CSE 519: Data Science
Steven Skiena
Stony Brook University

Lecture 17: Gradient Descent Search and Regularization

Issues with Closed Form Solution

This closed form for linear regression is concise and elegant, but issues include:

- Inversion slow for large systems
- Formulation is brittle: the linear algebra magic is hard to extend to other formulations

This motivates the gradient descent approach to solving regression.

Regression as Parameter Fitting

We seek coefficients that minimize the sum of squared error of the points over all possible coefficients.

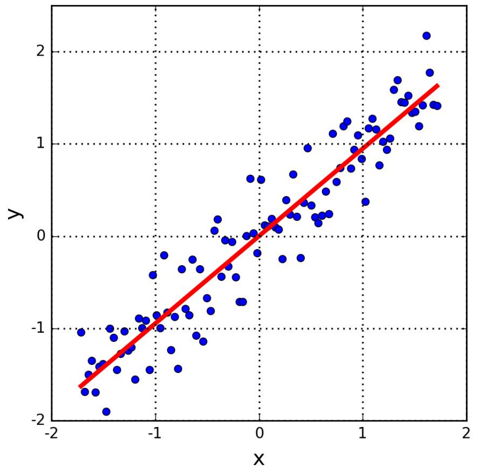
$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

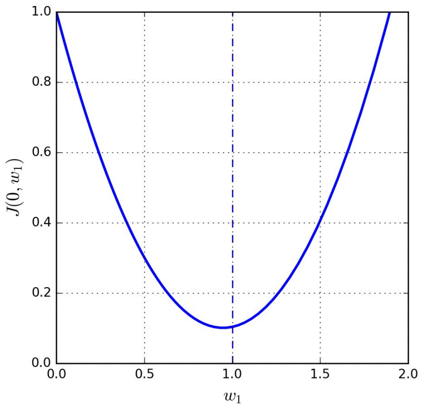
Here the regression line is:

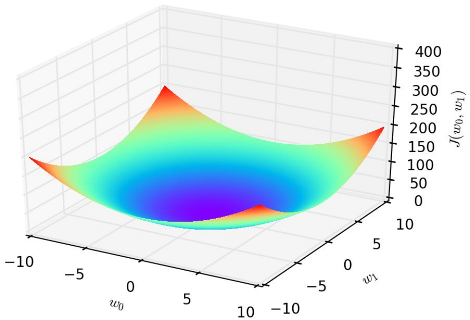
$$f(x) = w_0 + w_1 x$$

Lines in Parameter Space

The error function J(w0,w1) is convex, making it easy to find the single local/global minima.







Gradient Descent Search

A space with only one local/global minima is called convex.

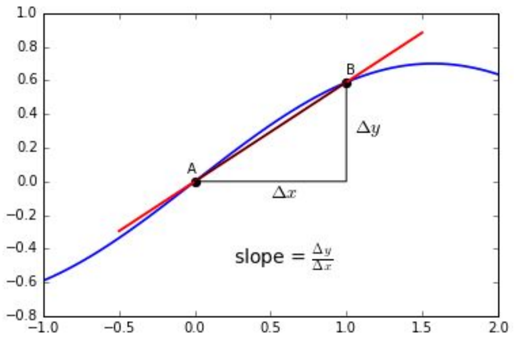
When a search space is convex, it is easy to find the minima: just keep walking down.

The fastest direction down is defined by the slope or tangent at the current point.

The Fastest Way Down

The direction down at a point is given by its derivative, which specified by its tangent line:

This *could* be approximately computed by finding the point (x+dx,y(x+dx)) and fitting the line with (x,y(x))



Partial Derivatives

The symbolic way of computing the gradient requires computing the partial derivative of the objective function:

$$
\begin{aligned}
\frac{\partial}{\partial w_j} &= \frac{2}{\partial w_j} \frac{1}{2n} \sum_{i=1}^{n} (f(x_i) - b_i)^2 \\
&= \frac{2}{\partial w_j} \frac{1}{2n} \sum_{i=1}^{n} (w_0 + (w_1 x_i) - b_i)^2
\end{aligned}
$$

Gradient Descent for Regression

Gradient descent algorithm

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\qquad (\text{for } j = 1 \text{ and } j = 0) \\
&\}
\end{aligned}
$$

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Which Functions are Convex?

Remember your calculus!

