ITCS 5356: Introduction to Machine Learning

Linear Regression

Razvan C. Bunescu

Department of Computer Science @ CCI

rbunescu@uncc.edu

Supervised Learning

- **Task** = learn an (unknown) function $f : X \rightarrow Y$ that maps input instances $\mathbf{x} \in X$ to output targets $y = f(\mathbf{x}) \in Y$:
  - **Classification**:
    - The output $y \in Y$ is one of a finite set of discrete categories.
  - **Regression**:
    - The output $y \in Y$ is continuous, or has a continuous component.

- Target function $f(\mathbf{x})$ is known (only) through (noisy) set of training examples:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$$

Supervised Learning

- **Task** = learn an (unknown) function $f : X \rightarrow Y$ that maps input instances $\mathbf{x} \in X$ to output targets $y = f(\mathbf{x}) \in Y$:
  - function $f(\mathbf{x})$ is known (only) through (noisy) set of training examples:
    - Training data: $(\mathbf{x}_1,y_1), (\mathbf{x}_2,y_2), \ldots (\mathbf{x}_n,y_n)$

- **Task** = build a function $h(\mathbf{x})$ such that:
  - *h* matches *f* well on the *training data*:

    => *h* is able to fit data that it has seen.
  - *h* also matches target *f* well on *test data*:

    => *h* is able to generalize to unseen data.

Parametric Approaches to Supervised Learning

- **Task** = build a function $h(\mathbf{x})$ such that:
  - h matches f well on the training data:
    => h is able to fit data that it has seen.
  - h also matches f well on test data:
    => h is able to generalize to unseen data.

- **Task** = choose h from a "nice" *class of functions* that depend on a vector of parameters $\mathbf{w}$:
  - $h(\mathbf{x}) \equiv h_{\mathbf{w}}(\mathbf{x}) \equiv h(\mathbf{w},\mathbf{x})$
  - **what classes of functions are "nice"?**
    - *linear* ⊂ *convex* ⊂ *continuous* ⊂ *differentiable* ⊂ ...

Linear Regression

1. **(Simple) Linear Regression**
   – **House price** prediction as a function of *floor size*.

2. **Multiple Linear Regression**
   – **House price** prediction as a function of *floor size*, *age*, *bedrooms*.
   – Normal equations.

