COMP 551 - Applied Machine Learning
Lecture 2– Linear Regression

William L. Hamilton

(with slides and content from Joelle Pineau)

Some logistics (revisited)

- Course website: https://www.cs.mcgill.ca/~wlh/comp551/
  - Used for: Schedule, links to lectures slides, homework materials, and readings
  - TA office hours are up on the website.
- MyCourses
  - Used for: Announcements, quizzes, grading, and discussions.
  - There is now a discussion forum for finding project partners.

Quizzes

- There will be **two quizzes a week (starting this week).**
  - One available Tuesday-Wednesday.
  - The other available Thursday-Friday.
- They will be on MyCourses.
- The two weekly quizzes will be similar, but not identical.
- **Only your best quiz from each week will count towards your grade.**
- **Important: This week's quizzes are for practice and self-assessment. They will not count towards your grade.**

Python tutorial hours

- There will be Python tutorial hours in Trottier 3120:
  - 6-8pm on Friday, Sept. 13th
  - 4:30-6:30pm on Monday, Sept. 16th.
  - 3-5 pm on Tuesday, Sept. 17th.
  - 4-6 pm on Wednesday, Sept. 18th.

- Come if you have never used Python and need help!
- There is limited space and they will be first-come-first-served.
- Please attend the whole session if you go.

Supervised learning

- Given a set of training examples: $x_i = < x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,m}, y_i >$

  $x_{i,j}$ is the *j*th feature of the *i*th example

  y_i is the desired output (or target) for the *i*th example.
- We want to learn a function $f : X_1 \times X_2 \times \ldots \times X_m \rightarrow Y$, which maps the input variables to the output/target.
- Formally, f is called the hypothesis (or model).

tumor size	texture	perimeter	shade	outcome	size change
18.02	rough	117.5	0 (very light)	Y	-0.14
16.05	smooth	112.2	4 (dark)	Y	-0.10
18.9	smooth	102.3	1 (light)	N	+0.21

Prediction problems

- Classification
  - E.g., predicting whether a treatment is successful vs. unsuccessful
  - Y is a finite discrete set (e.g., successful vs. unsuccessful treatment)
- Regression
  - E.g., predicting the future size of a tumor
  - $Y = \Re$ (i.e., we are predicting a real number)

tumor size	texture	perimeter	shade	outcome	size change
18.02	rough	117.5	0 (very light)	Y	-0.14
16.05	smooth	112.2	4 (dark)	Y	-0.10
18.9	smooth	102.3	1 (light)	N	+0.21

Variable types

- Numerical, real number measurements (usually *quantitative*).
  - Assumes that similar measurements are similar in nature.
- Categorical, from a discrete set (often *qualitative*)
  - E.g. {Spam, Not-spam}
- Ordinal, from a discrete set, without metric relation, but allows ranking.
  - E.g. {first, second, third}

tumor size	texture	perimeter	shade	outcome	size change
18.02	rough	117.5	0 (very light)	Y	-0.14
16.05	smooth	112.2	4 (dark)	Y	-0.10
18.9	smooth	102.3	1 (light)	N	+0.21

The i.i.d. assumption

- In supervised learning, the examples x_i in the training set are assumed to be independently and identically distributed.

The i.i.d. assumption

- In supervised learning, the examples x_i in the training set are assumed to be independently and identically distributed.
  - Independently: Every x_i is freshly sampled according to some probability distribution D over the data domain X.
  - Identically: The distribution D is the same for all examples.

Why?

