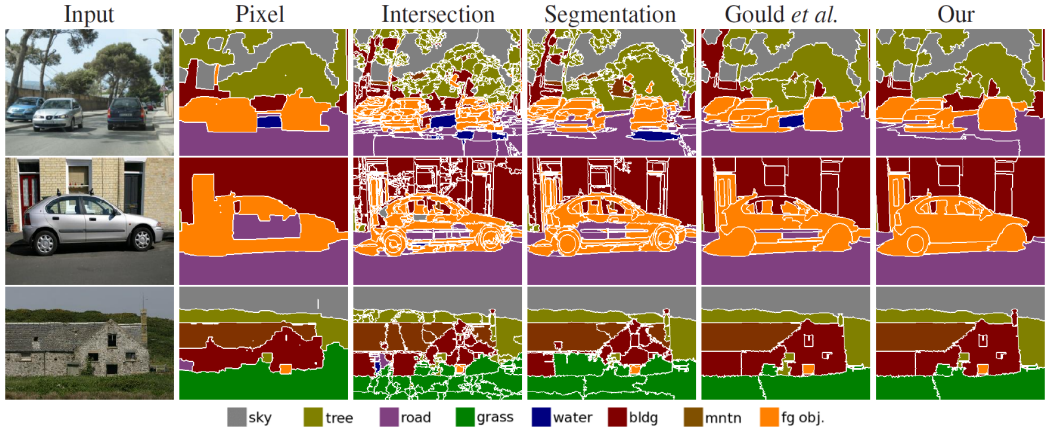


Image Segmentation

Contents

Regression Models

We begin by discussing regression.

$$D = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- $\mathbf{x} \in \mathbb{R}^m$, $\mathcal{Y} = \mathbb{R}$
- Find "best" model h

Consider two approaches:

- Non-Parametric: directly utilize D for predictions
- Parametric: learn parameters of a model from D

Regression Models

We begin by discussing regression.

$$D = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

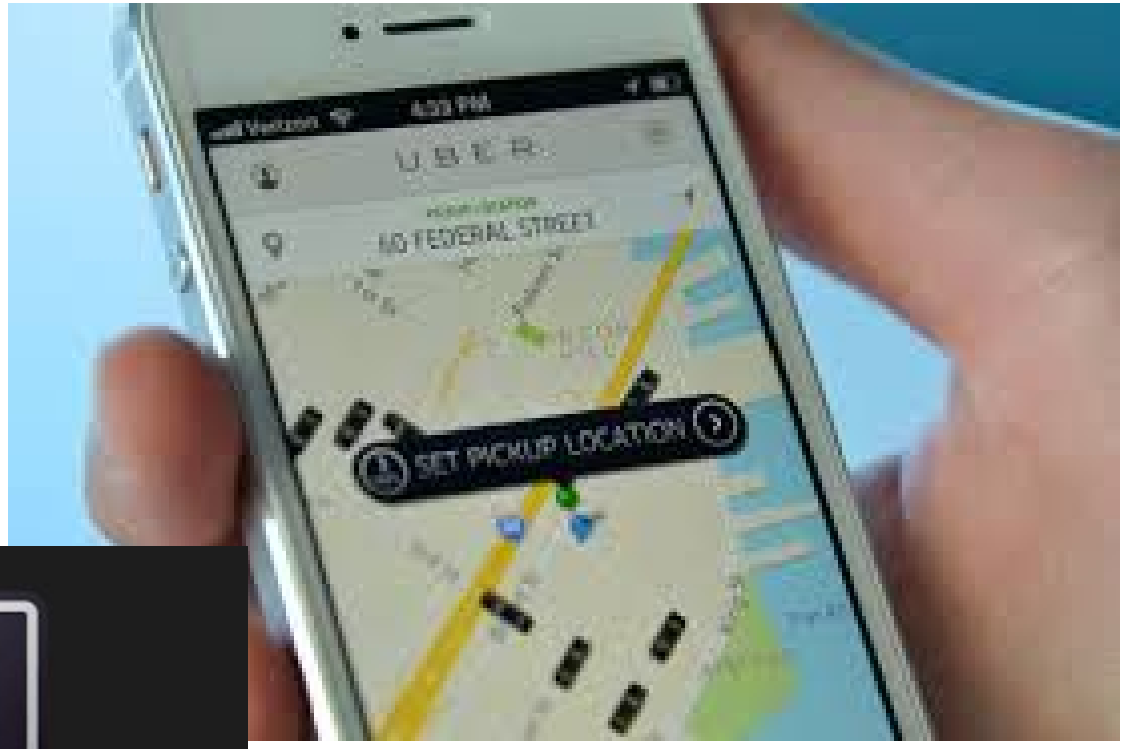
- $\mathbf{x} \in \mathbb{R}^m$, $\mathcal{Y} = \mathbb{R}$
- Find "best" model h

