LINEAR REGRESSION (PART II)

CS 334: Machine Learning

Slides adapted from Joyce Ho, Lee Cooper, Joydeep Ghosh, Carlos Carvalho, and Ryan Tibshirani

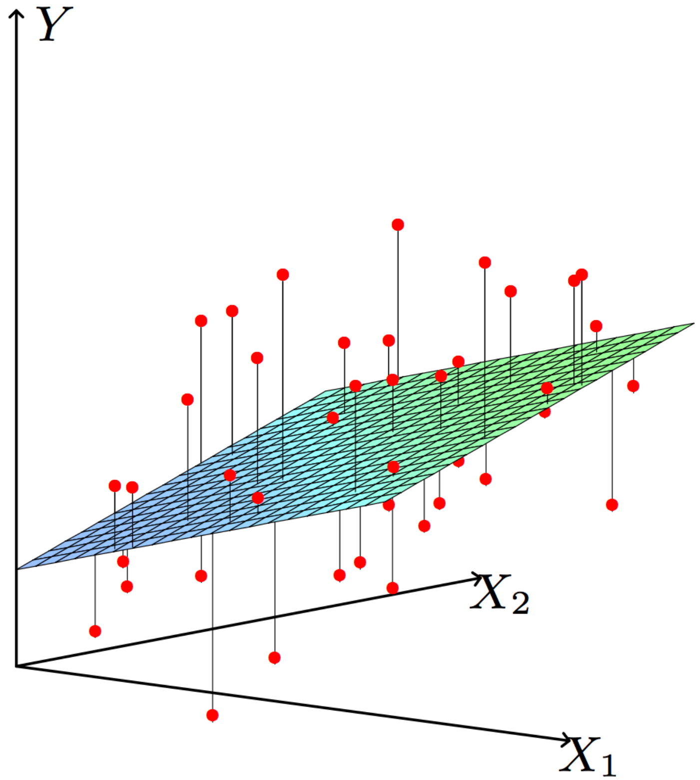
REVIEW: REGRESSION: LEAST SQUARES

- Find parameters that minimizes some cost function
- Residual: difference between actual Y and predicted Y
- Least squares: minimize residual sum of squares (RSS or SSR)

$$RSS(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i)^2)$$

$$= \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

(matrix representation later)



$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{p} x_i\beta_i$$

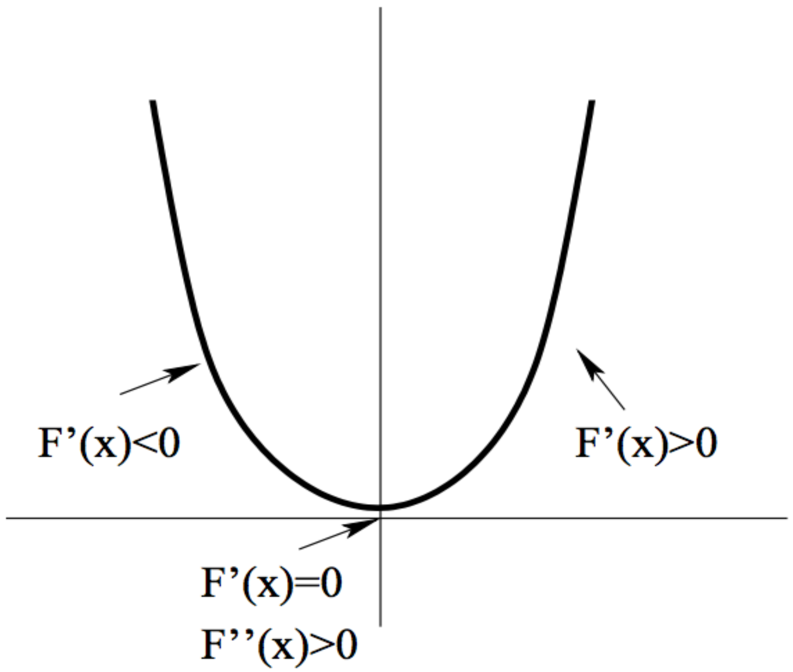
How to find the solution?

Figure 3.1 (Hastie et al.)

LEARNING THE PARAMETERS

- Closed form (direct solution): set partial derivatives to zero and solve parameters, check that Hessian is greater than 0

- Iterative algorithms: Gradient descent (GD) and Stochastic gradient descent (SGD)



Partial derivatives: https://www.khanacademy.org/math/multivariable-calculus

REVIEW: DERIVATIVE RULES

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	$2x$
Square Root	$\sqrt{x}$	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	$\ln(x)$	$1/x$
	$\log_a(x)$	$1 / (x \ln(a))$
Trigonometry (x is in radians)	$\sin(x)$	$\cos(x)$
	$\cos(x)$	$-\sin(x)$
	$\tan(x)$	$\sec^2(x)$
Inverse Trigonometry	$\sin^{-1}(x)$	$1/\sqrt{(1-x^2)}$
	$\cos^{-1}(x)$	$-1/\sqrt{(1-x^2)}$
	$\tan^{-1}(x)$	$1/(1+x^2)$

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}
Sum Rule	$f + g$	$f' + g'$
Difference Rule	$f - g$	$f' - g'$
Product Rule	fg	$f g' + f' g$
Quotient Rule	f/g	$(f' g - g' f)/g^2$
Reciprocal Rule	$1/f$	$-f'/f^2$
Chain Rule (as "Composition of Functions")	$f \circ g$	$(f' \circ g) \times g'$
Chain Rule (using ′)	$f(g(x))$	$f'(g(x))g'(x)$
Chain Rule (using $\frac{d}{dx}$)	$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

DIRECTION SOLUTION: SIMPLE LINEAR REGRESSION

- Find β_0 and β_1 that minimizes squared residual sum of residuals (SSR)

$$
\begin{aligned}
SSR &= \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= \sum_{i=1}^{n}\left(y_i^2 - 2y_i(\beta_0 + \beta_1 x_i) + \beta_0^2 + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2\right)
\end{aligned}
$$