Empirical risk minimization

For a given function class F and training sample S,

- Define a notion of error (*left intentionally vague for now*):

$$L_S(f) = \#\text{ mistakes made on the sample } S$$

Empirical risk minimization

For a given function class F and training sample S,

- Define a notion of error (*left intentionally vague for now*):

  $L_S(f)$ = # mistakes made on the sample S

- Define the Empirical Risk Minimization (ERM):

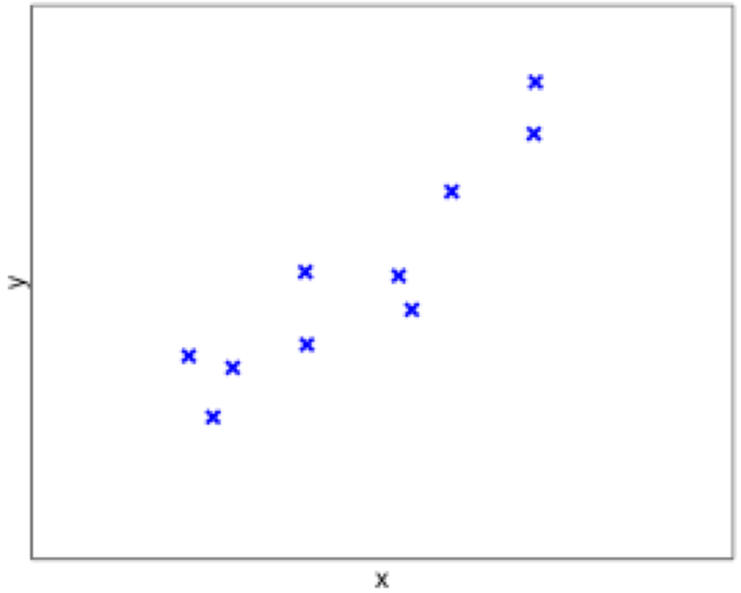
  $$\mathrm{ERM}_F(S) = \mathrm{argmin}_{f \text{ in } F}\, L_S(f)$$

  where argmin returns the function f (or set of functions) that achieves the minimum loss on the training sample.

- Easier to minimize the error with i.i.d. assumption

Empirical risk minimization

- What <u>hypothesis class</u> should we pick?



Observe x	*Predict* y
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.1	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

Linear hypothesis

- Suppose y is a linear function of $\mathbf{x}$:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_m x_m$$

$$= w_0 + \sum_{j=1:m} w_j x_j$$

- The $\mathbf{w}_j$ are called parameters or weights.
- To simplify notation, we add an attribute $x_0 = 1$ to the m other attributes (also called bias term or intercept).

Linear hypothesis

- Suppose y is a linear function of $\mathbf{x}$:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \dots + w_m x_m$$

$$= w_0 + \sum_{j=1:m} w_j x_j$$

- The $\mathbf{w}_j$ are called parameters or weights.
- To simplify notation, we add an attribute $x_0=1$ to the m other attributes (also called bias term or intercept).

How should we pick the *weights*?

Least-squares solution method

- The linear regression problem: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1:m} w_j x_j$

  where m = the dimension of observation space, i.e. number of features.
- Goal: Find the best linear model (i.e., weights) given the data.
- Many different possible evaluation criteria!

- Most common choice is to find the $\mathbf{w}$ that minimizes:

$$\mathrm{Err}(\mathbf{w}) = \sum_{i=1:n} (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

(A note on notation: Technically $\mathbf{w}$ and $\mathbf{x}$ are vectors of size $m+1$, i.e., number of features + the dummy/intercept term. However, for notational simplicity from now on we say that $\mathbf{w}$ and $\mathbf{x}$ are vectors of size m, with $m=$"number of features $+ 1$" and just treat the intercept as an extra feature)

Least-squares solution method

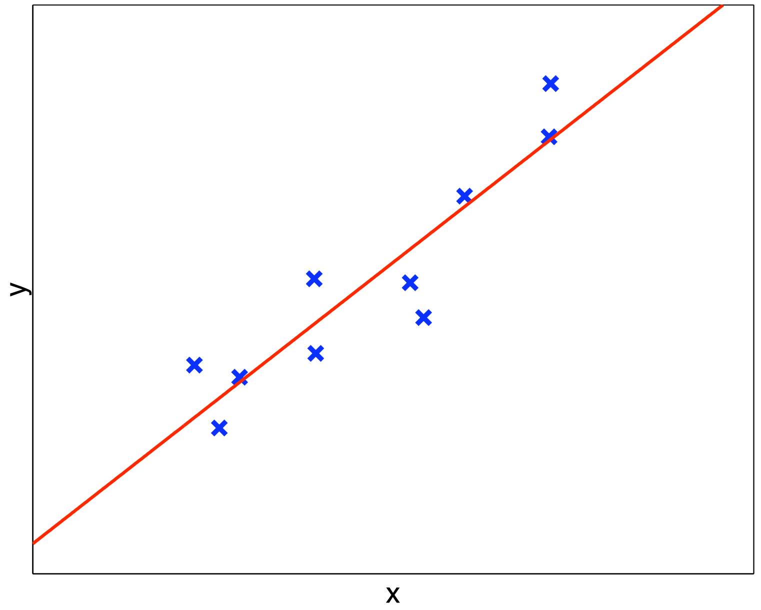
- **Goal:** Find a function of the form $f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1:m} w_j x_j$