House Price Prediction

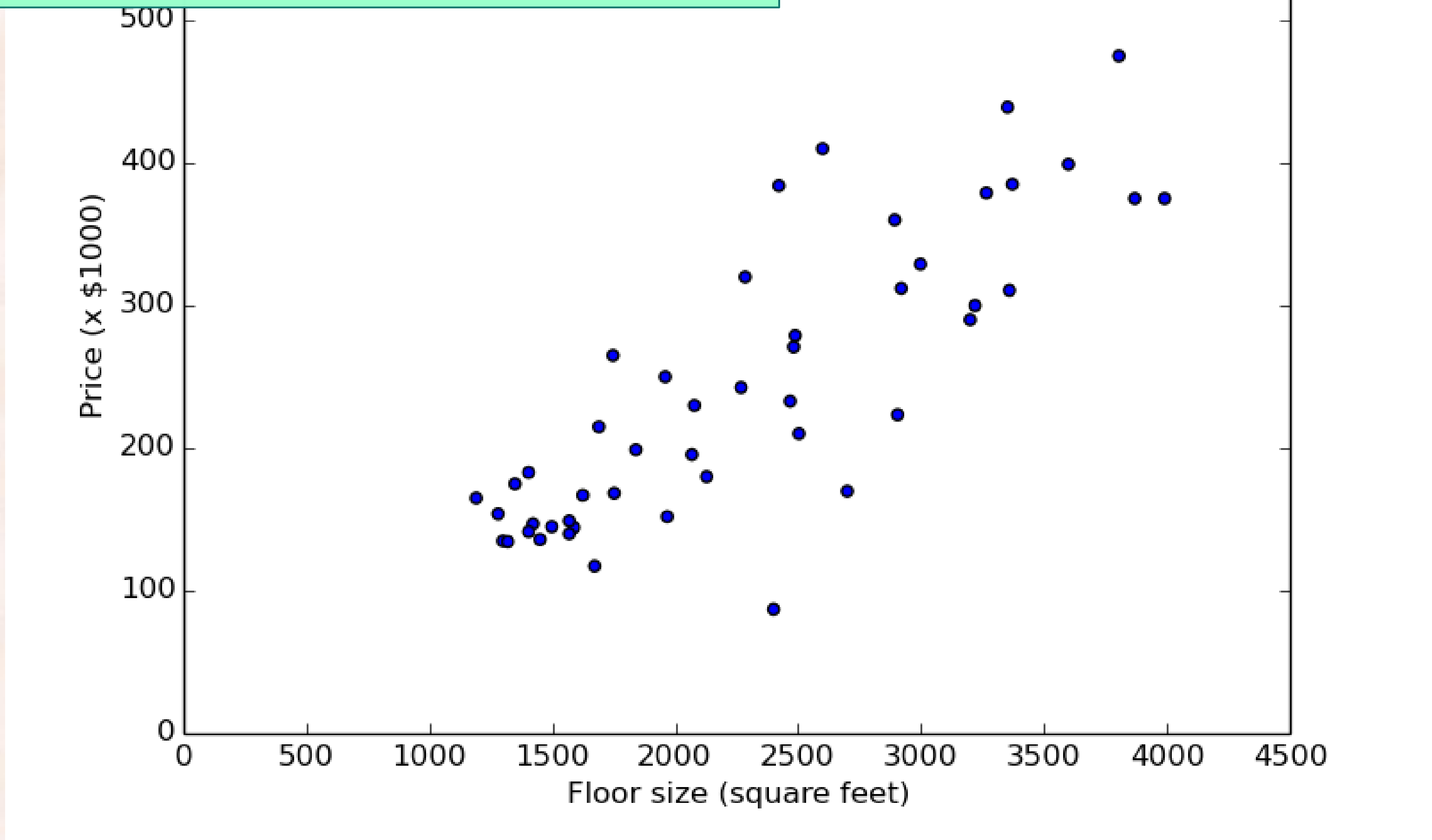
- Given the floor *size* in square feet, predict the selling price:
  - x is the size, y is the price
  - Need to learn a function h such that $\hat{y} = h(x) \approx f(x) = y$.

- Is this classification or regression?
  - **Regression**, because price is real valued.
    - and there are many possible prices.
  - (Simple) linear regression, because one input value.
    - Would a problem with only two labels 0.5 and 1.0 still be regression?

