Linear and Polynomial Regression

Lecture 3

Last Time

- ***Halfspaces*** are hypotheses defined as hyperplanes that separate two classes
- Training data that can be separated perfectly is ***linearly separable***
  - More general term for any hypothesis class with perfect fit for training data: ***realizable***
- The ***perceptron*** algorithm is a simple method for learning halfspaces
  - Finds ERM solution efficiently if training data is linearly separable
- Textbook: sections 9.0, 9.1.0, 9.1.2

This Class

- How can we build linear predictors for predicting continuous values?
- Textbook: section 9.2

Linear Regression

Regression

- ***Regression**** refers to modeling a relationship (often one-to-one, usually smooth) between a target variable and other variables

- *A term often used in subtly confusing ways. For example, we’ll talk about a regression model called “logistic regression” that predicts the probability of a discrete label, essentially a classification task. Don’t sweat it too much.

Continuous Regression

What if we want to learn a program that outputs continuous values?

Last class: $\mathcal{Y} = \{1, -1\}$

Now: $\mathcal{Y} = \mathbb{R}$

What applications can you think of?

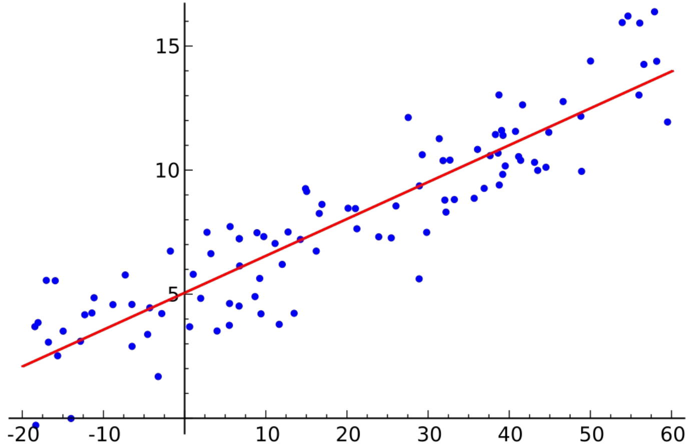
Linear Regression

Linear regression is a common hypothesis class for the following domain and label set:

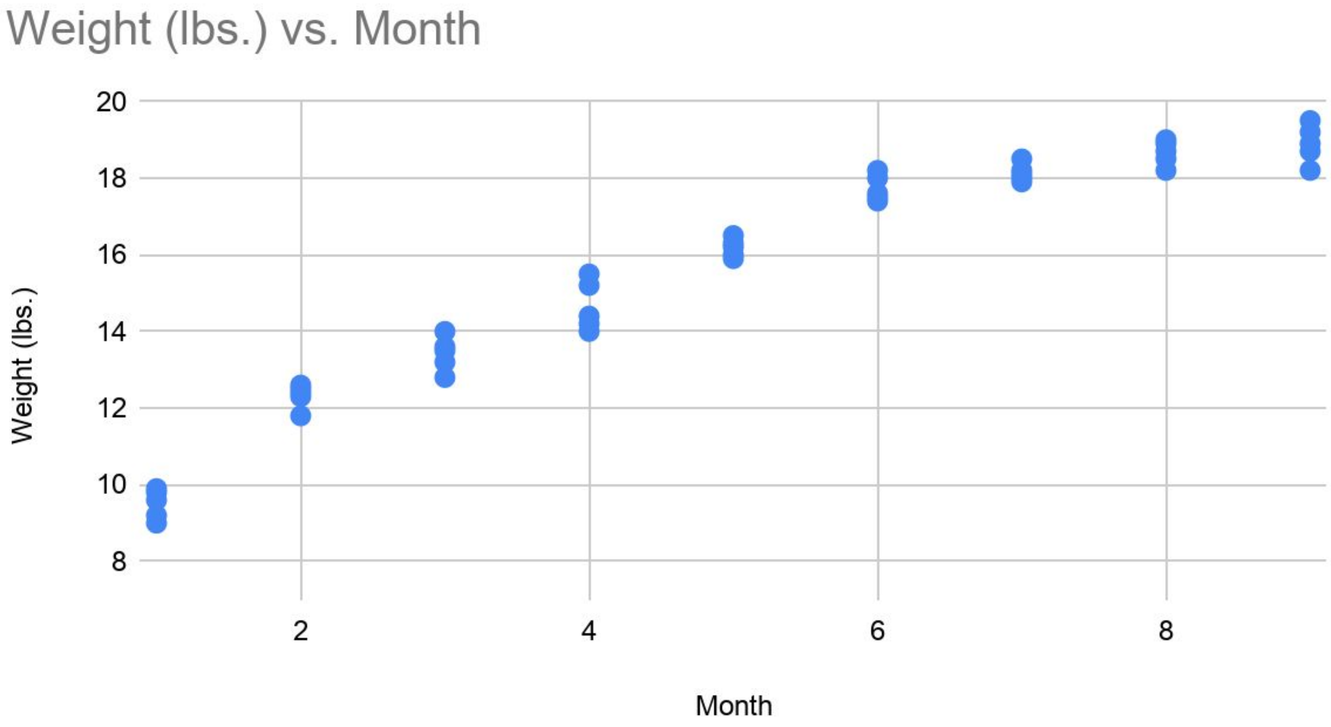
$$\mathcal{X} = \mathbb{R}^d \qquad \mathcal{Y} = \mathbb{R}$$