- Solve β_0 by setting partial derivative with respect to β_0 to 0

$$
\begin{aligned}
\frac{\partial SSR}{\partial \beta_0} &= \sum_{i=1}^{n}(-2y_i + 2\beta_0 + 2\beta_1 x_i) \\
0 &= \sum_{i=1}^{n}\left(-y_i + \hat{\beta}_0 + \hat{\beta}_1 x_i\right) \\
0 &= -n\bar{y} + n\hat{\beta}_0 + \hat{\beta}_1 n\bar{x} \\
\hat{\beta}_0 &= \bar{y} - \hat{\beta}_1\bar{x}
\end{aligned}
$$

DIRECT SOLUTION: SIMPLE LINEAR REGRESSION

- Solve β_1 by setting partial derivative with respect to β_1 to 0

$$
\begin{aligned}
SSR &= \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= \sum_{i=1}^{n}\left(y_i^2 - 2y_i(\beta_0 + \beta_1 x_i) + \beta_0^2 + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial SSR}{\partial \beta_1} &= \sum_{i=1}^{n}\left(-2x_i y_i + 2\beta_0 x_i + 2\beta_1 x_i^2\right) \\
0 &= -\sum_{i=1}^{n} x_i y_i + \hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 \\
0 &= -\sum_{i=1}^{n} x_i y_i + (\bar{y} - \hat{\beta}_1 \bar{x}) \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 \\
\hat{\beta}_1 &= \frac{\sum_{i=1}^{n} x_i (y_i - \bar{y})}{\sum_{i=1}^{n} x_i (x_i - \bar{x})}
\end{aligned}
$$

EXAMPLE

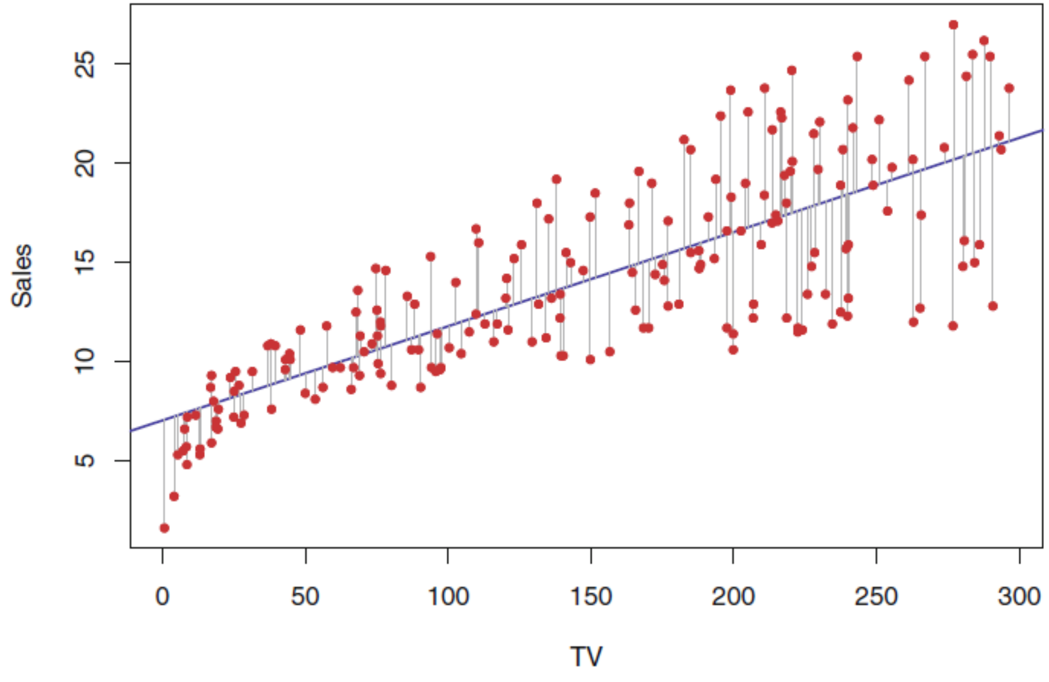


FIGURE 3.1. *For the* Advertising *data, the least squares fit for the regression*

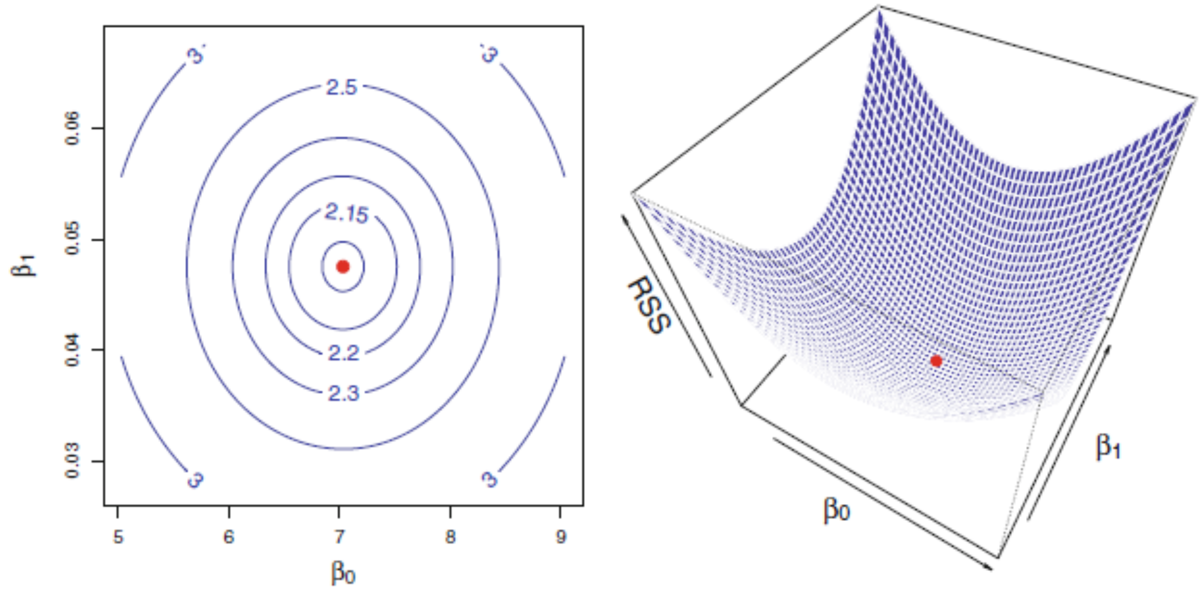


FIGURE 3.2. *Contour and three-dimensional plots of the RSS on the* Advertising *data, using* sales *as the response and* TV *as the predictor. The red dots correspond to the least squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, given by (3.4).*