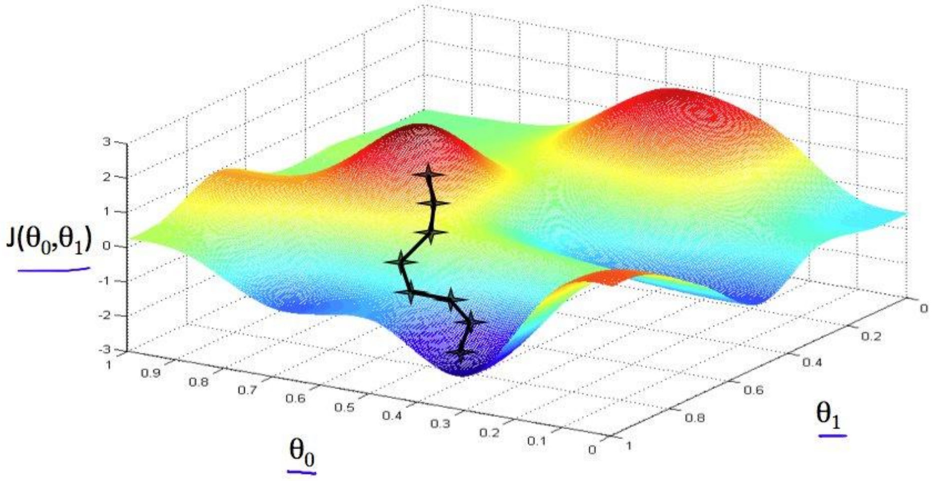
Whenever the second derivative is zero, you get a maxima or minima.

Thus analysis of such derivatives can tell which functions are and are not convex.

Gradient descent search can get trapped in local minima only for non-convex functions.

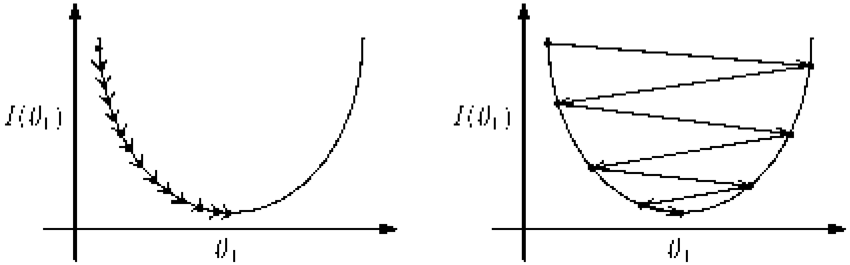
Getting Trapped in Local Optima

Always going upward does not reach the ski slope from a two story cabin in the valley.



Effect of Learning Rate / Step Size

- Taking too small steps results in slow convergence to the optima.
- But too large a step overshoots the goal.



What is the Right Learning Rate?

Monitor the value of the loss function J() over the course of optimization.

If progress is too slow, increase by a multiplicative factor (say 3) or accept.

If J gets larger, the step size is too large, decrease by a multiplicative factor (say ⅓).

Library functions should use algorithms for this.

Stochastic Gradient Descent

Evaluating the partial derivative takes time linear in the number of examples for each step!

A good heuristic is to use only a few examples to estimate the derivative, and hope it is down.

Optimizing the learning rate and the batch size for gradient descent leads to very fast optimization for convex functions.

Too Many Features?

Providing a rich set of features to regression is good, but remember Occam’s Razor:

> “The simplest explanation is best.”

Ideally our regression would select the most important variables and fit them, but our objective function only tries to minimize sum of squares error.

Regularization

The trick is to add terms to the objective function seeking to keep coefficients small:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

We pay a penalty proportional to the sum of squares of the coefficients, thus ignoring sign.

This rewards us for setting coefficients to zero.

Interpreting/Penalizing Coefficients

When variables have mean zero, its coefficient magnitude is a measure of value to the objective function.

Penalizing the sum of squared coefficients is *ridge regression* or *Tikhonov regularization*.

Penalizing the absolute value of the coefficients (L_1 metric vs. L_2) is *LASSO regularization*.

What is the right Lambda?

How do we set the constant lambda:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Big-enough lambda emphasizes small parameters, i.e. set to all zeros.

Small-enough lambda freely uses all parameters to minimize training error.

We seek balance between over/under fitting.

Tradeoffs Between Fit / Complexity

A good fit to the training data with few parameters is more robust than a slightly better fit with many parameters.

Metrics to help with model selection include:

- Akaike Information Criteria: $AIC = 2k - 2\ln(L)$
- Baysian Info Critera: $-2 \cdot \ln p(x|M) \approx \mathrm{BIC} = -2 \cdot \ln \hat{L} + k \cdot (\ln(n) - \ln(2\pi)).$

k is a parameter count and L an error metric.

Normal Form with Regularization

The normal form equation can be generalized to deal with regularization…

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

Or we can just use gradient descent with the proper loss function and derivatives.