Consider two approaches:

- Non-Parametric: directly utilize D for predictions
- Parametric: learn parameters of a model from D

Example: Uber

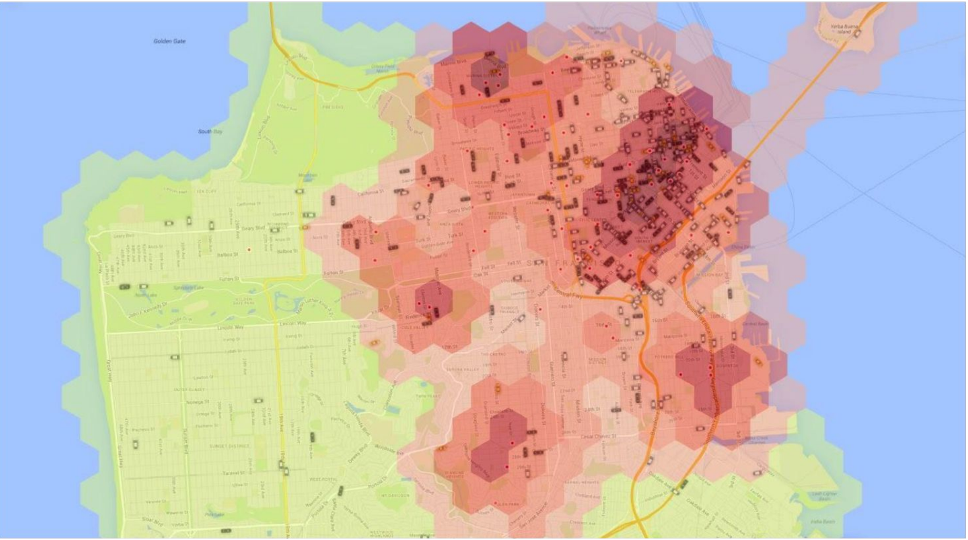
Predictions of travel time, price, supply, demand