DIRECT SOLUTION: SIMPLE LINEAR REGRESSION

- Elementwise representation can be cumbersome
- Many features/coefficients in practice

$$
\begin{aligned}
SSR &= \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= \sum_{i=1}^{n}\left(y_i^2 - 2y_i(\beta_0 + \beta_1 x_i) + \beta_0^2 + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial SSR}{\partial \beta_1} &= \sum_{i=1}^{n}\left(-2x_i y_i + 2\beta_0 x_i + 2\beta_1 x_i^2\right) \\
0 &= -\sum_{i=1}^{n} x_i y_i + \hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 \\
0 &= -\sum_{i=1}^{n} x_i y_i + (\bar{y} - \hat{\beta}_1 \bar{x}) \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 \\
\boxed{\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i(y_i - \bar{y})}{\sum_{i=1}^{n} x_i(x_i - \bar{x})}}
\end{aligned}
$$

(elementwise representation)

VECTORIZATION

- Rewrite the linear regression model and solution methods in matrices and vectors
- Simpler and more compact
- Utilize linear algebra libraries for faster computations

Linear algebra: https://www.khanacademy.org/math/linear-algebra

REVIEW: NOTATION

- Vector: $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x} = X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- Matrix: $\mathbf{A} \in \mathbb{R}^{m \times n}$

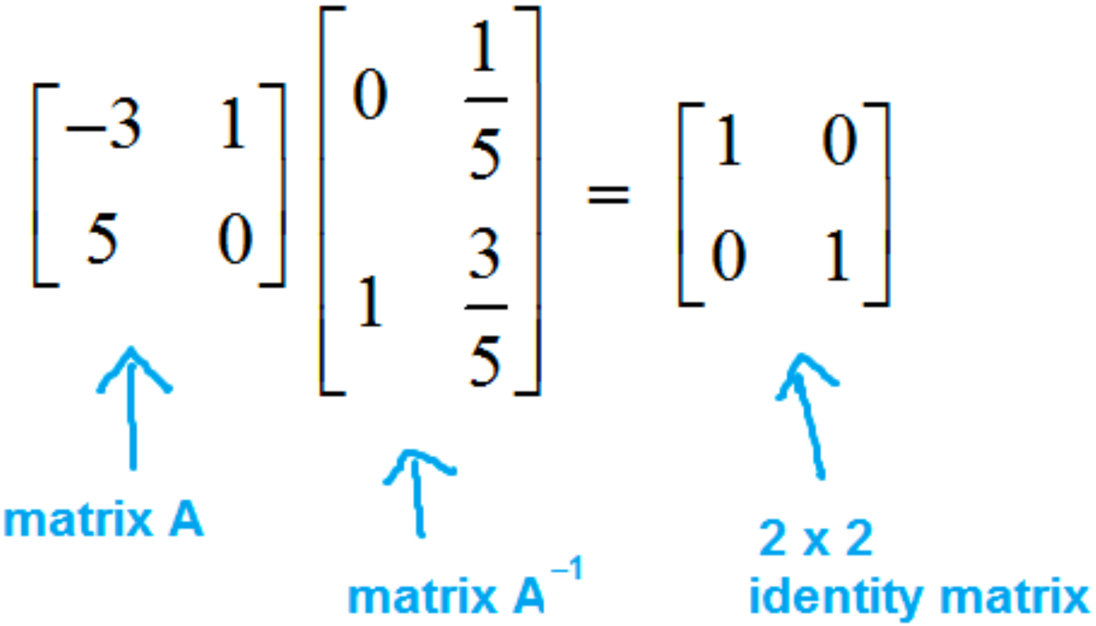
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

REVIEW: MATRIX INVERSE

- Unique matrix such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}$$

- A is invertible and non-singular if inverse exists
- A is singular if not invertible
- A must be full rank to have an inverse



https://deepai.org/machine-learning-glossary-and-terms/invertible-matrix

REVIEW: MATRIX/VECTOR MANIPULATION

Rule	Comments
$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$	order is reversed, everything is transposed
$(\mathbf{a}^T\mathbf{B}\mathbf{c})^T = \mathbf{c}^T\mathbf{B}^T\mathbf{a}$	as above
$\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$	(the result is a scalar, and the transpose of a scalar is itself)
$(\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC}$	multiplication is distributive
$(\mathbf{a}+\mathbf{b})^T\mathbf{C} = \mathbf{a}^T\mathbf{C}+\mathbf{b}^T\mathbf{C}$	as above, with vectors
$\mathbf{AB} \neq \mathbf{BA}$	multiplication is **not** commutative

http://www.gatsby.ucl.ac.uk/teaching/courses/sntn/sntn-2017/resources/Matrix_derivatives_cribsheet.pdf

REVIEW: GRADIENTS

- Generalize derivatives to several variables

- Gradient of function f:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

REVIEW: VECTOR DERIVATIVES

Scalar derivative			Vector derivative		
$f(x)$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}x}$	$f(\mathbf{x})$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{B}$	$\rightarrow$	$\mathbf{B}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{b}$	$\rightarrow$	$\mathbf{b}$
x^2	$\rightarrow$	$2x$	$\mathbf{x}^T\mathbf{x}$	$\rightarrow$	$2\mathbf{x}$
bx^2	$\rightarrow$	$2bx$	$\mathbf{x}^T\mathbf{B}\mathbf{x}$	$\rightarrow$	$2\mathbf{B}\mathbf{x}$

http://www.gatsby.ucl.ac.uk/teaching/courses/sntn/sntn-2017/resources/Matrix_derivatives_cribsheet.pdf

LINEAR REGRESSION: MATRIX REPRESENTATION

- Outcome variables

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Predictor variables
  n x (p+1)

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

- Coefficients

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

LINEAR REGRESSION: MATRIX REPRESENTATION

- Outcome variables

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Predictor variables
  n x (p+1)

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

- Coefficients

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

- Prediction

$$\mathbf{x}\beta = \begin{bmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \vdots \\ \beta_0 + \beta_1 x_n \end{bmatrix}$$

