COMP 551 – Applied Machine Learning
Lecture 2: Linear regression

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's Quiz (informal)

Write down the 3 most useful insights you gathered from the article:

"*A Few Useful Things to Know About Machine Learning*".

Supervised learning

- Given a set of **training examples**: $x_i = < x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}, y_i >$

  x_{ij} is the j^{th} feature of the i^{th} example

  y_i is the desired **output** (or **target)** for the i^{th} example.

  X_j denotes the j^{th} feature.

- We want to learn a function $f : X_1 \times X_2 \times \ldots \times X_n \to Y$

  which maps the input variables onto the output domain.

tumor size	texture	perimeter	...	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27
...					

Supervised learning

- Given a dataset $X \times Y$, find a function: $f : X \rightarrow Y$ such that $f(\mathbf{x})$ is a good predictor for the value of y.
- Formally, f is called the **hypothesis**.
- Output Y can have many types:
  - If $Y = \Re$, this problem is called **regression**.
  - If Y is a finite discrete set, the problem is called **classification**.
  - If Y has 2 elements, the problem is called **binary classification**.

Prediction problems

- The problem of predicting <u>tumour recurrence</u> is called:

  <u>classification</u>

- The problem of predicting the <u>time of recurrence</u> is called:

  <u>regression</u>

- Treat them as two separate supervised learning problems.

tumor size	texture	perimeter	...	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27
...					

Variable types

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

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*):

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

- Define the Empirical Risk Minimization (ERM):

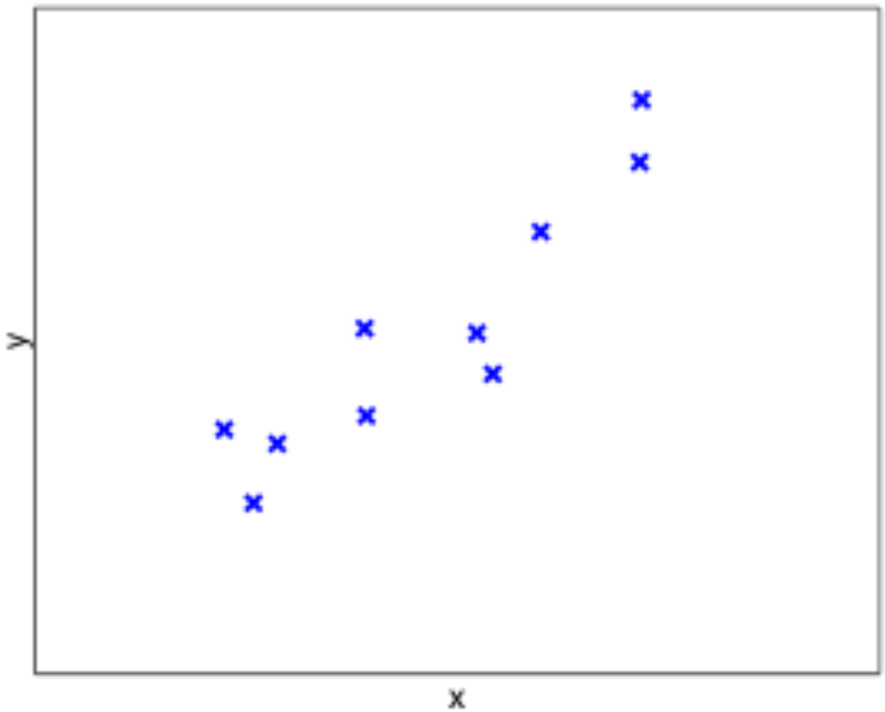
$$ERM_F(S) = argmin_{f \text{ in } F}\, L_S(f)$$

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.