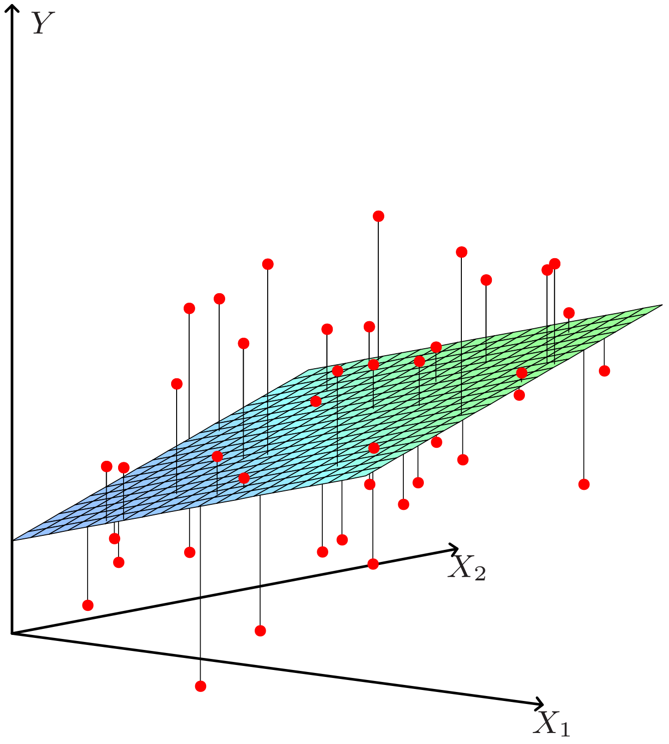
- such that

$$\boxed{f_{\mathbf{w}} = \operatorname{argmin} \sum_{i=1:n} (\underbrace{y_i}_{\text{True target value}} - \underbrace{\mathbf{w}^T\mathbf{x}_i}_{\text{Predicted value for the target}})^2}$$

This is "least-squares" solution

Least-squares solution





Least-squares solution method

- Re-write in matrix notation: $\mathbf{f_w(X)} = \mathbf{Xw}$

$$\mathrm{Err}(\mathbf{w}) = (\mathbf{y} - \mathbf{Xw})^T(\mathbf{y} - \mathbf{Xw})$$

where $\mathbf{X}$ is the $n \times m$ matrix of input data,
$\mathbf{y}$ is the $n \times 1$ vector of output data,
$\mathbf{w}$ is the $m \times 1$ vector of weights.

Least-squares solution method

- Re-write in matrix notation: $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{Xw}$

  $$\mathrm{Err}(\mathbf{w}) = (\mathbf{y} - \mathbf{Xw})^T(\mathbf{y} - \mathbf{Xw})$$

  where $\mathbf{X}$ is the $n \times m$ matrix of input data,
  $\mathbf{y}$ is the $n \times 1$ vector of output data,
  $\mathbf{w}$ is the $m \times 1$ vector of weights.

- To minimize, take the derivative w.r.t. *w:*

  $$\partial\mathrm{Err}(\mathbf{w})/\partial\mathbf{w} = -2\,\mathbf{X}^T(\mathbf{y}-\mathbf{Xw})$$

  - You get a system of m equations with m unknowns.