- Residual

$$\mathbf{e}(\beta) = \mathbf{y} - \mathbf{x}\beta$$

- MSE

$$\begin{aligned} MSE(\beta) &= \frac{1}{n}\mathbf{e}^T\mathbf{e} \\ &= \frac{1}{n}(\mathbf{y} - \mathbf{x}\beta)^T(\mathbf{y} - \mathbf{x}\beta) \end{aligned}$$

DIRECT SOLUTION: MATRIX FORM

- Goal: find coefficient vector β that minimizes MSE

$$
\begin{aligned}
MSE(\beta) &= \frac{1}{n}\mathbf{e}^T\mathbf{e} \\
&= \frac{1}{n}(\mathbf{y}-\mathbf{x}\beta)^T(\mathbf{y}-\mathbf{x}\beta) \\
&= \frac{1}{n}(\mathbf{y}^T-\beta^T\mathbf{x}^T)(\mathbf{y}-\mathbf{x}\beta) \\
&= \frac{1}{n}\left(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{x}\beta-\beta^T\mathbf{x}^T\mathbf{y}+\beta^T\mathbf{x}^T\mathbf{x}\beta\right)
\end{aligned}
$$

- Computer the gradient of the MSE with respect to β

$$
\begin{aligned}
\nabla MSE(\beta) &= \frac{1}{n}\left(\nabla\mathbf{y}^T\mathbf{y}-2\nabla\beta^T\mathbf{x}^T\mathbf{y}+\nabla\beta^T\mathbf{x}^T\mathbf{x}\beta\right) \\
&= \frac{1}{n}\left(0-2\mathbf{x}^T\mathbf{y}+2\mathbf{x}^T\mathbf{x}\beta\right) \\
&= \frac{2}{n}\left(\mathbf{x}^T\mathbf{x}\beta-\mathbf{x}^T\mathbf{y}\right)
\end{aligned}
$$

- Set the gradient to 0, solve β

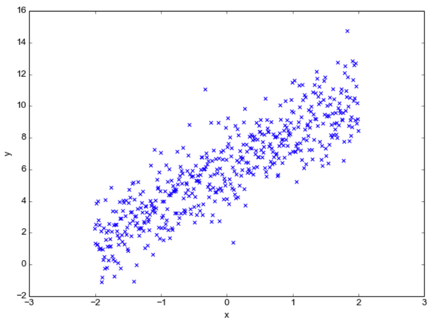
$$\mathbf{x}^T\mathbf{x}\widehat{\beta}-\mathbf{x}^T\mathbf{y}=0$$

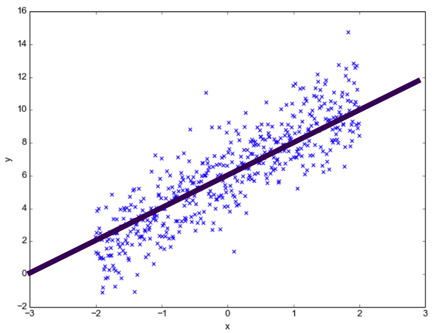
$$\boxed{\widehat{\beta}=(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}}$$

http://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/13/lecture-13.pdf

LINEAR REGRESSION

- Training:





$$\mathbf{x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$$

- Prediction: $\mathrm{X} \longrightarrow \hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$

LINEAR ALGEBRA: PYTHON (HINT FOR HW3)

- Create an array of ones: numpy.ones
- Concatenation: numpy.concatenate
- Multiplication: numpy.matmul
- Transpose numpy.transpose
- Inverse: numpy.linalg.inv

GEOMETRY OF LS SOLUTION

- Outcome vector is orthogonally projected onto hyperplane spanned by input features

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

- The "hat" matrix or projection matrix

$$\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$$

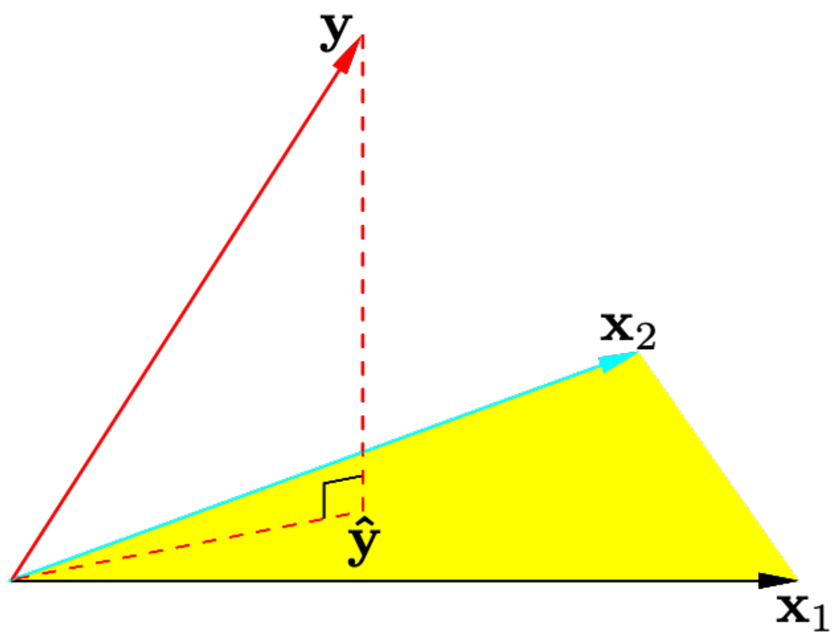


Figure 3.2 (Hastie et al.)

LEARNING THE PARAMETERS

- Closed form solution :

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- What happens to computation and space complexity for large datasets (e.g., lots of samples and lots of features)?