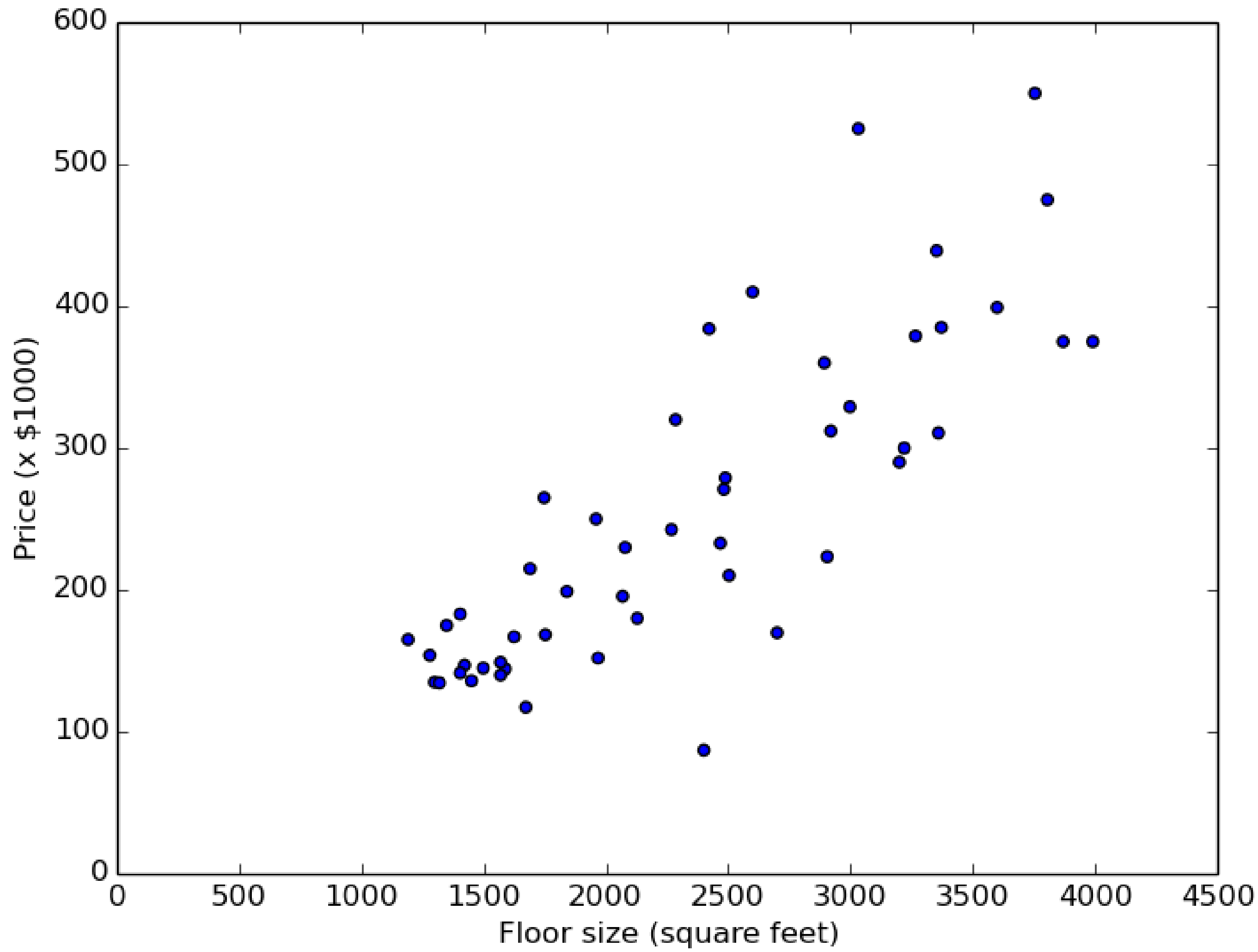
House Prices in Athens

50 houses, randomly selected from the 106 houses or townhomes:

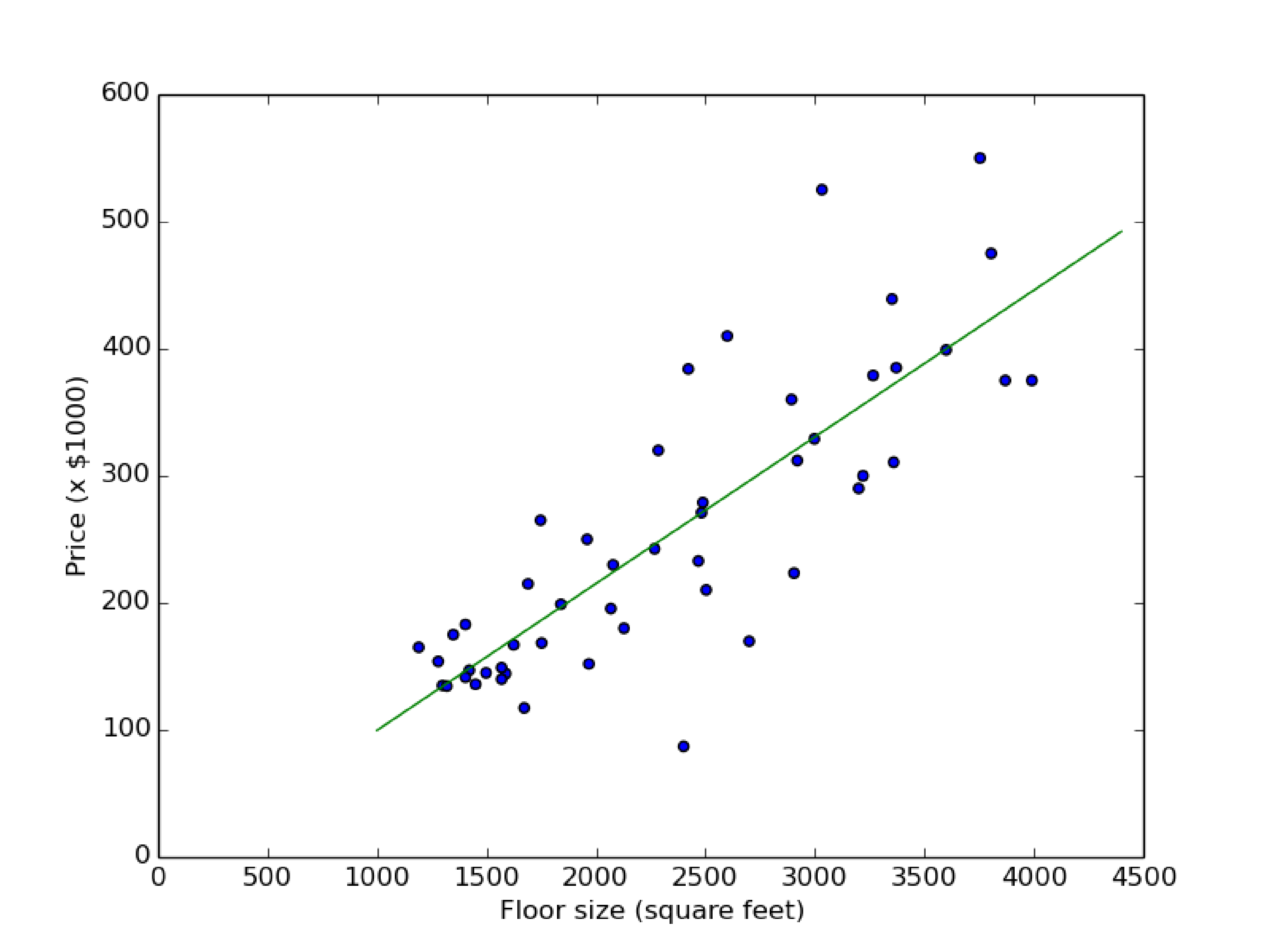
- sold recently in Athens, OH.
- built 1990 or later.



