Gradient Descent

Michail Michailidis & Patrick Maiden

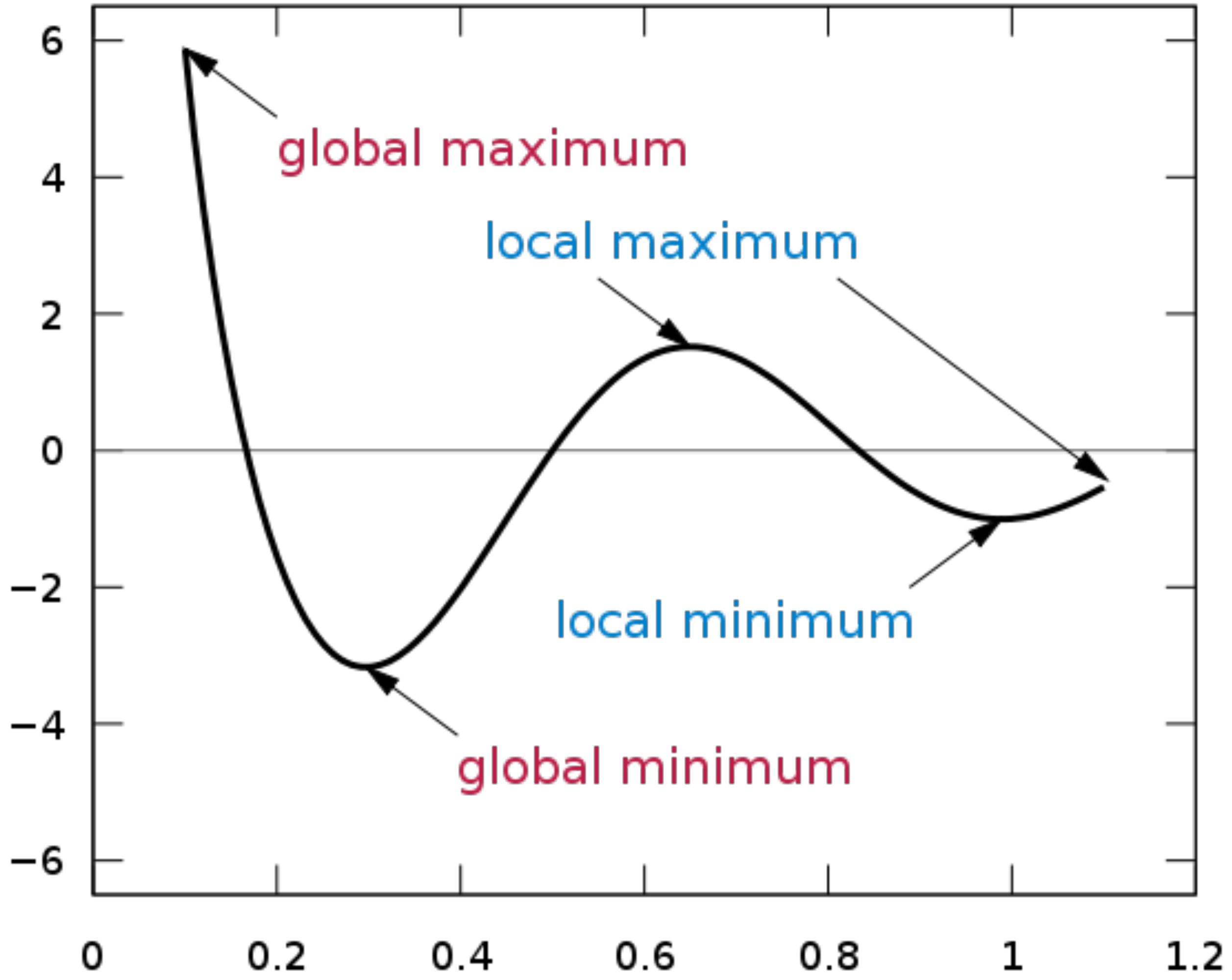
Outline

- Motivation
- Gradient Descent Algorithm
  - Issues & Alternatives
- Stochastic Gradient Descent
- Parallel Gradient Descent
- HOGWILD!