Again we use a linear predictor, but now its value is the output (instead of taking the sign)

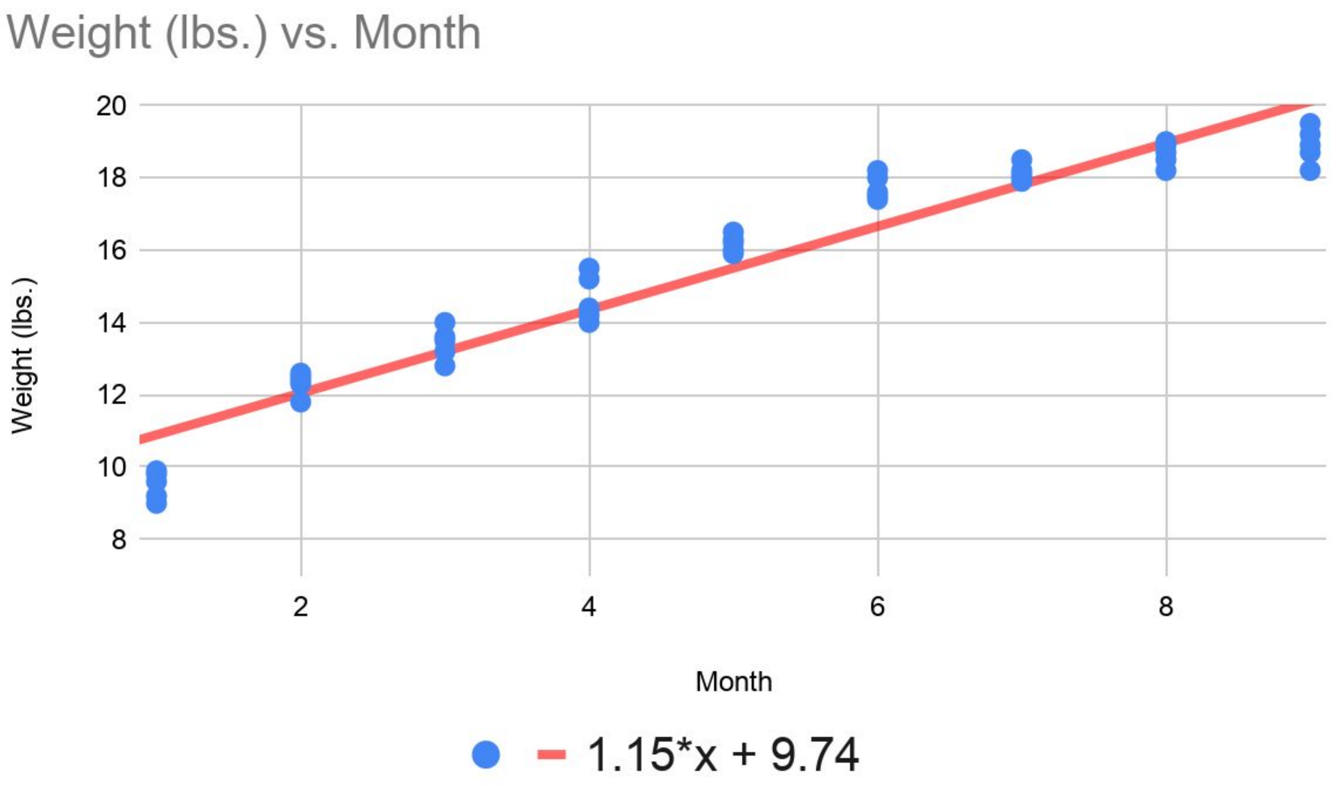
$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



Example: Baby Weight



Example: Baby Weight



ML algorithm = representation
+ loss function + optimizer

Loss for Linear Regression

- Squared loss, a.k.a. mean squared error (MSE):

$$L_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m}\left(h_{\mathbf{w}}(\mathbf{x}_i) - y_i\right)^2$$

- Another option is absolute value loss:

$$L_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m}\left|h_{\mathbf{w}}(\mathbf{x}_i) - y_i\right|$$

Question

We'll Have to Wait for the Question

ML algorithm = representation
+ loss function + optimizer

Optimizer for Linear Regression

- Least Squares

$$\operatorname*{arg\,min}_{\mathbf{w}} L_S(h_{\mathbf{w}}) = \operatorname*{arg\,min}_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2$$

- How do we solve this optimization problem?

