10-315 Notes
Optimization and Linear Regression

Carnegie Mellon University
Machine Learning Department

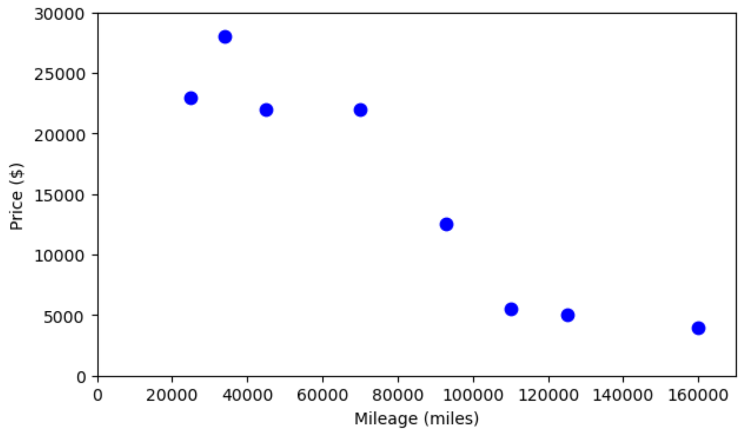
There is an incredible amount of ML that we can learn from just looking at $y = mx + b$. Well, of course, we also have to look at data as well as this linear model. But, seriously, mastering the details behind fitting this linear model to data will give us the backbone for so much in this course and machine learning, including all the fanciest neural nets.

Contents

1 Selling my car

Suppose I'm trying to sell my sweet Pontiac Vibe to get some extra spending cash. I looked through some sites that sell cars online and grabbed a few data points, specifically the mileage and price for eight cars. I want to create an ML model (a **hypothesis function** h) that takes a car's mileage and predicts the selling price, $\hat{y} = h(x)$.

Demo: interactive_regression.ipynb

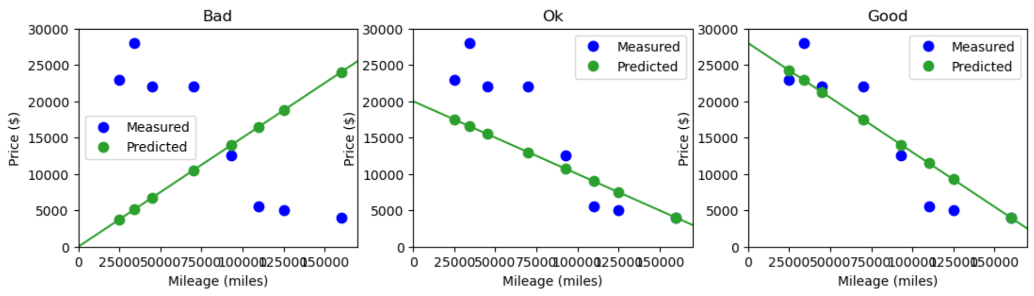


1.1 Linear model

It looks like it might be reasonable to model this data with a linear function (Or affine function; we'll talk more about linear vs affine later). Here is a linear hypothesis function with input variable x and parameters w and b, $h(x; w, b) = wx + b$. For now, these are all real-valued scalars. The parameter w represents the slope of the line. We often use w in machine learning; this is the same as the m that you may have seen used for slope in algebra.

Notation alert: sometimes we'll separate arguments of a function with a semi-colon to highlight the input variables from parameters such as in $h(x; w, b)$.

