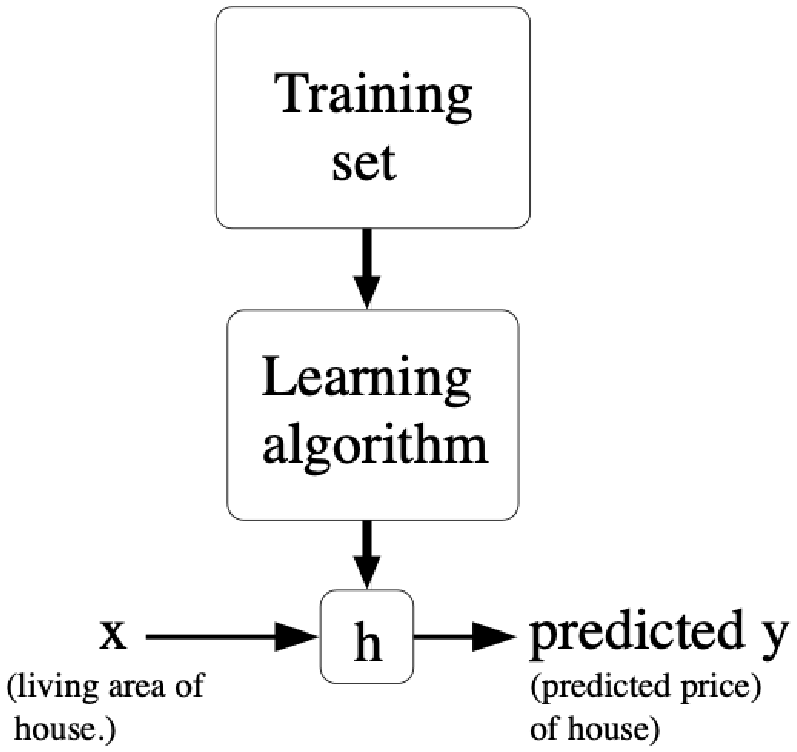
📍

Linear Regression Cheat Sheet

Simplest Machine Learning Algorithm

Prediction is the key task in machine learning. From input-output training data, we learn a function to predict new outputs. One of the simplest approaches is linear regression, which assumes a continuous linear relationship between inputs and output.

Recall	Notes
Conditions	Supervised learning: mapping from input-output training data (oppose to input-only training) $$Y \leftarrow X$$ Linear relationship and output is continuous.
Two Phases	Machine Learning has 2 phases: Learning and Predictions. Learning: • The **learning algorithm** maps the input-output training dataset, X, Y to a hypothesis function, $h(x)$ $$h\left(x\right) \leftarrow \Bigg(\quad \Bigg) \left(X, Y\right)$$ Prediction: • The **hypothesis function**, $h(x)$, predicts output. $$\hat{y} = h\left(x\right)$$



Variables / Notations

x = Inputs / Features , y = Outputs / Target, $\left(x^{(i)}, y^{(i)}\right)$ = Input-Output pair notation where i = index, m = training samples,$h_\theta\left(x\right) = h\left(x\right)$ = Hypothesis function (two notations), n = number of features.

Hypothesis Function Structure

Hypothesis function, $h\left(x\right)$

$$h\left(x\right) = \theta^T x$$

Parameters of learning algorithm, θ

$$\theta = \begin{pmatrix} \theta_0 \\ . \\ . \\ . \\ \theta_n \end{pmatrix}$$

Inputs/features, x, where we have a dummy input, x_0, always defined as 1:

$$x = \begin{pmatrix} x_0 \\ . \\ . \\ . \\ x_n \end{pmatrix}$$

$$x_0 = 1$$

- How to choose initial θ values

Different techniques, but will start with all 0s then try to update it in a way that minimizes the cost function because we want $h\left(X\right)$ to be close to Y for our training examples.

$$\theta_0 = \begin{pmatrix} 0 \\ . \\ . \\ . \\ 0 \end{pmatrix}$$

Cost Function, $J\left(\theta\right)$

Cost function, $J\left(\theta\right)$, represents the error between predicted outputs and actual outputs.

$$J\left(\theta\right) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Gradient Descent

Goal: Find a local minima of Cost Function, $J\left(\theta\right)$

1. Start at any value (Example, $\theta_0 = \vec{0}$)
    a. Initial value on the surface of the cost equation doesn't matter much for linear regression.

2. Update parameter to go down fast
    a. Want to keep improving parameters of the learning algorithm to go downhill on the cost function as fast as possible.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial x} \left(J \left(\theta \right) \right)$$

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta \left(x^{(i)} \right) \right) x_j^{(i)}$$

3. Repeat until you get to a local optimum.

Tips

Too large alpha values can overshoot, which yields to running past minimum. Too small values yields too a lot of iterations required. Practically, try a few values of alpha on an exponential scale and pick the value that drives down the learning rate the fastest. If the cost function is increasing instead of decreasing that is a very strong sign that the learning rate is too large.

$$\alpha := .01, .02, .04, .08$$

Batch Vs Stochastic

Batch gradient descent scans through the entire training set before taking a single step, where a stochastic gradient descent takes a step with each example. Stochastic is often preferred over batch gradient descent when the data set is large.

Special Properties

No local optima when gradient descent on linear regression

Normal equation for linear regression allows you to solve for optimal parameter values in one step oppose to using an iterative algorithm like gradient descent.