Least Squares

- Want to solve

$$\arg\min_{\mathbf{w}} L_S(h_{\mathbf{w}}) = \arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2$$

- We start by finding the gradient of the objective with respect to $\mathbf{w}$ and setting it equal to the zero vector

$$\frac{2}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x_i}\rangle - y_i)\mathbf{x}_i = \mathbf{0}$$

- Calculus tells us that this is a necessary condition for any optimizer $\mathbf{w}_S$

Rewriting the Least Squares Condition

A useful way to rewrite $\frac{2}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x_i}\rangle - y_i)\mathbf{x}_i = \mathbf{0}$ in equivalent forms:

$A\mathbf{w} = \mathbf{b}$ where $A = \left(\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top\right)$ and $\mathbf{b} = \sum_{i=1}^{m} y_i \mathbf{x}_i$

or

$$A = \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix}^T \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} \vdots & & \vdots \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

If A is invertible...

- Easy! One unique ERM: $\mathbf{w}_S = A^{-1}\mathbf{b}$

- A is invertible if and only if $\mathbf{x}_1, \ldots, \mathbf{x}_m$ span $\mathbb{R}^d$.

If A is not invertible...

It’s symmetric, so we can do an eigenvalue decomposition: $A = VDV^\top$

Now define D^+ where D^+_{ij} is D^{-1}_{ij} if D_{ij} is nonzero and 0 otherwise

Next define $A^+ = VD^+V^\top$ (Moore-Penrose Inverse)

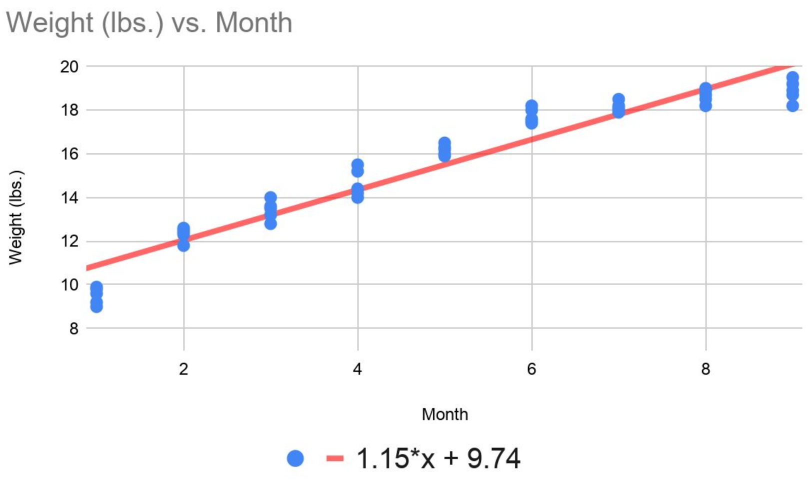
Then

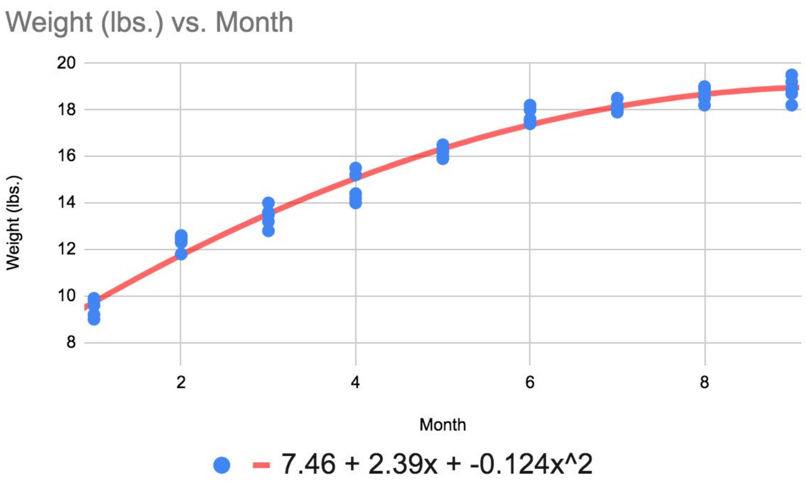
$$\mathbf{w}_S = A^+\mathbf{b}$$

Polynomial Regression

Polynomial Regression

- By expanding the representation, and therefore the hypothesis class, can also fit a more general polynomial function of the original features