We get to change values for **parameters** m and b to find the best line! Below are plots for three pairs of parameters: 'Bad', 'Ok', and 'Good'



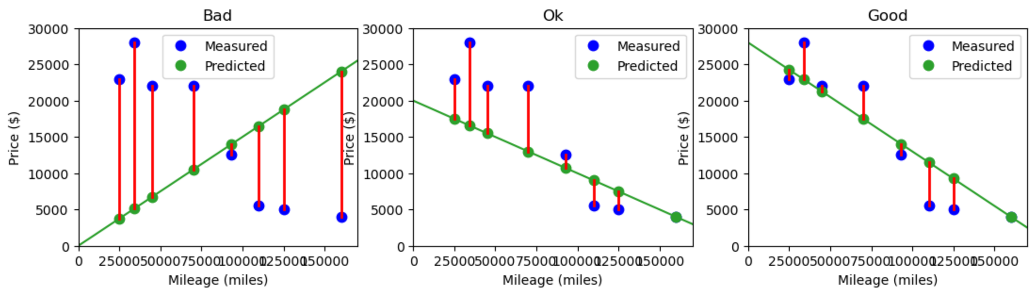
The "ok" values for parameters w and b are certainly better than the "bad" ones, and "pretty good" visually seems to be better than "ok". But, we can't rely on eyeballing ML models to see if they are any good.

To allow a machine to find the best parameter settings for us, we need a quantitative to compare which model is best. A **performance measure**!!

1.2 Performance measure for regression

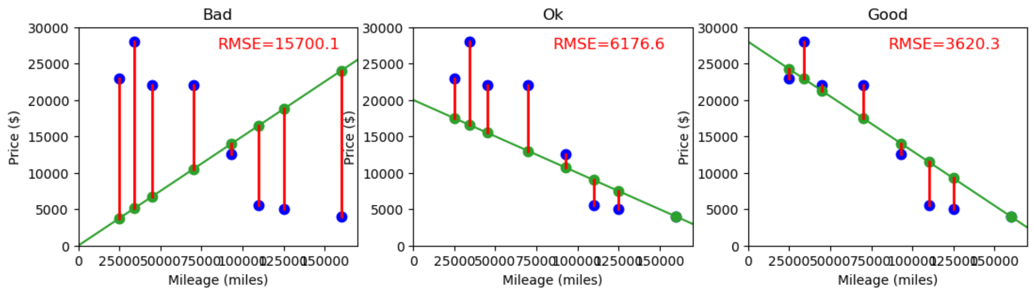
In regression, **error** is the difference between the predicted output and the measured output in our data. We do this for each data point:

$$\hat{y}^{(i)} = mx^{(i)} + b \tag{1}$$
$$error^{(i)} = y^{(i)} - \hat{y}^{(i)} \tag{2}$$