House Prices in Athens



House Prices in Athens

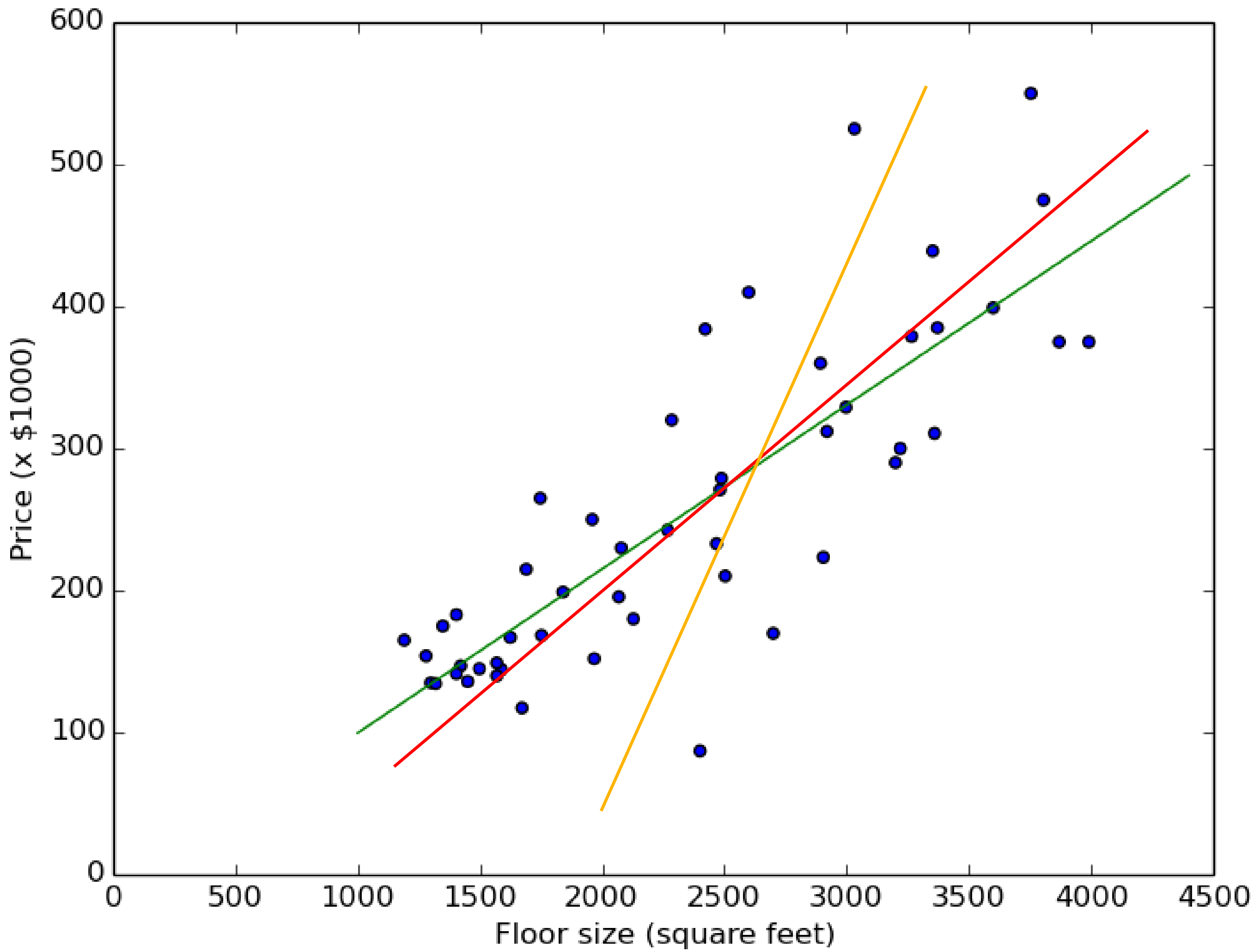


Linear Regression

- Use a linear function approximation:
  - $\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = [w_0, w_1]^T[1, x] = w_1x+w_0$.
    - w_0 is the intercept (or the bias term).
    - w_1 controls the slope.

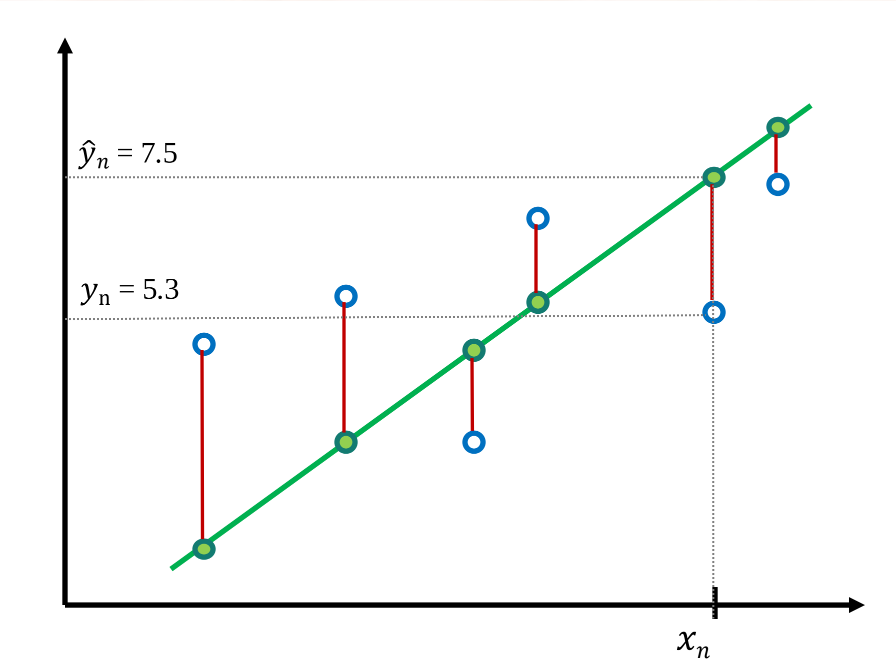
- How do we find the best line?
  - What do we mean by the "best"?
    - How do we quantify how good a line is?
      - Quantify the error that a line makes.
        - » How?

Which line is better?



Sum-of-Squares Error Function

$\hat{y}_n = h_{\mathbf{w}}(\mathbf{x}_n) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n = [w_0, w_1]\cdot[1, x_n] = w_1 x_n + w_0$



$J(\mathbf{w})$ is the **objective function**:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

$$= \frac{1}{2N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

Learning means find **w** that minimizes the objective function, i.e. the **cost**:

$$\hat{\boldsymbol{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$$

Linear Regression

- Use a linear function approximation:
  - $\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{T}\mathbf{x} = [w_0, w_1]^{T}[1, x] = w_1 x + w_0$.
    - w_0 is the intercept (or the bias term).
    - w_1 controls the slope.