UBER

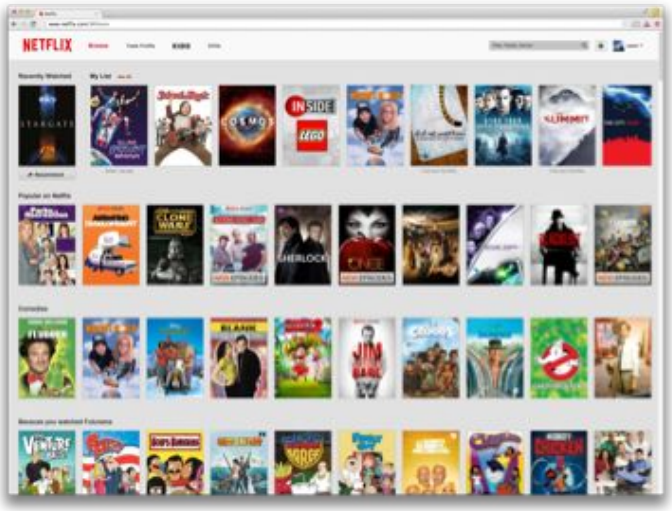

University of California, Los Angeles
Westwood
Westgate Heights
Los Angeles Country Club
Riviera
Brentwood Park
Brentwood
Westgate
Brentwood Country Club
Century City
West Los Angeles
Hillcrest Country Club
Santa Monica
Palms
Culver City
Exposition Blvd
Olympic Blvd
Santa Monica Blvd
Wilshire Blvd
Sunset Blvd
Kirkside Rd
Cresta Dr
Almayo Ave
Prosser Ave
Blythe Ave
Manning Ave
Kelton Ave
Veteran Ave
Sepulveda Blvd
Federal Ave
Mississippi Ave
Nebraska Ave
Harvard St
Princeton St
Yale St
Moreno Ave
Carlyle Ave
Marguerita Ave
Alta Ave
Idaho Ave
California Ave
Arizona Ave
Kansas Ave
Delaware Ave
Pearl St
Ocean Park Blvd
Navy St
Armacost Ave
Butler Ave
Rose Ave
Tabor St
Hughes Ave
Harlis St
Bagley Ave
Cardiff Ave
S Canfield Ave
Olin St
Corsica Dr
Amalfi Dr
Walden Dr
Roxbury Dr

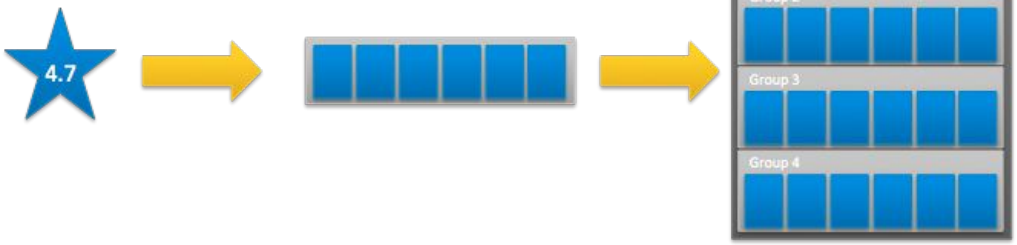
(Keith Chen)



(Keith Chen)