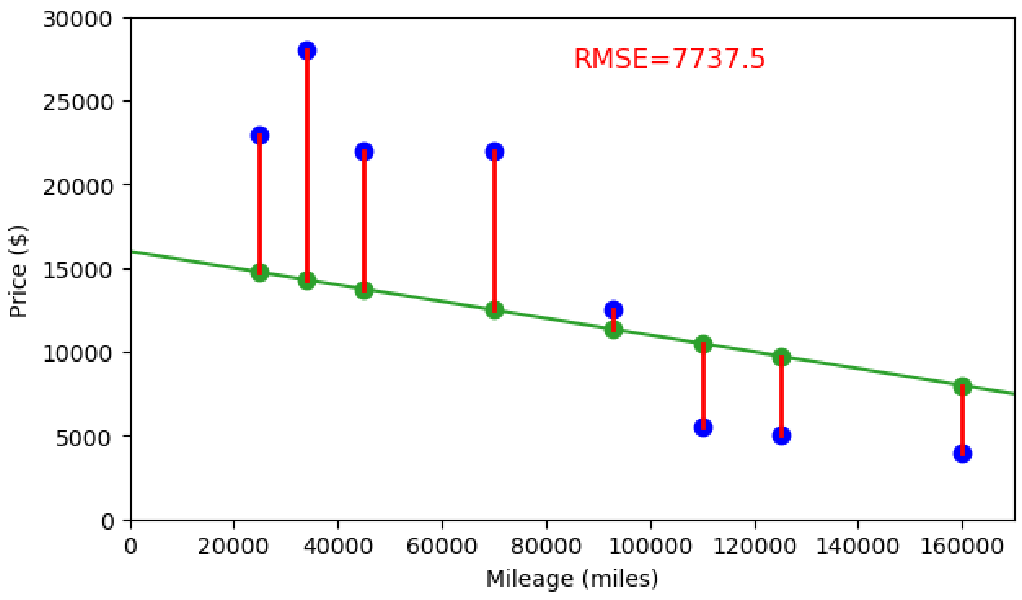


But now we have $N = 8$ different error numbers for each model. We really need to get down to one number that represents the performance for each model.

- Attempt 1 (fail): $\sum_{i=1}^{N} error^{(i)}$, **sum of errors**
  - Because the error values can be positive or negative, summing large positive and large negative will cancel out, making the "bad" model, left, actually have a very low sum.
- Attempt 2 (ok): $\frac{1}{N}\sum_{i=1}^{N} \left|error^{(i)}\right|$, **mean absolute error**
- Attempt 3 (preferred): $\frac{1}{N}\sum_{i=1}^{N} \left(error^{(i)}\right)^2$, **mean squared error (MSE)**
  - We'll learn more about why this is preferred over mean absolute error in a few weeks
  - Note: Sometimes we use root mean squared error (RMSE), the square root of MSE, to make this number match the scale of the output range.



Now we can say that the "good" model is quantitatively better than the others, with a root mean squared error of 3,620.3.



1.3 Questions

Some questions you could certainly be asking:

- How do we find the parameters (e.g. w and b) that give the *best* performance measure?

- What if my data doesn't perfectly fit in a line (e.g. my data is non-linear)?

- What if my data has more than one input feature (e.g. instead of just mileage, you want to consider mileage and fuel tank size)?

We'll get to these. But, first, let's talk through:

1. Math Background: Linear (and affine) functions and geometry
2. ML Problem Formulation and Optimization Notation
3. Optimization for Linear Regression

2 Math background

2.1 Linear (and affine) functions and geometry

2.1.1 Vocab / definitions:

linear combination: If $\mathbf{x}_1, ..., \mathbf{x}_K$ are vectors in $\mathbb{R}^M$, and $c_1, ..., c_K$ are scalars in $\mathbb{R}$, then the resulting vector, $c_1\mathbf{x}_1 + ... + c_K\mathbf{x}_K \in \mathbb{R}^M$, is called a linear combination of the vectors $\mathbf{x}_1, ..., \mathbf{x}_M$ and $c_1, ..., c_K$ are called the coefficients of the linear combination.

linear function: a function $f : \mathbb{R}^M \to \mathbb{R}$ is a linear function, if for all vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^M$ and scalars α and $\beta \in \mathbb{R}$, the property $f(\alpha\mathbf{x} + \beta\mathbf{v}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{v})$ holds. If a function f is linear, this extends to linear combinations of any number of vectors, and not just linear combinations of two vectors.

affine function: (just saying linear sometimes includes affine) A linear function plus a constant is called an affine function.

2.1.2 Linear vs affine

What linear usually means depends on the domain!

Linear algebra

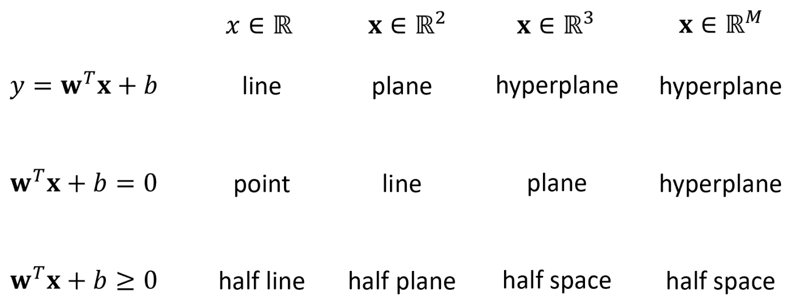
- Linear usually means strictly linear
- Linear: $\mathbf{y} = A\mathbf{x}$
- Affine: $\mathbf{y} = A\mathbf{x} + \mathbf{b}$