Least-squares solution method

- Re-write in matrix notation: $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$

  $\text{Err}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$

  where $\mathbf{X}$ is the $n \times m$ matrix of input data,
  $\mathbf{y}$ is the $n \times 1$ vector of output data,
  $\mathbf{w}$ is the $m \times 1$ vector of weights.

- To minimize, take the derivative w.r.t. *w:*

  $\partial\text{Err}(\mathbf{w})/\partial\mathbf{w} = -2\,\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w})$

  - You get a system of m equations with m unknowns.

- Set these equations to 0: $\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$

Least-squares solution method

- We want to solve for $\mathbf{w}$: $X^T (Y - X\mathbf{w}) = 0$

- Try a little algebra: $X^T Y = X^T X \mathbf{w}$

  $\hat{\mathbf{w}} = (X^TX)^{-1} X^T Y$

  ($\hat{\mathbf{w}}$ denotes the estimated weights)

Least-squares solution method

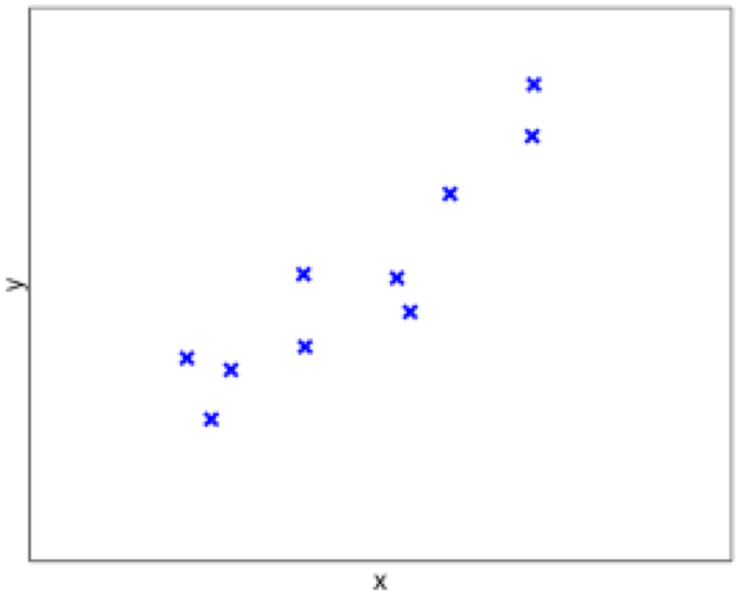
- We want to solve for $\mathbf{w}$: $X^T (Y - X\mathbf{w}) = 0$
- Try a little algebra: $X^T Y = X^T X \mathbf{w}$

  $\hat{\mathbf{w}} = (X^TX)^{-1} X^T Y$

  ($\hat{\mathbf{w}}$ denotes the estimated weights)
- The fitted data: $\hat{Y} = X\hat{\mathbf{w}} = X (X^TX)^{-1} X^T Y$
- To predict new data $X' \rightarrow Y'$: $Y' = X'\hat{\mathbf{w}} = X' (X^TX)^{-1} X^T Y$

Example of linear regression

- What is a plausible estimate of $\mathbf{w}$?



Observe x	*Predict* y
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.1	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

Example of linear regression

- What is a plausible estimate of $\mathbf{w}$?
- Recall that our least-squares estimate is
- $$\hat{\mathbf{w}} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{Y}$$

Observe	*Predict*
x	*y*
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.1	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

Data matrices

$$X^T X =$$

$$\left[\begin{array}{cccccccccc} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \times \left[\begin{array}{cc} 0.86 & 1 \\ 0.09 & 1 \\ -0.85 & 1 \\ 0.87 & 1 \\ -0.44 & 1 \\ -0.43 & 1 \\ -1.10 & 1 \\ 0.40 & 1 \\ -0.96 & 1 \\ 0.17 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc} 4.95 & -1.39 \\ -1.39 & 10 \end{array}\right]$$

Data matrices

$$X^T Y =$$

$$\begin{bmatrix} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$= \begin{bmatrix} 6.49 \\ 8.34 \end{bmatrix}$$