Regression

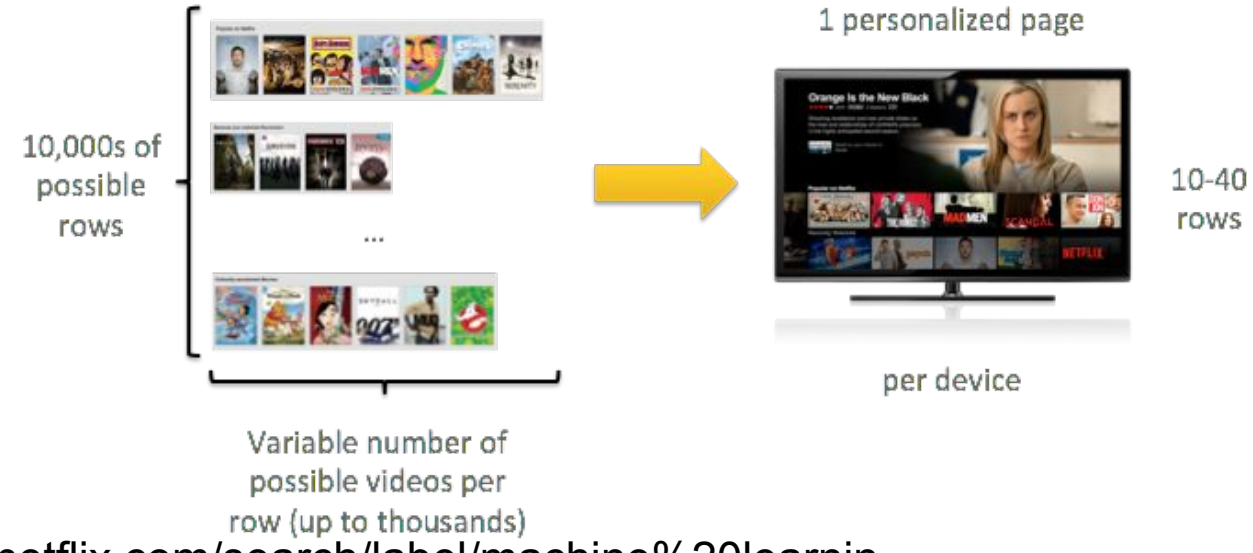




Personalized Rating Prediction

Personalized Ranking

Personalized Page Generation



http://techblog.netflix.com/search/label/machine%20learnin

In Brief: A Non-Parametric Regression Approach

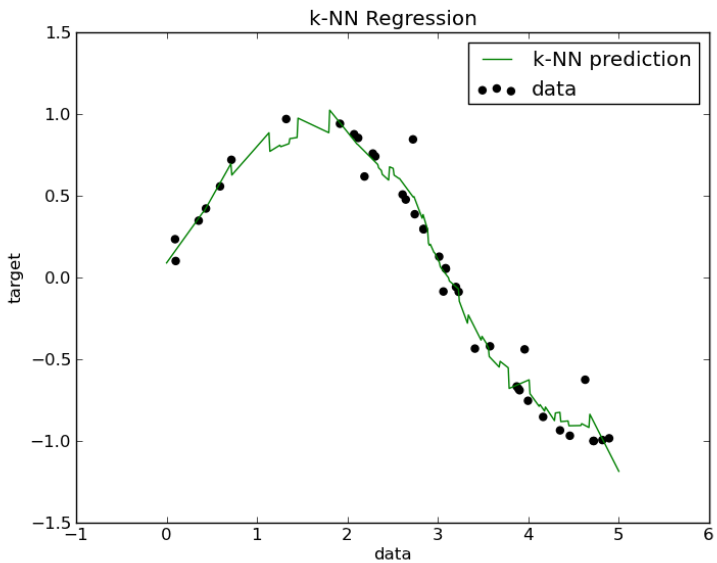
k-Nearest Neighbors learning rule

1. Given new input $\mathbf{x}$
2. Find k closest $\mathbf{x}$ training points $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(k)}, y^{(k)})$
3. Return average output value

$$\hat{y} = h(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} y^{(i)}$$

K-Nearest Neighbors

- No need to "learn" a model
- Requires keeping around training data
- Need to determine size of k
- Curse of dimensionality (later in the class)



kNN Regression

In Contrast: Parametric Models

- Parametric Model; $\hat{y} = h(\mathbf{x}; \mathbf{w})$
- $\mathbf{w}$; model parameters, learned from

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Training Procedure

1. Define a loss criterion $\mathcal{L}$
2. Identify a set of hypotheses h parameterized by $\mathbf{w}$
3. Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$.

In Contrast: Parametric Models

- Parametric Model; $\hat{y} = h(\mathbf{x}; \mathbf{w})$
- $\mathbf{w}$; model parameters, learned from

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Training Procedure

1. Define a loss criterion $\mathcal{L}$
2. Identify a set of hypotheses h parameterized by $\mathbf{w}$
3. Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$.

Linear Regression

- Hypothesis: there is some good linear function of input to find prediction $\hat{\mathbf{y}}$

Assumption:

- Parametric model where $h(\mathbf{x}; \mathbf{w})$ is a linear function of $\mathbf{x}$.
- We select this by choosing $\mathbf{w}$.