Geometry and algebra

- Linear usually means affine
- Line: $y = wx + b$
- Plane/hyperplane: $y = \mathbf{w}^T\mathbf{x} + b$

Vocab for linear (affine) geometry in various dimensions:

Note: $\mathbf{x}$ is the input variable vector



	$x \in \mathbb{R}$	$\mathbf{x} \in \mathbb{R}^2$	$\mathbf{x} \in \mathbb{R}^3$	$\mathbf{x} \in \mathbb{R}^M$
$y = \mathbf{w}^T\mathbf{x} + b$	line	plane	hyperplane	hyperplane
$\mathbf{w}^T\mathbf{x} + b = 0$	point	line	plane	hyperplane
$\mathbf{w}^T\mathbf{x} + b \geq 0$	half line	half plane	half space	half space

Machine learning

- Linear usually means affine
- Pay close attention to what the variables are!

In machine learning, the parameters are usually the variables, and the data is considered constant. This makes us take a different look at our model equations.

$$f(x) = wx + b \text{ is affine when } x \text{ is the variable} \tag{3}$$
$$f(b, w) = wx + b \text{ is actually linear in the parameter variables } w,\ b \tag{4}$$

Similarly,

$$f(x) = w_2x^2 + w_1x + b \text{ is quadratic when } x \text{ is the variable} \tag{5}$$
$$f(b, w_1, w_2) = w_2x^2 + w_1x + b \text{ is linear in the parameter variables } w_1,\ w_2, \text{ and } b!! \tag{6}$$

We'll often use linear algebra to make these linear models more explicit and concise, where we put all of our parameter variables in a vector, $\boldsymbol{\theta}$.

For $\boldsymbol{\theta} = \begin{bmatrix} b \\ w \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}$:

$$\begin{aligned} y &= wx + b && (7) \\ &= \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix} && (8) \\ &= \mathbf{x}^T \boldsymbol{\theta} && (9) \end{aligned}$$

For $\boldsymbol{\theta} = \begin{bmatrix} b \\ w_1 \\ w_2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$

$$\begin{aligned} y &= w_2x^2 + w_1x + b && (10) \\ &= \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ w_2 \end{bmatrix} && (11) \\ &= \mathbf{x}^T \boldsymbol{\theta} && (12) \end{aligned}$$

Again, these are linear functions because we are treating the data vectors, $\mathbf{x}$ as constant.

3 ML problem formulation and notation

Ok, we're almost back to linear regression. We just have a few more machine learning formalities to define before we can set up up our linear regression optimization.

3.1 Loss

In machine learning, **loss** is a non-negative, scalar measure of how different a predicted output, $\hat{y}$, is from the corresponding true output, y (or the measured output in our dataset). The loss function, denoted $\ell(y, \hat{y})$, depends on both the type of machine learning problem (e.g. classification or regression) and the specific application. Note: the term **error** is also dependent on the type of problem and is very similar to loss as we'll see below.

3.1.1 Classification loss functions

For classification, **classification error** is when the predicted label doesn't match the measured label.

Zero-one loss is a common loss that simply returns zeros when the label match and one when they don't. $\ell(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$, where the indicator function, $\mathbb{1}(z)$, returns one when z is true and zero otherwise.

A more general classification loss function could put more weight on different types of label mismatches. For example, in medicine, there could be significant differences in the severity of making false positive versus false negative errors, where **false positive errors** are when the prediction is ill but the true output is healthy and **false negative errors** are when the prediction is healthy but the true output is ill. This leads to an asymmetric loss function, as in the following loss table:

$$\ell(y,\hat{y}) = \begin{array}{r|cc} & \hat{y}=0 & \hat{y}=1 \\ \hline y=0 & 0 & 2 \\ y=1 & 10 & 0 \end{array}$$

Note in this table that our loss function defines a false negative error as a larger error than a false positive error. When training machine learning models, this type of asymmetric loss can be used to encourage the model to be more careful about making false predictions.