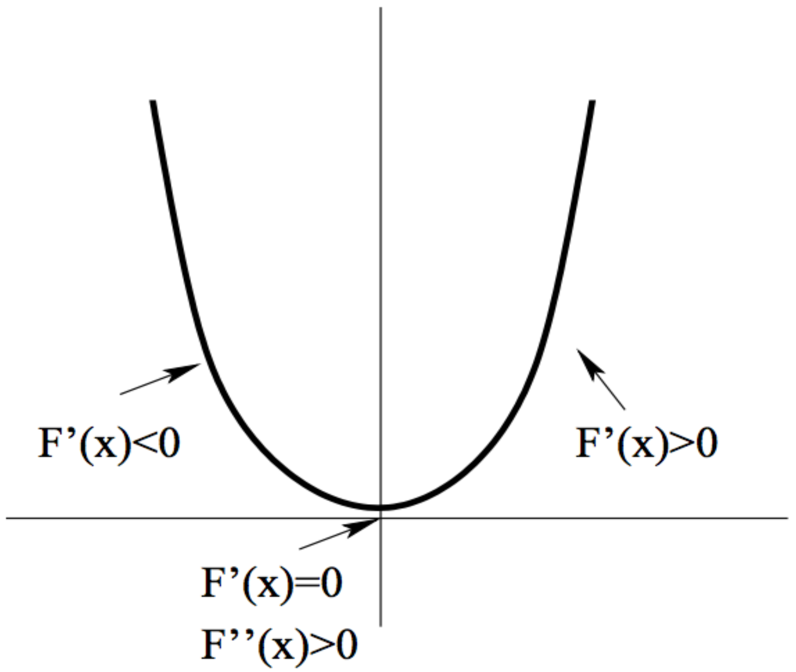
- Other optimization problems that do not have closed form solution?

What can we do differently?

LEARNING THE PARAMETERS

- Closed form (direct solution): set partial derivatives to zero and solve parameters, check that Hessian is great than 0

- Iterative algorithms: Gradient descent (GD) and Stochastic gradient descent (SGD)



Partial derivatives: https://www.khanacademy.org/math/multivariable-calculus

UNCONSTRAINED OPTIMIZATION

- Objective function: $\min f_0(\mathbf{x})$

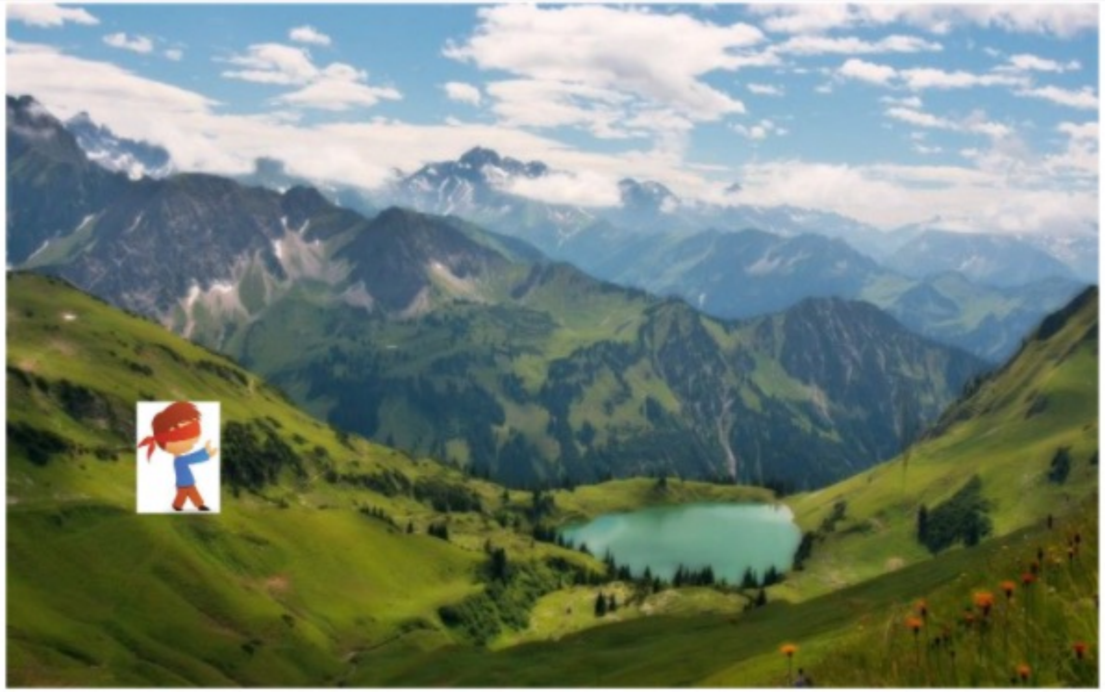
- Example: $\min \mathbf{x}^\top \mathbf{x}$

- How to find the optimal point?