Solving the problem

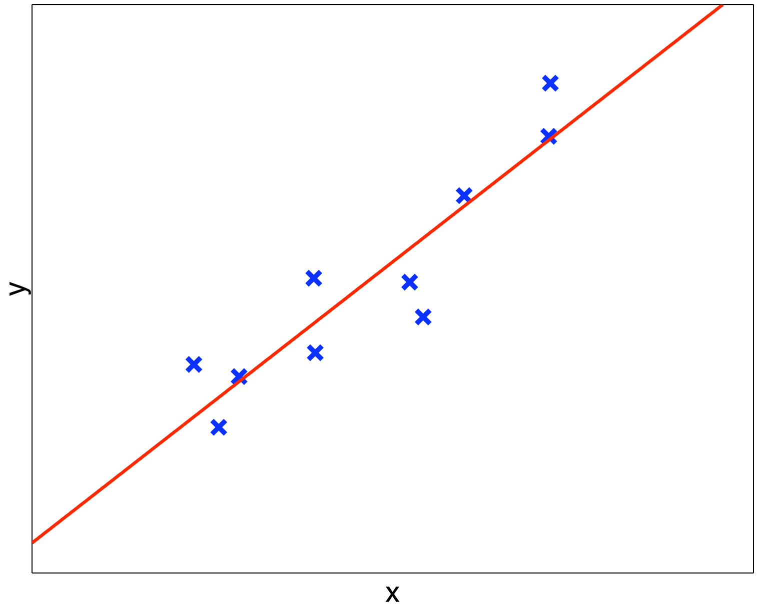
$$\mathbf{w} = (X^TX)^{-1}X^TY = \begin{bmatrix} 4.95 & -1.39 \\ -1.39 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 1.60 \\ 1.05 \end{bmatrix}$$

So the best fit line is $y = 1.60x + 1.05$.

Solving the problem

$$\mathbf{w} = (X^T X)^{-1} X^T Y = \begin{bmatrix} 4.95 & -1.39 \\ -1.39 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 1.60 \\ 1.05 \end{bmatrix}$$

So the best fit line is $y = 1.60x + 1.05$.



Least-squares solution method

- Linear fit for a prostate cancer dataset
  - Features X = {lcavol , lweight, age, lbph, svi, lcp, gleason, pgg45}
  - Output y = level of PSA (an enzyme which is elevated with cancer)..

Coefficient (i.e., learned weight, w_i)

- How does increasing x_i change the output y_i?

Standard error

- How confident/precise is the estimate of w_i?
- How "predictive" of y_i is x_i?

Term	Coefficient	Std. Error
Intercept	$w_0 =$ 2.46	0.09
lcavol	0.68	0.13
lweight	0.26	0.10
age	−0.14	0.10
lbph	0.21	0.10
svi	0.31	0.12
lcp	−0.29	0.15
gleason	−0.02	0.15
pgg45	0.27	0.15

Least-squares solution method

- Linear fit for a prostate cancer dataset
  - Features X = {lcavol , lweight, age, lbph, svi, lcp, gleason, pgg45}
  - Output y = level of PSA (an enzyme which is elevated with cancer)..

Large coefficient and small standard error

$\downarrow$

Expect a small change in x_j to have a large effect on y

Term	Coefficient	Std. Error
Intercept	$w_0 =$ 2.46	0.09
lcavol	0.68	0.13
lweight	0.26	0.10
age	−0.14	0.10
lbph	0.21	0.10
svi	0.31	0.12
lcp	−0.29	0.15
gleason	−0.02	0.15
pgg45	0.27	0.15

Computational cost of linear regression

- What operations are necessary?

Computational cost of linear regression

- What operations are necessary?
  - Overall: 1 matrix inversion + 3 matrix multiplications
  - $\mathbf{X}^T\mathbf{X}$ (other matrix multiplications require fewer operations.)
    - $\mathbf{X}^T$ is $m \times n$ and $\mathbf{X}$ is $n \times m$, so we need nm^2 operations.
  - $(X^TX)^{-1}$
    - $\mathbf{X}^T\mathbf{X}$ is $\mathbf{mxm}$, so we need m^3 operations.