- Learning = optimization:
  - Find **w** that obtains the best fit on the training data, i.e. find **w** that minimizes the **sum of square errors**:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 = \frac{1}{2N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} J(\mathbf{w})$$

Minimizing Sum-of-Squares Error

- Minimizing the Sum-of-Squares error function:

why squared?

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_\mathrm{n} - y_n)^2$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$$

- How do we find **w** that minimizes $J(\mathbf{w})$?
  - How do we find the minimum of a function that is **convex** and **differentiable**?
    - Find the parameters **w** that make the gradient equal $\nabla J(\mathbf{w})$ to 0.

What is the **gradient** of a function?

Mathematical Intermission: Differentiation

In class: Find solution by solving $\nabla J(\mathbf{w}) = \mathbf{0}$

Minimizing Sum-of-Squares Error

- Sum-of-Squares error function:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N} (h_{\mathbf{w}}(\mathbf{x}_{\mathrm{n}}) - y_n)^2$$

- How do we find **w*** that minimizes $E(\mathbf{w})$?

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$$

- Least Square solution is found by solving a system of 2 linear equations:

$$w_0 N + w_1 \sum_{n=1}^{N} x_n = \sum_{n=1}^{N} y_n$$

$$w_0 \sum_{n=1}^{N} x_n + w_1 \sum_{n=1}^{N} x_n^2 = \sum_{n=1}^{N} y_n\, x_n$$

Multiple Linear Regression

- Q: What if one feature is insufficient for good performance?
  - Example: house prices depend not only on *floor size*, but also number of *bedrooms*, *age*, *location*, …

- A: Use **Multiple Linear Regression**.

  $\mathbf{x} = [x_0, x_1, \ldots, x_M]^T$

  $\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

- Training examples: $(\mathbf{x}^{(1)}, y_1), (\mathbf{x}^{(2)}, y_2), \ldots (\mathbf{x}^{(N)}, y_N)$

Multiple Linear Regression

- **Learning** = minimize the Sum-of-Squares error function:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) \qquad J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

$$= \frac{1}{2N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_\mathrm{n} - y_n)^2$$

- How do we find the minimum of a function that is **convex** and **differentiable**?

Homework: Solve $\nabla J(\mathbf{w}) = \mathbf{0}$

Multiple Linear Regression

- **Learning** = minimize the Sum-of-Squares error function:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) \qquad J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(\mathbf{w}^T\mathbf{x}^{(n)} - y_n\right)^2$$

- Computing the gradient $\nabla J(\mathbf{w})$ and setting it to zero:

$$\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(n)} - t_n\right)\mathbf{x}^{(n)} = 0$$

- Solving for **w** yields $\mathbf{w} = \underbrace{\left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}}\mathbf{y}$
  - Prove it (homework).

The Moore-Penrose pseudo-inverse of X.

Normal Equations

- Solution is $\mathbf{w} = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}\mathbf{y}$

- X is the data matrix, or the **design matrix**:

$$X = \begin{pmatrix} \mathbf{x}^{(1)^{\mathrm{T}}} \\ \mathbf{x}^{(2)^{\mathrm{T}}} \\ \ldots \\ \ldots \\ \mathbf{x}^{(N)^{\mathrm{T}}} \end{pmatrix} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \ldots & x_M^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \ldots & x_M^{(2)} \\ & & \ldots & \\ & & \ldots & \\ x_0^{(N)} & x_1^{(N)} & \ldots & x_M^{(N)} \end{pmatrix}$$

For poly fit:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^M \\ 1 & x_2 & x_2^2 & \ldots & x_2^M \\ & & \ldots & & \\ & & \ldots & & \\ 1 & x_N & x_N^2 & \ldots & x_N^M \end{pmatrix}$$

- $\mathbf{y} = [y_1, y_2, \ldots, y_{\mathrm{N}}]^{\mathrm{T}}$ is the vector of labels.

Evaluation Measures

- **Root Mean Square Error (RMSE)**:

$$RMSE(\mathbf{w}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_\mathrm{n} - y_n)^2}$$

- **Mean Absolute Error (MAE)**:

$$MAE(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}|\hat{y}_n - y_n| = \frac{1}{N}\sum_{n=1}^{N}|\mathbf{w}^T\mathbf{x}_\mathrm{n} - y_n|$$