Linear Regression: Common

Learn $h(\mathbf{x}; \mathbf{w})$ with

- Parameters: $\mathbf{w} \in \mathbb{R}^{m-1}$, $w_0 \in \mathbb{R}$ bias
- Input: $\mathbf{x}$ where $x_j \in \mathbb{R}$ for $j \in 1, \ldots, (m-1)$ features
- Model Function:

$$\begin{aligned} h(\mathbf{x}; w_0, \mathbf{w}) &= w_0 + w_1 x_1 + \cdots + w_{m-1} x_{m-1} \\ &= \sum_{j=1}^{m} w_j x_j + w_0 \\ &= \mathbf{w}^\top \mathbf{x} + w_0 \end{aligned}$$

Linear Regression: Merged Bias

Trick to let $x_1 = 1$ and $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$, drop w_0

- Parameters: $\mathbf{w} \in \mathbb{R}^m$, w_1 is bias
- Model Function:

$$h(\mathbf{x}; \mathbf{w}) \quad = \quad \mathbf{w}^\top \mathbf{x}$$

- We will mostly use this form.

Performance Measure for Regression

- What makes a good model?
- Squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\left(y_i - h(\mathbf{x}_i;\mathbf{w})\right)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2.\end{aligned}$$

- Training: find minimizer of this loss (least squares)

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Performance Measure for Regression

- What makes a good model?
- Squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - h(\mathbf{x}_i;\mathbf{w})\big)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2.\end{aligned}$$

- Training: find minimizer of this loss (least squares)

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Performance Measure for Regression

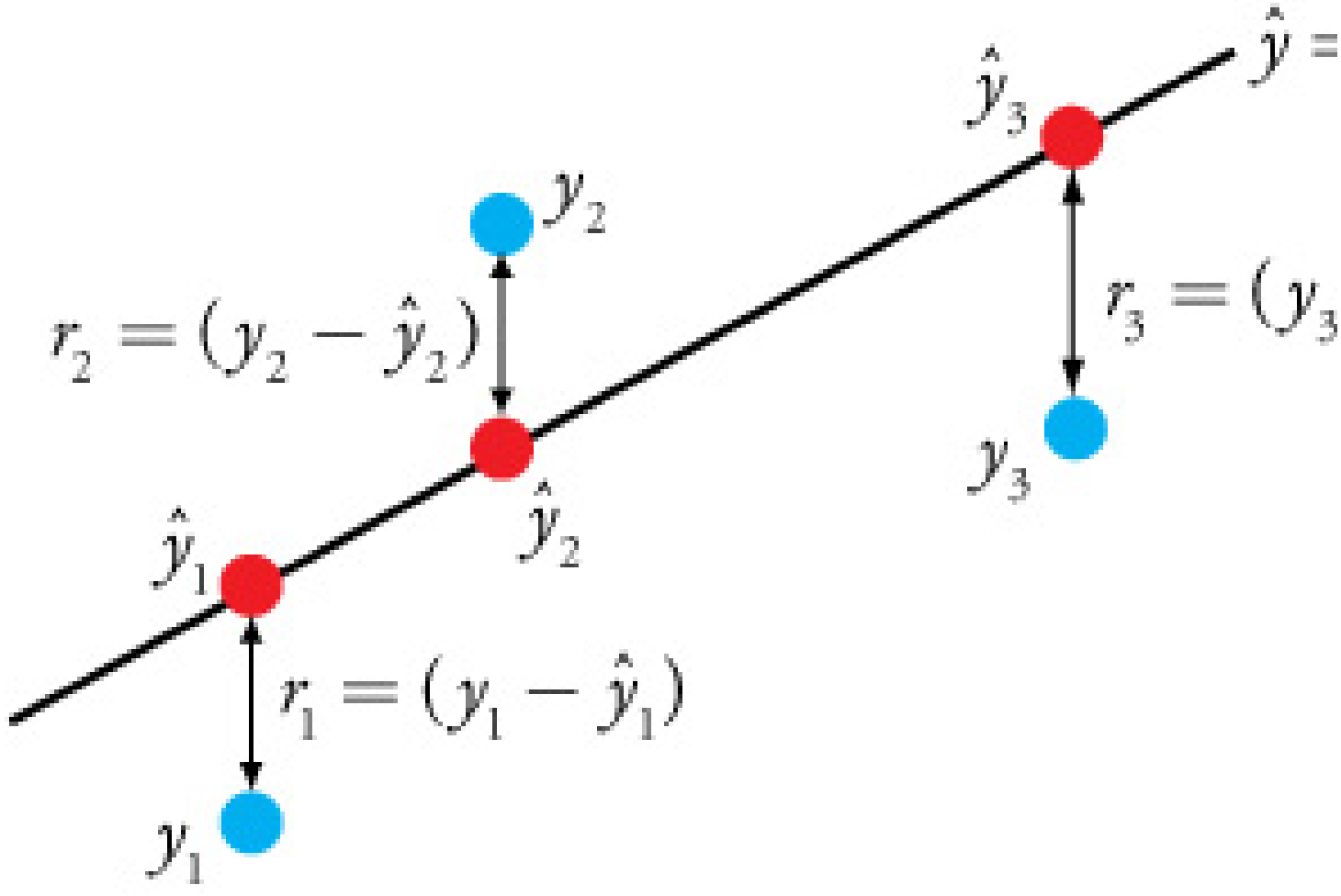
- What makes a good model?
- Squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - h(\mathbf{x}_i;\mathbf{w})\big)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2.\end{aligned}$$