3.1.2 Regression loss functions

Squared error is commonly used for regression problems as the output values are continuous and the size of the difference matters. The squared error loss function is $\ell(y, \hat{y}) = (y - \hat{y})^2$. Note: you may be correctly guessing that this is leading to our use of mean squared error.

3.2 Risk for ML models

3.2.1 Risk

The **risk** for ML hypothesis function h, $R(h)$, is the expected value of loss over the distribution of input and output data represented by input and output random variable X and Y:

$$R(h) = \mathbb{E}_{X,Y}[\ell(Y, h(X))]$$

.

However, we basically never have sufficient models for the real world. This is because it is very difficult to know the true joint probability distribution of input and output variables, so we do our best with empirical data samples, $\mathcal{D}$.

3.2.2 Empirical risk

However, we basically never have sufficient models for the real world, so we do our best with empirical data samples, $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$. For a given dataset, $\mathcal{D}$, $\hat{R}(h)$ is the empirical risk for the hypothesis function h:

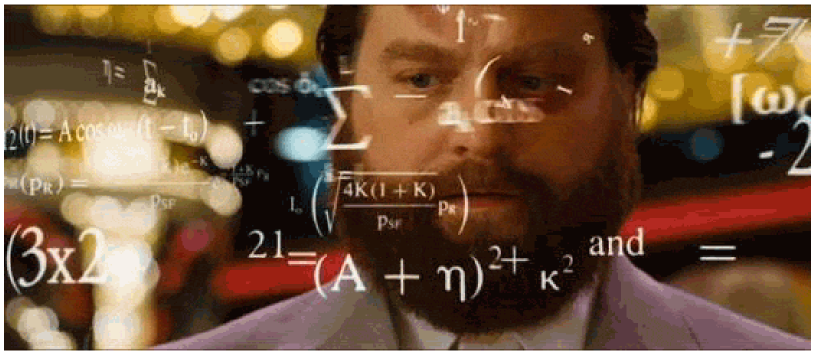
$$\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(y^{(i)}, h\left(x^{(i)}\right)\right)$$

3.2.3 Empirical risk minimization

Optimization! Find the hypothesis function (ML model) that minimizes the empirical risk for a given dataset:

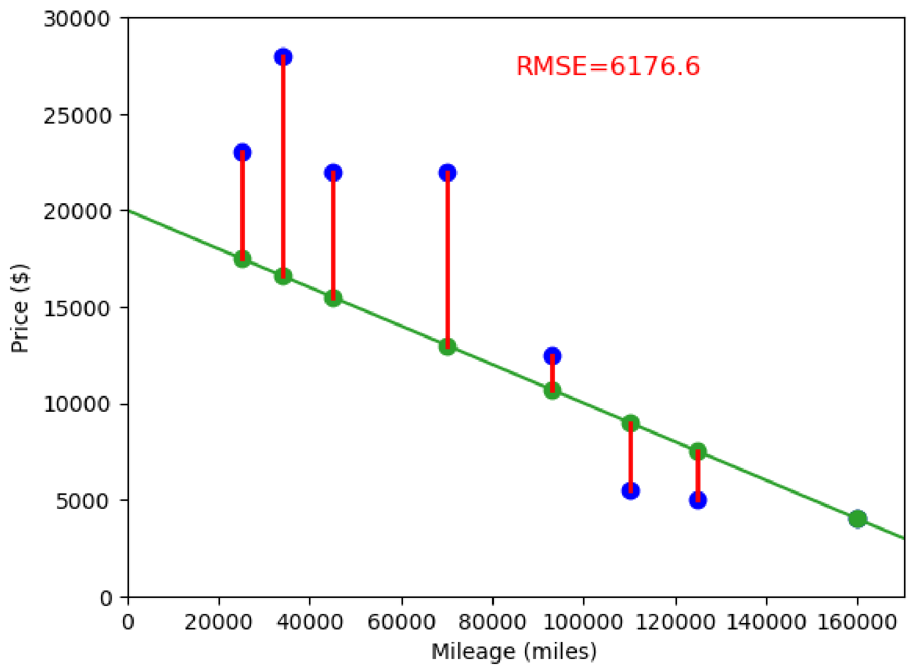
$$h^* = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\ \hat{R}(h)$$