A regression problem

- 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 $\mathbf{X}$:

$$f_W(\mathbf{X}) = w_0 + w_1 x_1 + \ldots + w_m x_m$$

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

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

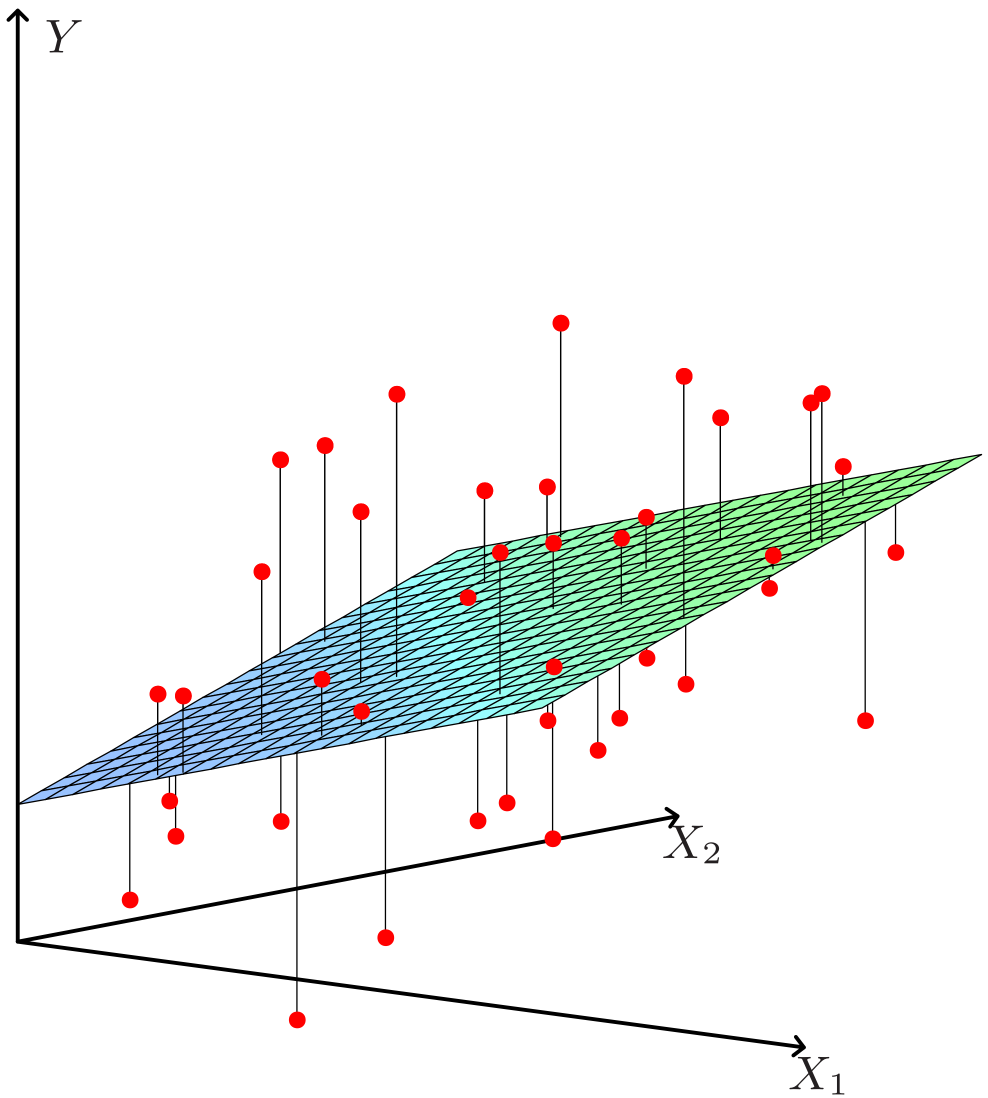
  where m = the dimension of observation space, i.e. number of features.

- **Goal**: Find the **best** linear model given the data.
- Many different possible **evaluation** criteria!
- Most common choice is to find the $\mathbf{w}$ that minimizes:

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

(A note on notation: Here $\mathbf{w}$ and $\mathbf{x}$ are column vectors of size $m+1$.)