Computational cost of linear regression

- What operations are necessary?
  - Overall: 1 matrix inversion + 3 matrix multiplications
  - $\mathbf{X}^T\mathbf{X}$ (other matrix multiplications require fewer operations.)
    - $\mathbf{X}^T$ is $m \times n$ and $\mathbf{X}$ is $n \times m$, so we need nm^2 operations.
  - $(X^TX)^{-1}$
    - $\mathbf{X}^T\mathbf{X}$ is $m \times m$, so we need m^3 operations.

- Overall, we have $O(m^3 + nm^2)$ (i.e, **polynomial**) computational cost, but handling really large datasets (e.g, many points or features) can still be an issue!

A more efficient alternative?

- Recall the least-squares solution: $\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$
- What if X is too big to compute this explicitly (e.g. $m \sim 10^6$)?

- Go back to the gradient step: $\mathrm{Err}(\mathbf{w}) = (Y - X\mathbf{w})^T (Y - X\mathbf{w})$

Gradient tells us how to move the parameters to maximally increase/decrease the error! $\longrightarrow$

$$\begin{aligned} \partial \mathrm{Err}(\mathbf{w})/\partial \mathbf{w} &= -2\, X^T (Y - X\mathbf{w}) \\ &= 2(X^T X\mathbf{w} - X^T Y) \end{aligned}$$

Gradient descent for linear regression

- We want to produce a sequence of weight solutions, $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2 \ldots,$ such that: $\mathrm{Err}(\mathbf{w}_0) > \mathrm{Err}(\mathbf{w}_1) > \mathrm{Err}(\mathbf{w}_2) > \ldots$

Gradient descent for linear regression

- We want to produce a sequence of weight solutions, $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2 \ldots,$ such that: $\mathrm{Err}(\mathbf{w}_0) > \mathrm{Err}(\mathbf{w}_1) > \mathrm{Err}(\mathbf{w}_2) > \ldots$

- The algorithm:

```
Given an initial weight vector $\mathbf{w}_0$,
Do for $k=1, 2, \ldots$
    $\boxed{\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \, \partial \mathrm{Err}(\mathbf{w}_k)/\partial \mathbf{w}_k}$
End when $|\mathbf{w}_{k+1}-\mathbf{w}_k| < \varepsilon$
```

Take a "step" in the (negative) direction specified by the gradient.

Gradient descent for linear regression

- We want to produce a sequence of weight solutions, $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2 \ldots$, such that: $\mathrm{Err}(\mathbf{w}_0) > \mathrm{Err}(\mathbf{w}_1) > \mathrm{Err}(\mathbf{w}_2) > \ldots$

- The algorithm:

```
Given an initial weight vector $\mathbf{w}_0$,
Do for $k=1, 2, \ldots$
    $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \, \partial \mathrm{Err}(\mathbf{w}_k) / \partial \mathbf{w}_k$
End when $|\mathbf{w}_{k+1} - \mathbf{w}_k| < \varepsilon$
```

- Parameter $\alpha_k > 0$ is the step-size (or <u>learning rate</u>) for iteration k.

Convergence

- Convergence depends in part on the α_k.

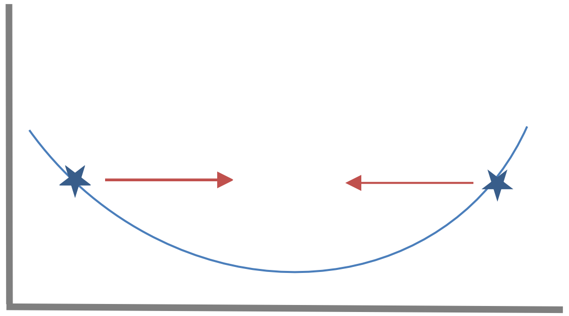
- If steps are too **large**: the $\mathbf{w}_k$ may oscillate forever.
  - This suggests that $\alpha_k \rightarrow 0$ as $k \rightarrow \infty$.

- If steps are too **small**: the $\mathbf{w}_k$ may not move far enough to reach a local minimum.