Polynomial Regression

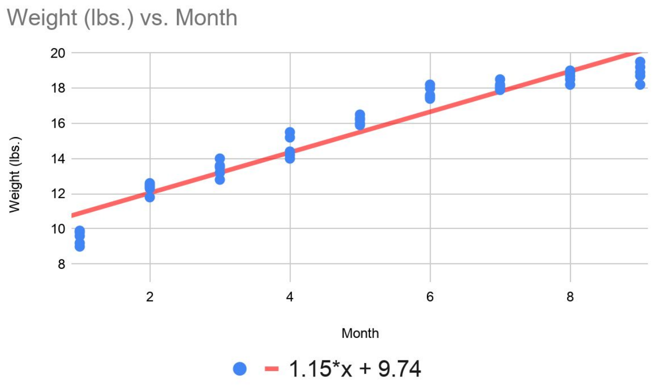
- Just like we made xor linearly separable, we can reduce empirical risk by transforming $\mathbf{x}' = \psi(\mathbf{x})$ and running the same algorithm

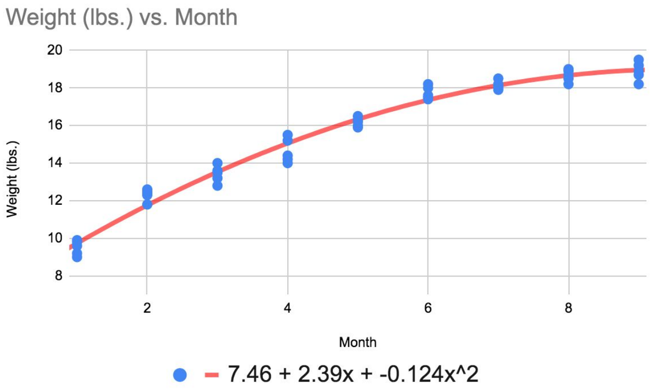
- Let's choose $\psi(\mathbf{x})$ to be the original features raised to increasing powers, i.e., if $\mathcal{X} = \mathbb{R}$ then $\psi(x) = (x, x^2, x^3)$

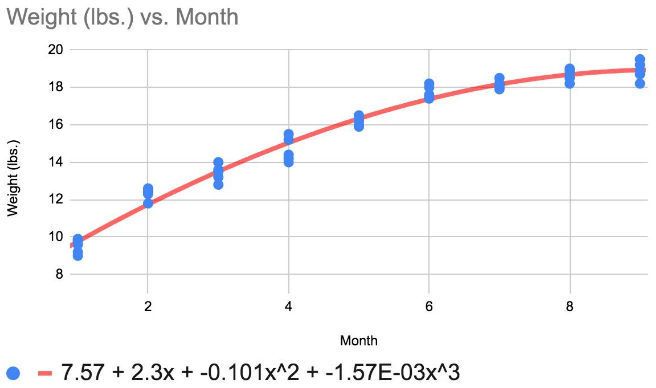
- Then running linear regression on $x' = \psi(x)$ is the same as learning the weights of a polynomial of this form:

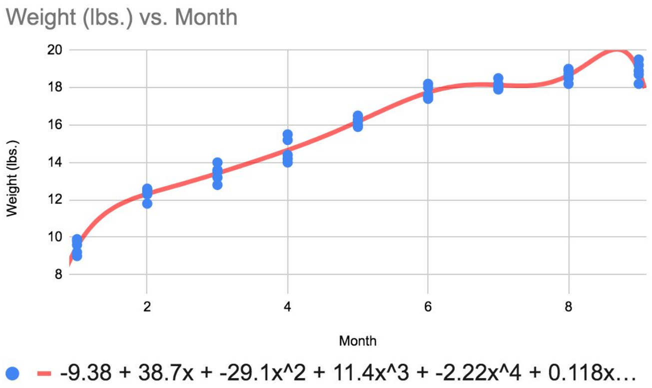
$$h(x) = w_1 + w_2 x + w_3 x^2 + w_4 x^3 + \cdots + w_{p+1} x^p$$

Polynomial Regression Demo









Question

We’ll Have to Wait for the Question

PSA from your Friendly Neighborhood Statisticians

In machine learning, modeling assumptions are rarely causal

Use extreme caution when interpreting a hypothesis or predicting counterfactuals!

The Most Important Things

- ***Linear regression*** is a hypothesis class for predicting a continuous value from a linear combination of the attributes

- ***Polynomial regression*** generalizes linear regression. Same loss and optimizer, but we expand representation by taking polynomials of chosen degree of each attribute (still linear in weights so same algorithm!)

- Textbook: section 9.2

Next Class

- How can we build linear predictors
    - for predicting probabilities of discrete classes?
    - for predicting more than two classes?
- Textbook: Chapters 9.3 12.1.1, 14.0, 14.1.0