Least-squares solution for $X \in \Re^2$



Least-squares solution method

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

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

  where X is the $n \times m$ matrix of input data,
  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. $\mathbf{w}$:

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

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

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

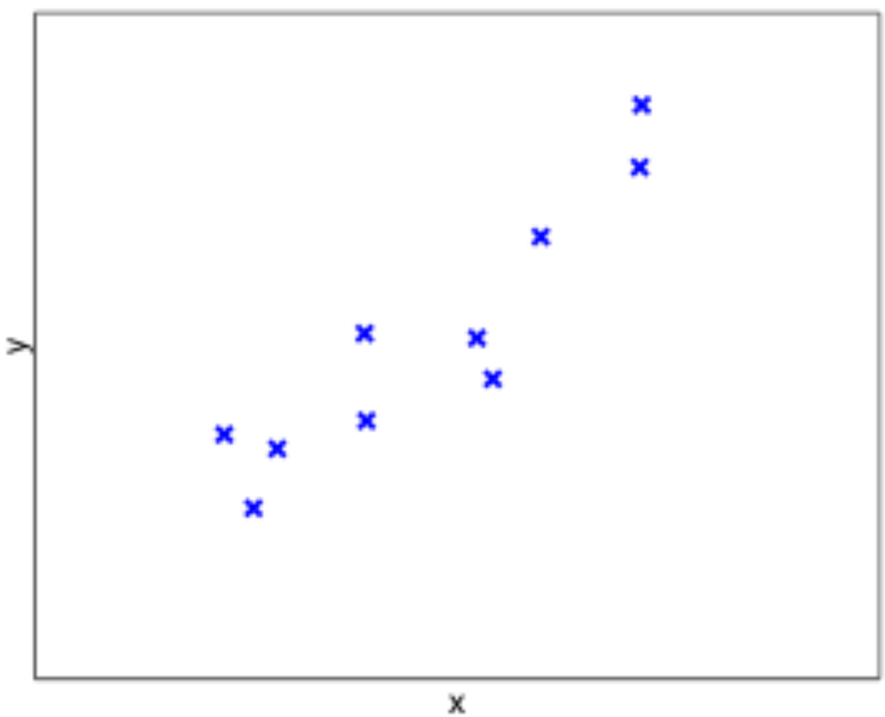
Least-squares solution method

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

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

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

Example of linear regression



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
-1.10	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

What is a plausible estimate of ***w*** ? **Try it!**

Data matrices

$$X^T X =$$

$$\begin{bmatrix} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.86 & 1 \\ 0.09 & 1 \\ -0.85 & 1 \\ 0.87 & 1 \\ -0.44 & 1 \\ -0.43 & 1 \\ -1.10 & 1 \\ 0.40 & 1 \\ -0.96 & 1 \\ 0.17 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4.95 & -1.39 \\ -1.39 & 10 \end{bmatrix}$$

Data matrices

$$X^T Y =$$

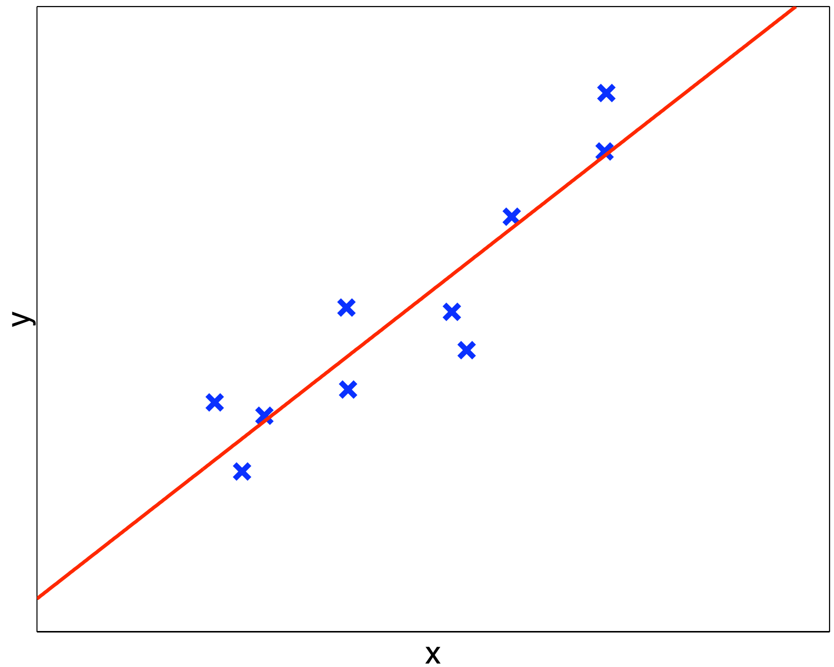
$$\left[\begin{array}{cccccccccc} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \times \left[\begin{array}{c} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{array}\right]$$

$$= \left[\begin{array}{c} 6.49 \\ 8.34 \end{array}\right]$$

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 $y = 1.60x + 1.05$.