Convergence

- Convergence depends in part on the α_k.

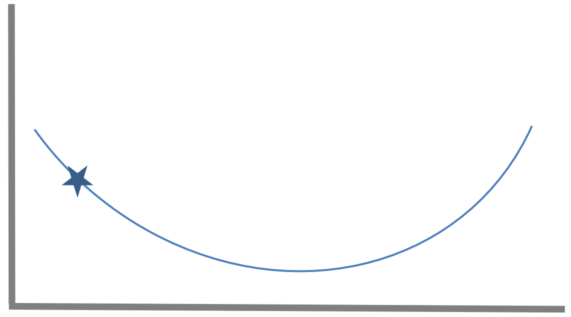
- If steps are too **large**: the $\mathbf{w}_k$ may oscillate forever.
  - This suggests that $\alpha_k \rightarrow 0$ as $k \rightarrow \infty$.



Convergence

- Convergence depends in part on the α_k.

- If steps are too **small**: the $\mathbf{w}_k$ may not move far enough to reach a local minimum.



Robbins-Monroe conditions

- The α_k are a Robbins-Monroe sequence if:

$$\sum_{k=0:\infty} \alpha_k = \infty$$
$$\sum_{k=0:\infty} \alpha_k^2 < \infty$$

- These conditions are sufficient to ensure convergence of the $\mathbf{w}_k$ to a local minimum of the error function.

Robbins-Monroe conditions

- The α_k are a Robbins-Monroe sequence if:

$$\sum_{k=0:\infty} \alpha_k = \infty$$

$$\sum_{k=0:\infty} \alpha_k^2 < \infty$$

- These conditions are sufficient to ensure convergence of the $\mathbf{w}_k$ to a <u>local minimum</u> of the error function.

E.g. $\alpha_k = 1 / (k + 1)$

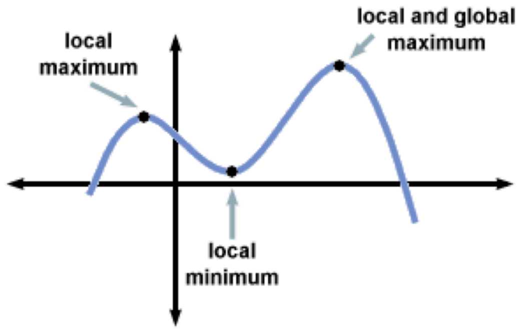
E.g. $\alpha_k = 1/2$ for $k = 1, \ldots, T$

$\alpha_k = 1/2^2$ for $k = T+1, \ldots, (T+1)+2T$

etc.

Local vs. global minima

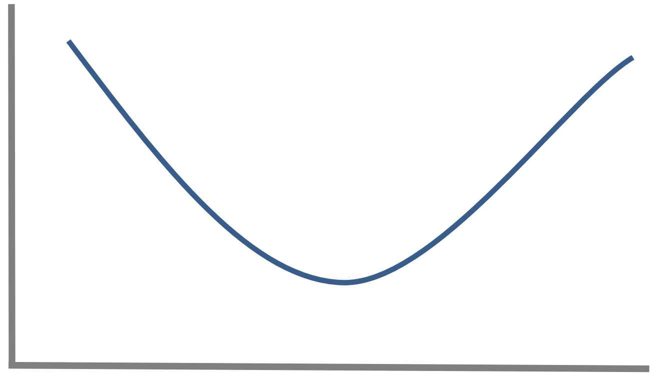
- Convergence is NOT to a global minimum, only to local minimum.