GROUP ACTIVITY

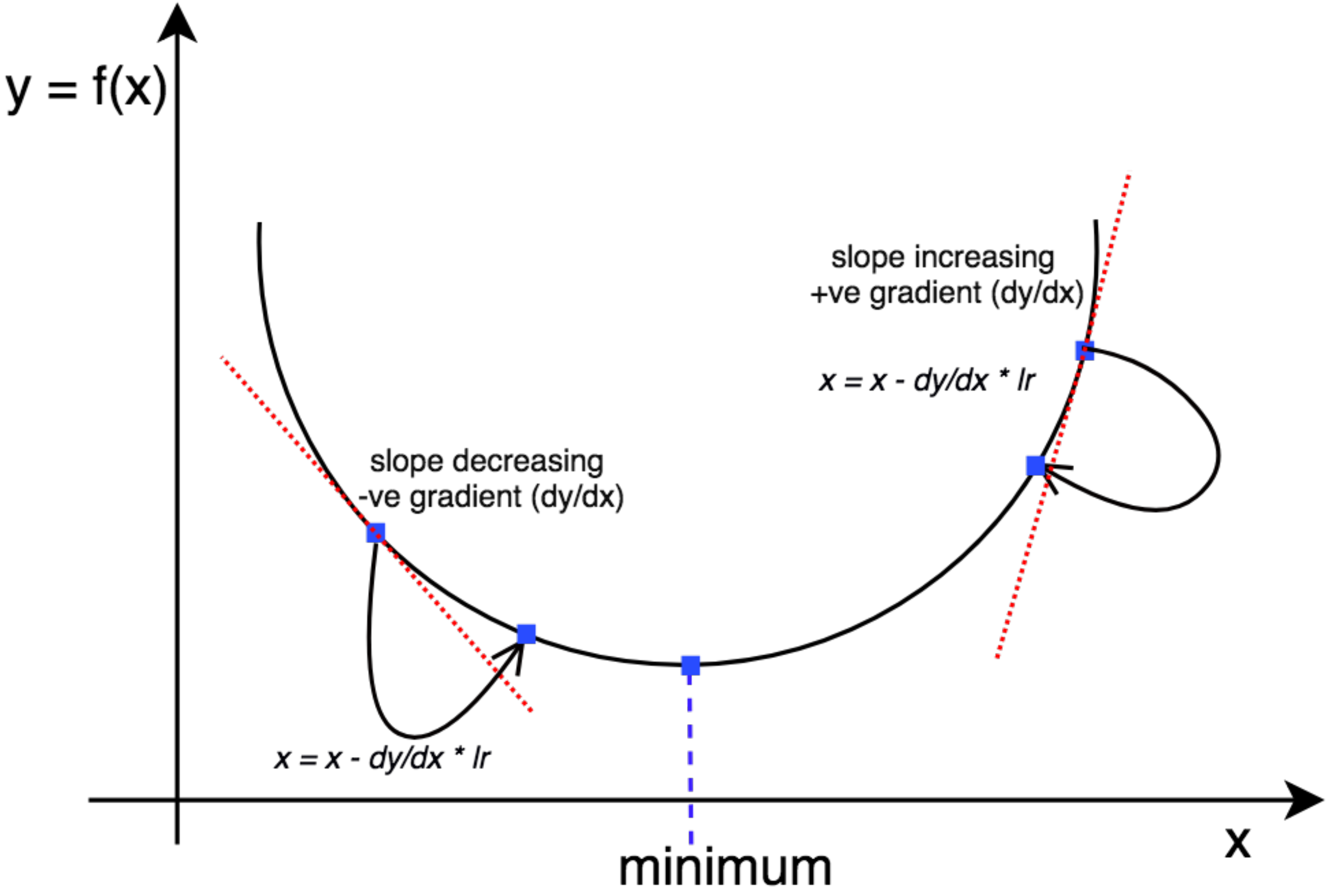
BLINDFOLDED MOUNTAIN CLIMBING

Your friend blindfolds you and drops you on the side of a mountain. Your goal is to get to the lake at the bottom of the mountain (valley) as quickly as possible. All you can do is feel the mountain around where you are standing, and take steps. How would you get there?

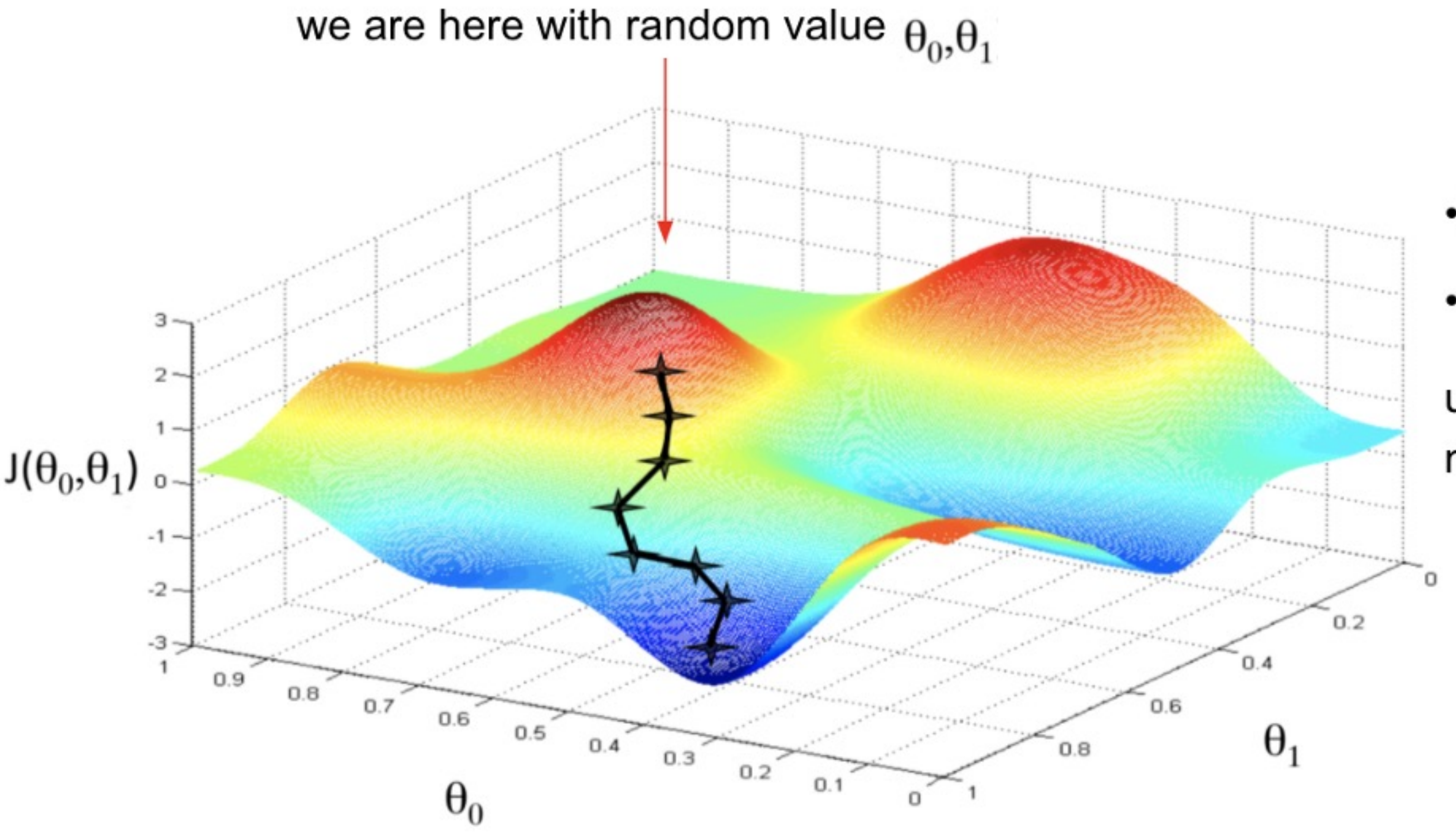


GRADIENT DESCENT (GD)