Interpreting the solution

- 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).
  - High coefficient weight (in absolute value) = important for prediction.

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?
  - Overall: 1 matrix inversion + 3 matrix multiplications
  - X^TX (other matrix multiplications require fewer operations.)
    - X^T is mxn and X is nxm, so we need nm^2 operations.
  - $(X^TX)^{-1}$
    - X^TX is mxm, so we need m^3 operations.

- We can do linear regression in polynomial time, but handling large datasets (many examples, many features) can be problematic.

An alternative for minimizing mean-squared error (MSE)

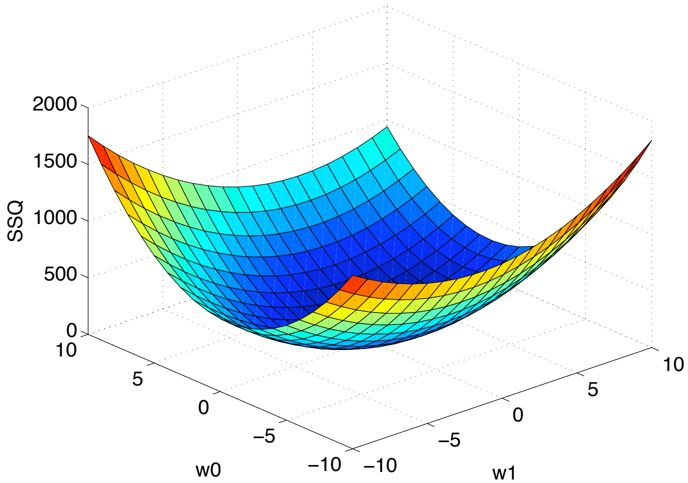
- Recall the least-square solution: $\hat{\mathbf{w}} = (X^TX)^{-1} X^T Y$
- What if X is too big to compute this explicitly (e.g. $m \sim 10^6$)?
- Go back to the gradient step: $Err(\mathbf{w}) = (Y - X\mathbf{w})^T(Y - X\mathbf{w})$

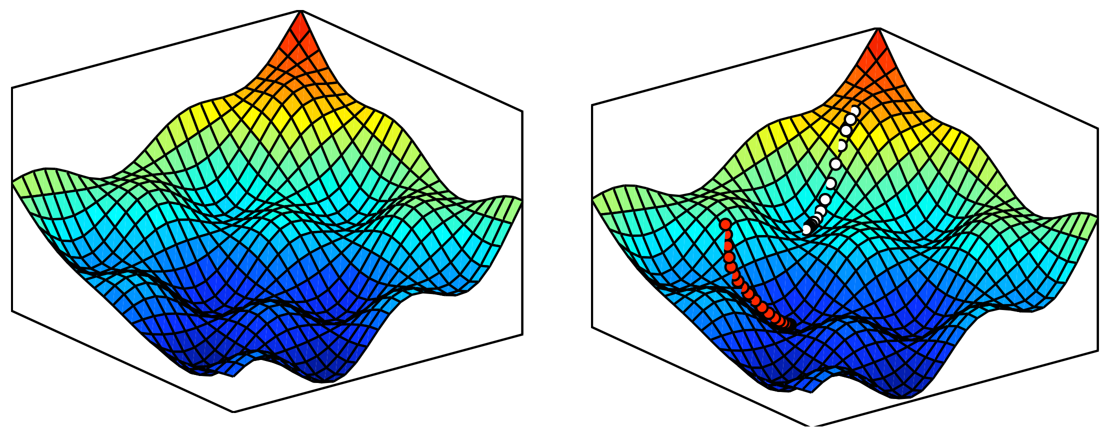
$$\partial Err(\mathbf{w})/\partial \mathbf{w} = -2 X^T (Y - X\mathbf{w})$$

$$\partial Err(\mathbf{w})/\partial \mathbf{w} = 2(X^TX\mathbf{w} - X^TY)$$

Gradient-descent solution for MSE

- Consider the error function:





- The gradient of the error is a vector indicating the direction to the minimum point.
- Instead of directly finding that minimum (using the closed-form equation), we can take small steps towards the minimum.

Gradient-descent solution for MSE

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

- The algorithm:

  Given an initial weight vector $\mathbf{w}_0$,
  Do for $k=1, 2, \ldots$
  $$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \, \partial Err(\mathbf{w}_k)/\partial \mathbf{w}_k$$
  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 $\boldsymbol{w}_k$ may oscillate forever.
  - This suggests that $\alpha_k \rightarrow 0$ as $k \rightarrow \infty$.

- If steps are too **small**: the $\boldsymbol{w}_k$ 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 $\boldsymbol{w}_k$ to a **<u>local minimum</u>** of the error function.

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

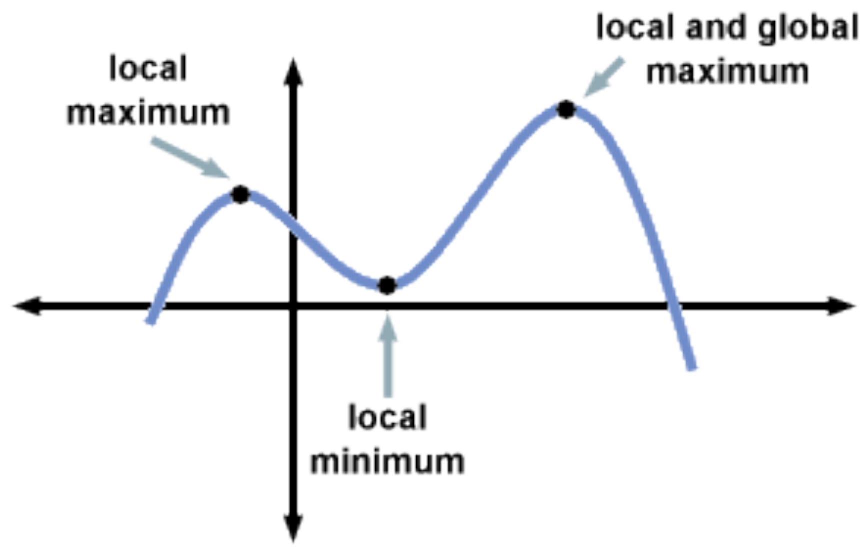
E.g. $\alpha_k = 1/2$ for $k = 1, \ldots, T$

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

etc.

Local minima

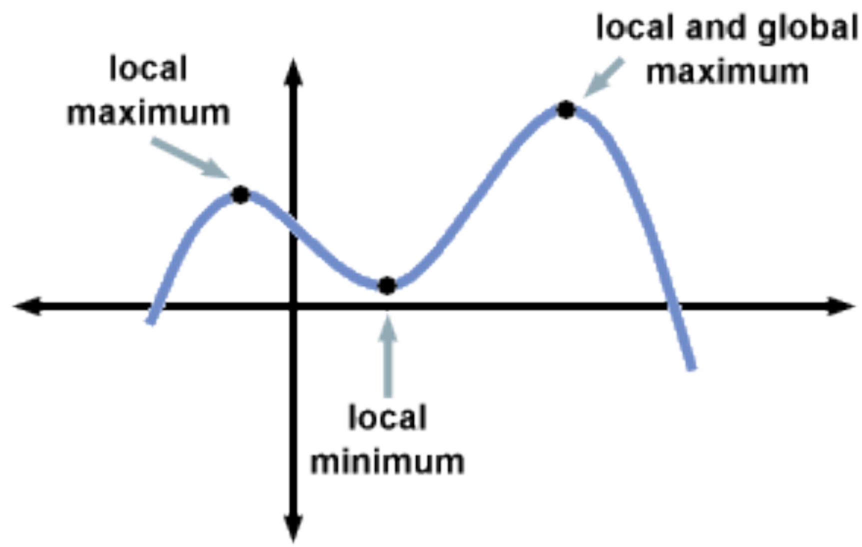
- Convergence is **<u>NOT</u>** to a global minimum, only to local minimum.