- Training: find minimizer of this loss (least squares)

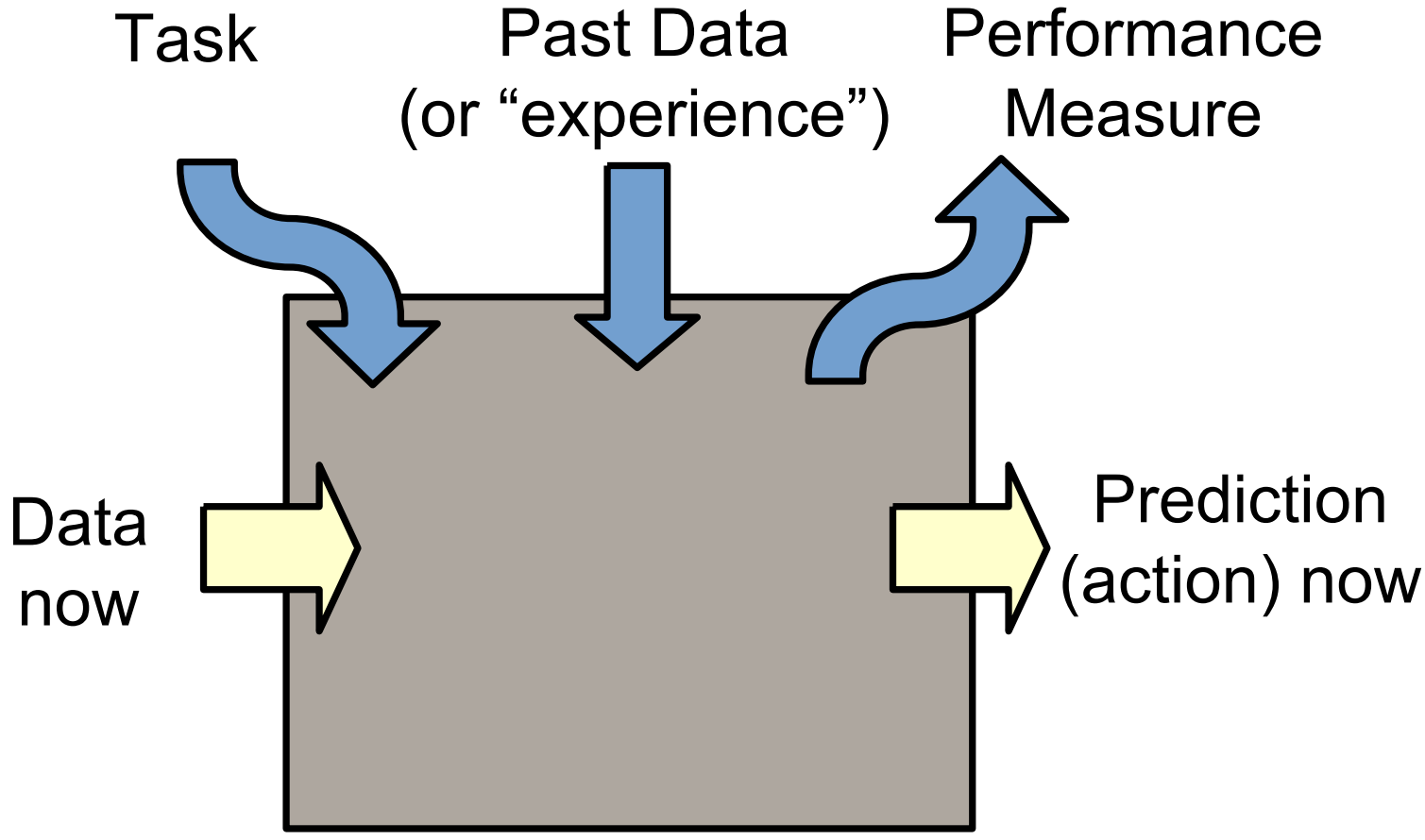
$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Residual Terms