Normal Equation

$$\nabla_\theta J\left(\theta\right) = X^T X\theta - X^\theta \hat{y}$$

$$X^T X\theta = X^\theta \vec{y}$$

$$\theta = \left(X^T X\right)^{-1} X^\theta \vec{y}$$

Note: If X is non-invertible, that usually means you have redundant features. Features are linearly dependent. Use pseudo-inverse you get the right answer. If you can try to figure out what feature is repeated, leading to this problem

Locally Weighted Linear Regression

Non-parametric learning algorithm, which means the amount of stuff grows with data size.

Cost function, $J\left(\theta\right)$, represents the error between predicted outputs and actual outputs.

$$J\left(\theta\right) = \frac{1}{2}\sum_{i=1}^{m} w^{(i)}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$w^{(i)} = e^{\frac{-\left(x^{(i)}-x\right)^2}{2\tau^2}}$$

If $\left|x^{(i)} - x\right|$ is small, $w^{(i)} \approx 1$.If $\left|x^{(i)} - x\right|$ is large, $w^{(i)} \approx 0$.

Tau is a parameter that represent bandwidth.

Summary

Machine learning has 2 phases: Learning and Prediction. Learning Algorithm creates a hypothesis function. The hypothesis function predicts output. The cost function represents the error between predicted and actual outputs. During training, the learning algorithm tunes the parameters (of learning) to minimize error via gradient descent. Gradient descent is a popular algorithm with 3 steps:

start at any value, move in the direction that goes down the surface quickest, repeat the second step until at at local extrema.   Stochastic gradient descent updates the parameter of learning for every input-output pair, while batch gradient descent only updates the parameters of learning after scanning through the entire training set. Locally weighted regressions is a trick to fit very non-linear data.

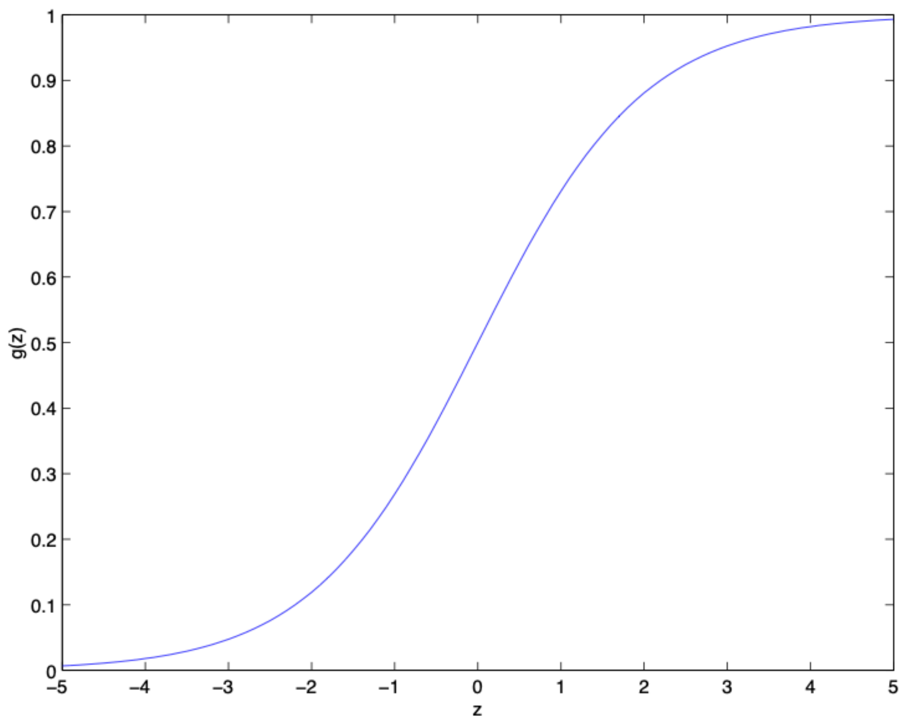
📌 **Preview:** Linear Regression and Logistic Regression are both part of a broader class of algorithms called Generalized Linear Algorithms. The next section introduces logistic regression and there's a lot of similarities, which will be omitted.

Logistic Regression

Logistic Regression is by far the most commonly used classification algorithm.

Recall	Notes
Hypothesis Function Structure	The logistic regression hypothesis function, $h\,(x)$, is a function of a sigmoid function, $g(z)$.

Logistic Vs Linear



$$g\left(z\right)=\frac{1}{1+e^{-z}}$$

$$h_{\theta}\left(x\right)=g\left(\theta^{T}x\right)=\frac{1}{1+e^{-\theta^{T}x}}$$

Logistic Regression Properties:

1. Output is between 0 and 1.

$$h_{\theta}\left(x\right)\epsilon\left[0,1\right]$$

2. Probability of output given input, parametrized by θ.

$$p\left(y\mid x;\theta\right)=h_{\theta}\left(x\right)^{y}\left(1-h_{\theta}\left(x\right)\right)^{1-y}$$

a. if $y=1$ then $p\left(y=1\mid x;\theta\right)=h_{\theta}\left(x\right)$

b. if $y=0$ then $p\left(y=0\mid x;\theta\right)=1-h_{\theta}\left(x\right)$

3. Stochastic gradient ascent of log likelihood

a. Surface level same as gradient descent of cost function of linear regression with the hypothesis function being different.

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta \left(x^{(i)} \right) \right) x_j^{(i)}$$

Summary

In logistic regression, we're doing maximize likelihood estimation, which is finding the parameters of learning algorithm that maximizes the log likelihood. Essentially choosing θ that maximizes $l(\theta)$. It turns out that the gradient ascent of log likelihood is the surface level the same equation as the gradient descent of the cost function in linear regression with the hypothesis function being different.

📌 **Preview:** Linear Regression and Logistic Regression are both part of a broader class of algorithms called Generalized Linear Algorithms. The next section introduces logistic regression and there's a lot of similarities, which will be omitted.