- The blue line represents the error function. There is *<u>no guarantee</u>* regarding the amount of error of the weight vector found by gradient descent, compared to the globally optimal solution.

Local minima

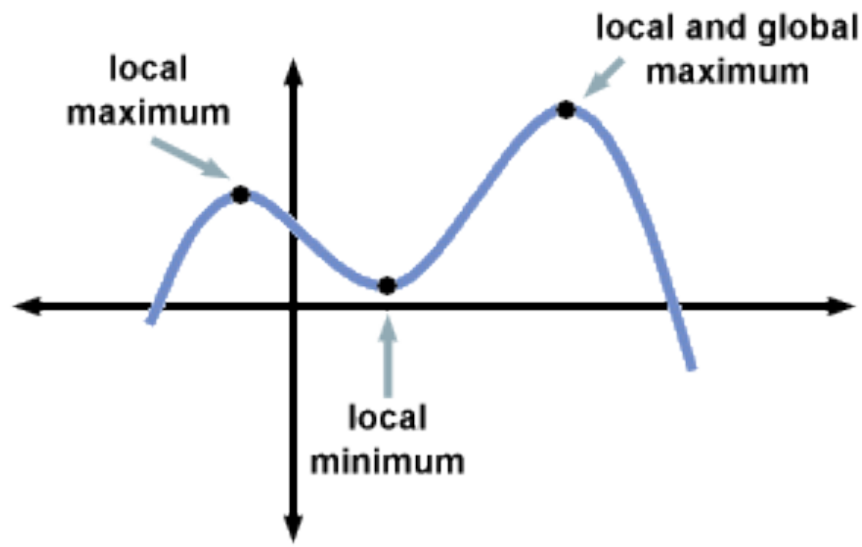
- Convergence is **NOT** to a global minimum, only to local minimum.



- For linear function approximations using Least-Mean Squares (LMS) error, this is not an issue: only **ONE** global minimum!
  - Local minima affects many other function approximators.

Local minima

- Convergence is **<u>NOT</u>** to a global minimum, only to local minimum.



- For linear function approximations using Least-Mean Squares (LMS) error, this is not an issue: only **ONE** global minimum!
  - Local minima affects many other function approximators.
- *<u>Repeated random restarts</u>* can help (in all cases of gradient search).

A 3rd optimization method: QR decomposition (optional)

- Consider the usual criteria: $X^T(Y - X\mathbf{w})=0$
- Assume X can be decomposed: $X = QR$

  where Q is an nxm orthogonal matrix (i.e. $Q^TQ=I$), and R is an mxm upper triangular matrix.

- Replace X in equation above: $(QR)^TY = (QR)^T(QR)\mathbf{w}$
- Distribute the transpose: $R^TQ^TY = R^TQ^TQR\mathbf{w}$
- Let $Q^TQ=I$ and multiply by $(R^T)^{-1}$ $Q^TY = R\mathbf{w}$
- Solution: $\hat{\mathbf{w}} = R^{-1}Q^TY$ The fitted outputs are: $\hat{Y} = QQ^TY$
- This method is more numerically stable than others, and R^{-1} is fast to compute because upper triangular.
- Alternately, we can use **singular value decomposition**.

What you should know

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

To-do

- Reproduce the linear regression example (slides 15-18), solving it using the software of your choice.
- 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 **midterm** date in agenda: Nov. 22, 6-8pm, Leacock 132.
- Tutorial times (appearing soon): www.cs.mcgill.ca/~jpineau/comp551/schedule.html
- Office hours (confirmed): www.cs.mcgill.ca/~jpineau/comp551/syllabus.html