- Main Idea: Take a step proportional to the negative gradient (steepest slope) to get closer to the minimum
- Learning rate determines the proportion (step size)



GRADIENT DESCENT (GD)



- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

until we hopefully end up at a minimum

GD: ALGORITHM

Algorithm 1: Gradient Descent

while *Not Converged* **do**

$x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla f(x)$

end

return $x^{(k+1)}$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

GD: ALGORITHM

Algorithm 1: Gradient Descent

while *Not Converged* **do**

$$x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla f(x)$$

end

return $x^{(k+1)}$

Stopping criteria

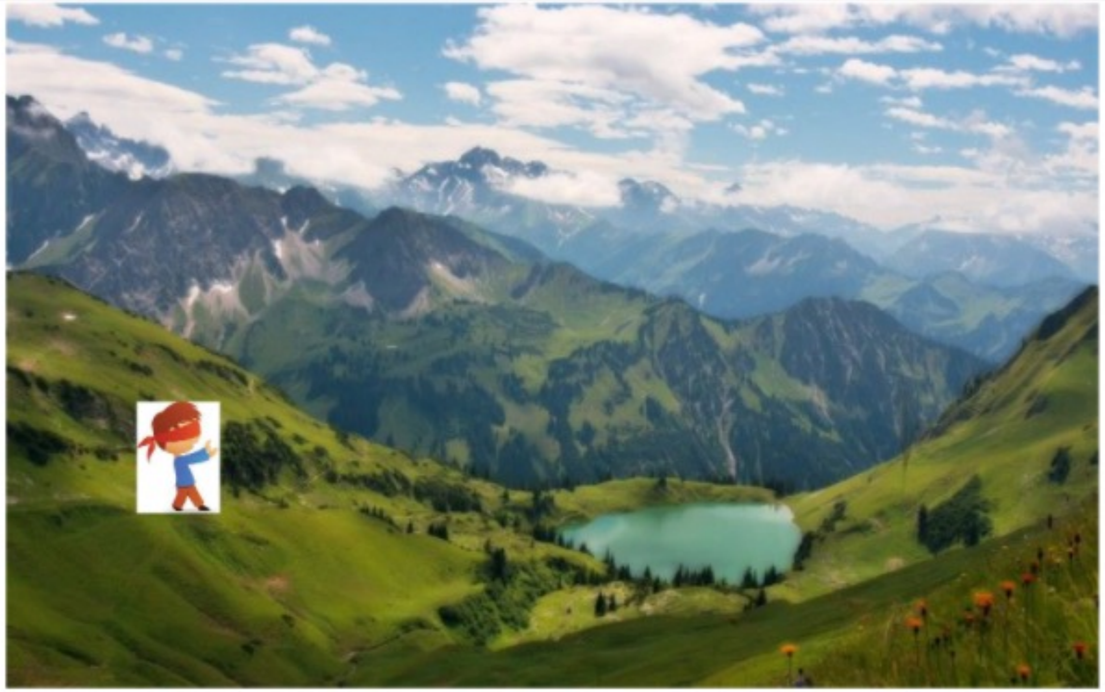
Step size
(learning rate)

STOPPING CRITERIA

- When the norm of the gradient is close to 0
- When the coefficients have stabilized
- When the performance on validation data is not getting better

STEP SIZE

- Fixed step size

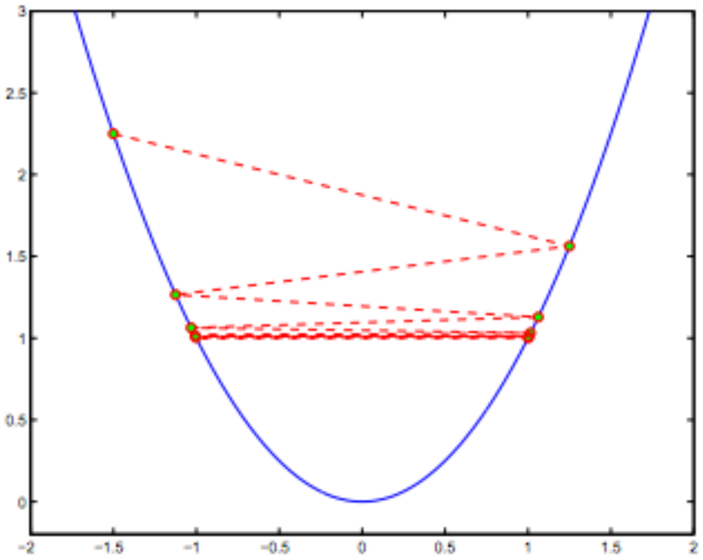


What happens if the step size is too big or too small?

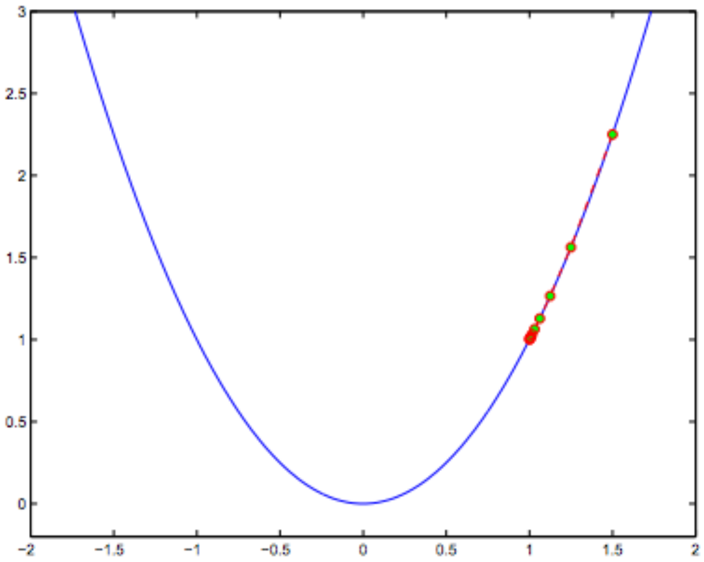
https://www.cs.cmu.edu/~ggordon/10725-F12/scribes/10725_Lecture5.pdf