Contributing loss terms for 1D regression.

What is machine learning?



Contents

Machine Learning

Training Procedure

1. Define a loss criterion $\mathcal{L}$
2. Identify a set of hypotheses $h(\mathbf{x}; \mathbf{w})$
3. Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$, i.e.

$$\arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

Learning is done through optimization.

Optimization

Minimizing loss functions can be hard...

Therefore optimization is a central part of machine learning.

- Closed-form solutions (Rare)
- Gradient descent
- Linear and quadratic programming
- Newton-like methods
- Various global optimization ideas and heuristics
- Stochastic optimization
- Lots more...

Optimization

Minimizing loss functions can be hard...

Therefore optimization is a central part of machine learning.

- Closed-form solutions (Rare)
- Gradient descent
- Linear and quadratic programming
- Newton-like methods
- Various global optimization ideas and heuristics
- Stochastic optimization
- Lots more...

Main Tool: Gradients

Typical case (with possibly parameterized g)

$$g(\mathbf{z}) : \mathbb{R}^n \mapsto \mathbb{R}$$

Gradient (vector of partial derivatives)

$$\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial}{\partial z_1} g(\mathbf{z}) \\ \frac{\partial}{\partial z_2} g(\mathbf{z}) \\ \vdots \\ \frac{\partial}{\partial z_n} g(\mathbf{z}) \end{bmatrix}$$

(We will always write as column vectors)

Matrix Calculus Identities

$$\frac{\partial}{\partial \mathbf{z}} \mathbf{z}^\top \mathbf{a} = \frac{\partial}{\partial \mathbf{z}} \mathbf{a}^\top \mathbf{z} = \mathbf{a}$$

$$\frac{\partial}{\partial \mathbf{z}} \mathbf{z}^\top \mathbf{A}\mathbf{z} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{z}$$

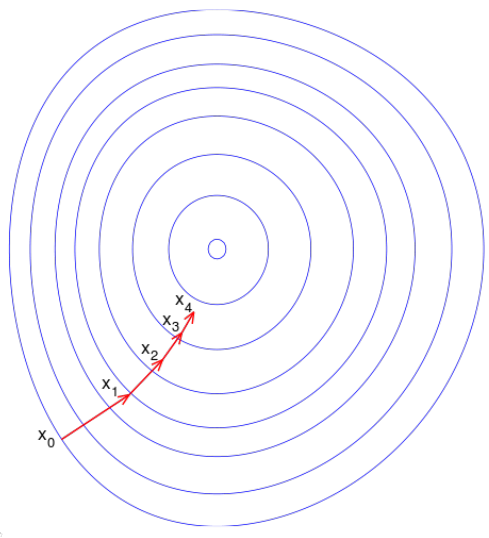
- Many other identities and derivations in Matrix Cookbook
- HW 1 Recommendations.