where $\mathcal{H}$ is the set of hypothesis functions that we are considering. Often $\operatorname{argmin}_{h\in\mathcal{H}}$ is replaced by $\operatorname{argmin}_{\theta}$ when we fix the structure of the hypothesis function (e.g. a specific structure we might consider is the linear model structure: $h(x; w, b) = wx + b$) and then search over parameters values, θ, (e.g. w and b) to explore the space of linear hypothesis models, $\mathcal{H}$.



Holy cow, so much notation. This will be so much better when we apply this to our simple linear regression problem!

4 Optimization for linear regression



```
Training data:
x: [ 25000  34000  45000  70000  93000 110000 125000 160000]
y: [23000 28000 22000 22000 12500  5500  5000  4000]
```

4.1 Empirical risk minimization for linear regression model

4.1.1 Setup

- Input: $x \in \mathbb{R}$
- Output: $y \in \mathbb{R}$
- Data:

$$\mathcal{D} = \left\{ \left(x^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N=8} \tag{13}$$

$$=(25000, 23000), (34000, 28000), (45000, 22000), (70000, 22000), \tag{14}$$

$$(93000, 12500), (110000, 5500), (125000, 5000), (160000, 4000) \tag{15}$$

- Hypothesis function stucture: $h(x; w, b) = wx + b$
- Parameters: w, b
- Loss function: Squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$

4.1.2 Optimization formulation

Empricial risk:

$$\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(y^{(i)}, h\left(x^{(i)}\right)\right) \tag{16}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - h\left(x^{(i)}\right)\right)^2 \quad \text{<— Mean squared error! (substitute } \ell\text{)} \tag{17}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \left(wx^{(i)} + b\right)\right)^2 \text{(substitute linear function } h\text{)} \tag{18}$$

$$\tag{19}$$

Now, using our assumption on the structure of our hypothesis (mainly, using linear hypotheses) and the loss functions, we have taken our original optimization problem from

$$h^* = \operatorname*{argmin}_{h\in\mathcal{H}} \hat{R}(h)$$

to something that can be directly calculated:

$$w^*, b^* = \operatorname*{argmin}_{w\in\mathbb{R},\, b\in\mathbb{R}} \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \left(wx^{(i)} + b\right)\right)^2$$

4.1.3 Optimization formulation (linear algebra version 1)

Parameters: $\boldsymbol{\theta} = \begin{bmatrix} b \\ w \end{bmatrix}$; Input data points: $\mathbf{x}^{(i)} = \begin{bmatrix} 1 \\ x^{(i)} \end{bmatrix}$; Output data points are still: $y^{(i)}$

Empricial risk:

$$\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \mathbf{x}^{(i)T}\boldsymbol{\theta}\right)^2 \tag{20}$$

$$\tag{21}$$

Empirical risk minimzation:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^2} \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \mathbf{x}^{(i)T}\boldsymbol{\theta}\right)^2 \tag{22}$$

$$\tag{23}$$

4.1.4 More to come in lecture!

In class we'll work on:

- Optimization formulation: linear algebra version 2 (utilizing the L2 norm)
- Methods to solve this optimization and find the optimal parameters
- Answering our questions about non-linear data and data with more than one input feature