GD: IMPORTANCE OF STEP SIZE

- Challenge is to find a good step size to avoid step size that is too long or too short



too long => divergence



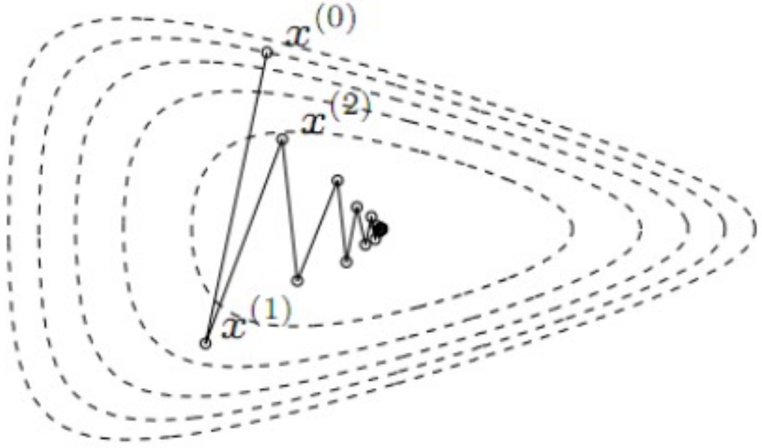
too short => slow convergence

STEP SIZE

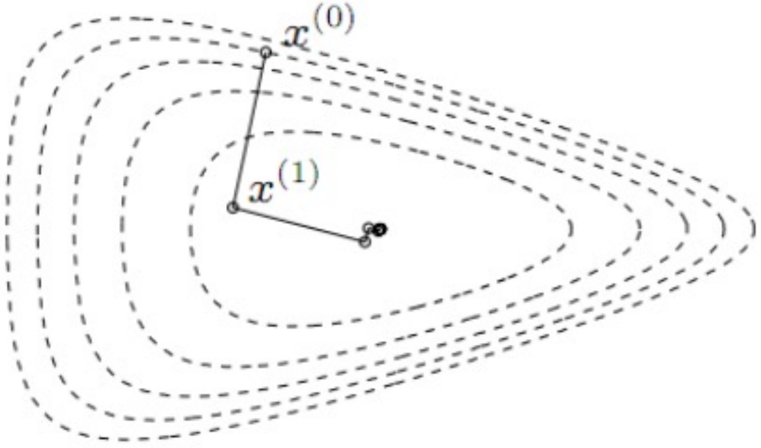
- Fixed step size
- Exact line search: find best step size for each iteration (not practical)
- Backtracking line search: start with a large step size and iteratively shrink the step size (backtracking) until expected decrease of the objective function

GD: LINE SEARCH

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



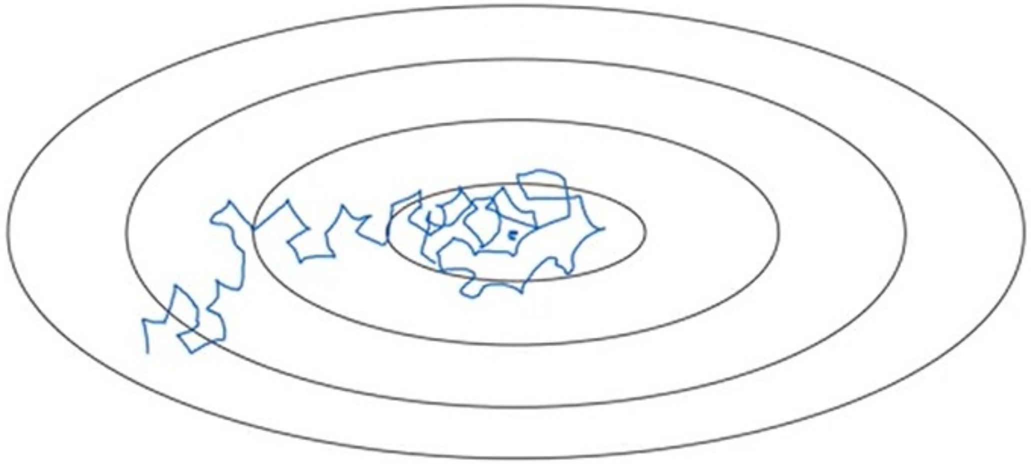
backtracking line search



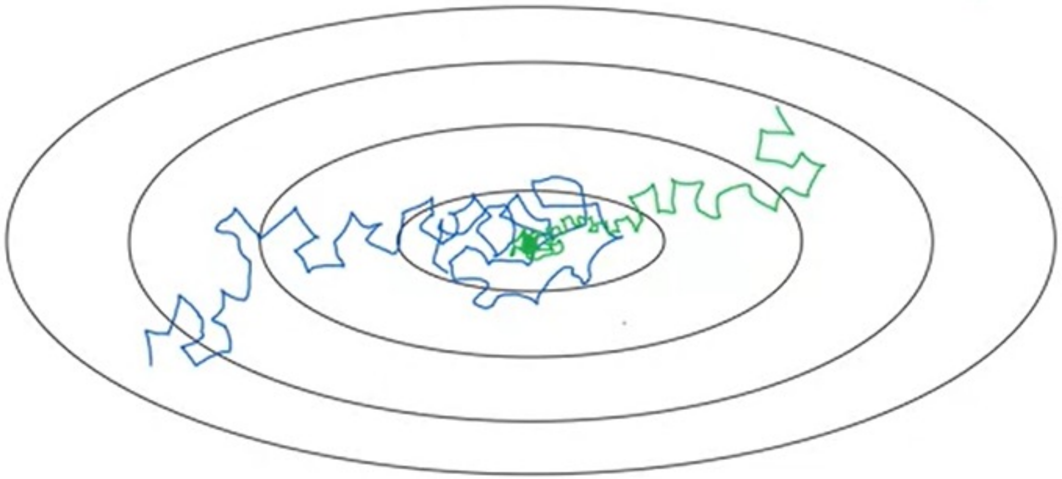
exact line search

Boyd & Landenberghe's Book on Convex Optimization

LEARNING RATE DECAY



Algorithm converging with a constant learning rate (Noisy and represented with Blue)



Algorithm converging while decaying Learning Rate over time (less noisy and represented with green)

- Start with a large learning rate, then slowly reduce/decay it until local minimum is obtained

https://medium.com/analytics-vidhya/learning-rate-decay-and-methods-in-deep-learning-2cee564f910b