Preview: Gradient Descent

Minimize loss by repeated gradient steps (when no closed form):

1. Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$
2. Update parameters with rate η

$$\mathbf{w}' \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$



Gradient steps on a simple $m = 2$ loss function.

Back to Linear Regression: Least Squares Loss

Nice case, closed-form solution.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - h(\mathbf{x}_i; \mathbf{w}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2.$$

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i \tag{1}$$

$$= -\sum_{i=1}^{n} y_i \mathbf{x}_i + \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right) \mathbf{w}. \tag{2}$$

Back to Linear Regression: Least Squares Loss

Nice case, closed-form solution.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - h(\mathbf{x}_i; \mathbf{w}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2.$$

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i \tag{1}$$

$$= -\sum_{i=1}^{n} y_i \mathbf{x}_i + \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right) \mathbf{w}. \tag{2}$$

Matrix Version

$$D = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- Inputs (Design matrix): $\mathbf{X} \in \mathbb{R}^{n \times m}$,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix} \tag{3}$$

- Outputs (target vector): $\mathbf{y} \in \mathbb{R}^{n(\times 1)}$,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{4}$$

Least Squares Loss (Matrix Form)

Same as above, but using matrix calculus identities.

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) \\ &= \left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\mathbf{w}}\mathcal{L}(\mathbf{w}) &= \frac{1}{2}\frac{\partial}{\partial\mathbf{w}}\mathbf{y}^\top\mathbf{y} - \frac{\partial}{\partial\mathbf{w}}2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \frac{\partial}{\partial\mathbf{w}}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} \\ &= \frac{1}{2}(-2\mathbf{X}^\top\mathbf{y} + (\mathbf{X}^\top\mathbf{X} + (\mathbf{X}^\top\mathbf{X})^\top)\mathbf{w}) \\ &= -\mathbf{X}^\top\mathbf{y} + \mathbf{X}^\top\mathbf{X}\mathbf{w}\end{aligned}$$

Least Squares Loss (Matrix Form)

Same as above, but using matrix calculus identities.

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) \\ &= \left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\mathbf{w}}\mathcal{L}(\mathbf{w}) &= \frac{1}{2}\frac{\partial}{\partial\mathbf{w}}\mathbf{y}^\top\mathbf{y} - \frac{\partial}{\partial\mathbf{w}}2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \frac{\partial}{\partial\mathbf{w}}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} \\ &= \frac{1}{2}(-2\mathbf{X}^\top\mathbf{y} + (\mathbf{X}^\top\mathbf{X} + (\mathbf{X}^\top\mathbf{X})^\top)\mathbf{w}) \\ &= -\mathbf{X}^\top\mathbf{y} + \mathbf{X}^\top\mathbf{X}\mathbf{w}\end{aligned}$$

Least Squares Matrix

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = -\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Set to 0, and solve for optimal parameters $\mathbf{w}^*$

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

- (FYI: Known as Moore-Penrose pseudo-inverse

$$(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$$

  Generalization of inverse for non-square matrix).

Model 1: Linear Regression

- Input: $\mathbf{x} \in \mathbb{R}^m$
- Output: $y \in \mathbb{R}$
- Parameters: $\mathbf{w} \in \mathbb{R}^m$
- Model:

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$$

- Loss Function:

$$\mathcal{L}_D(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \big(y_i - h(\mathbf{x}_i; \mathbf{w})\big)^2$$

- Optimization

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Demo

Demo of Linear Regression