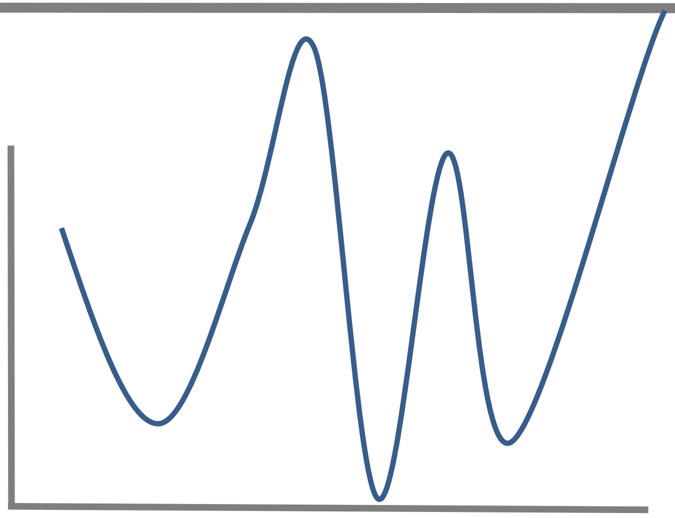
Only an issue for "non-convex" functions. For "convex" functions (e.g., simple linear regression) all local minima are also global minima.

- The blue line represents the error function. There is *no guarantee* regarding the amount of error of the weight vector found by gradient descent, compared to the globally optimal solution. (Random restarts can help.)

Convex vs. non-convex (informally)



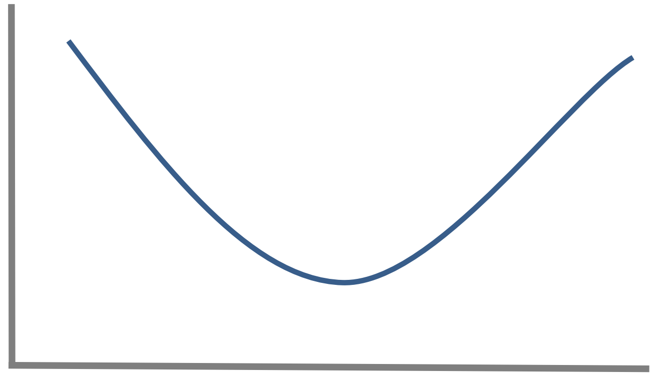
Convex



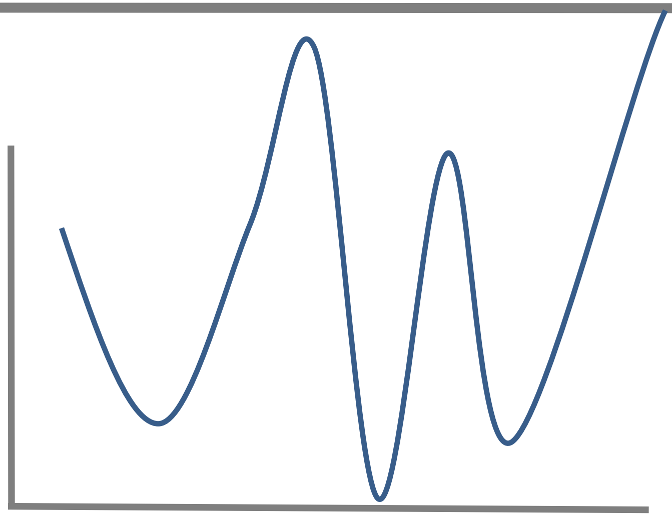
Non-convex

Most state-of-the-art approaches (e.g., deep learning) are non-convex!

Convex vs. non-convex (informally)



Convex



Non-convex

Most state-of-the-art approaches (e.g., deep learning) are non-convex!

But linear regression is convex!

What you should know

- Definition and characteristics of a supervised learning problem.
- Linear regression (hypothesis class, error function, algorithm).
- Closed-form least-squares solution method (algorithm, computational complexity, stability issues).
- Gradient descent method (algorithm, properties).

To-do list

- Reproduce the linear regression example (slides 22-27), solving it using the software of your choice.
- Practice/self-assessment quizzes this week. You should be able to get 100% on these on the second try!
- Suggested complementary readings:
  - Ch.2 (Sec. 2.1-2.4, 2.9) of Hastie et al.
  - Ch.3 of Bishop.
  - Ch.9 of Shalev-Schwartz et al.
- Write down the midterm date/time (November 18th, 6-8pm) and contact the head TA (Joey Bose, joey.bose@mail.mcgill.ca) right away if you know you need to be away at this time.