Motivation

- It is good for finding global minima/maxima if the function is convex
- It is good for finding local minima/maxima if the function is not convex
- It is used for optimizing many models in Machine learning:
    - **It is used in conjunction with:**
        - Neural Networks
        - Linear Regression
        - Logistic Regression
        - Back-propagation algorithm
        - Support Vector Machines

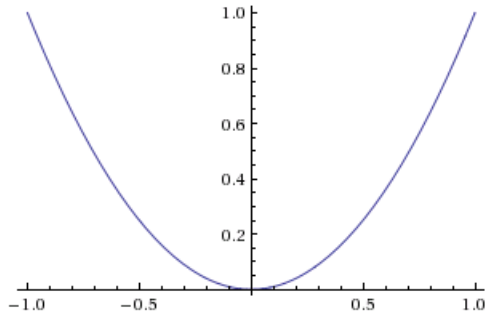
Function Example



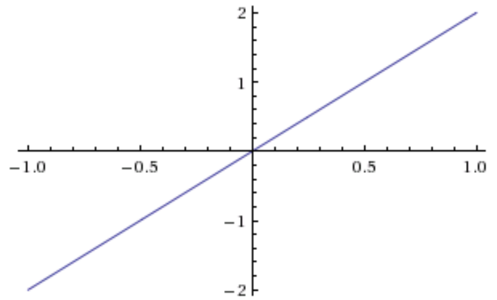
Quickest ever review of multivariate calculus

- Derivative
- Partial Derivative
- Gradient Vector

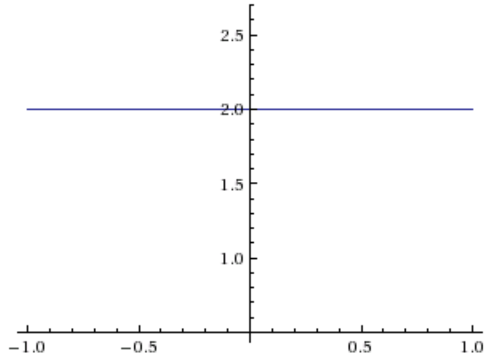
Derivative



$f(x)=x\uparrow 2$



$f'(x)=df/dx=2x$



$f''(x)=d\uparrow 2\, f/dx=2$

- Slope of the tangent line

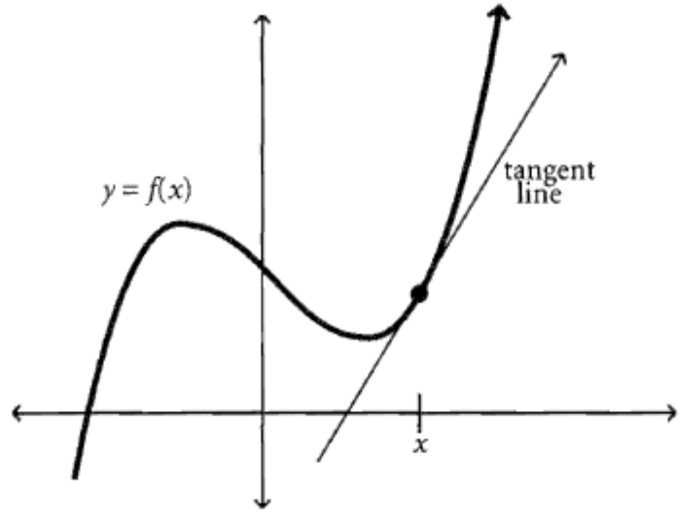
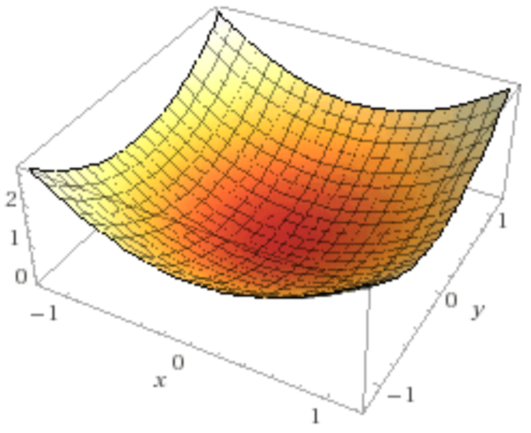


Figure 6.2

- Easy when a function is univariate

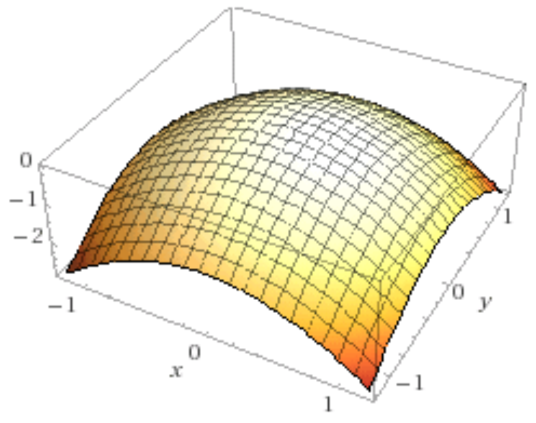
Partial Derivative – Multivariate Functions

For multivariate functions (e.g two variables) we need partial derivatives – one per dimension. Examples of multivariate functions:



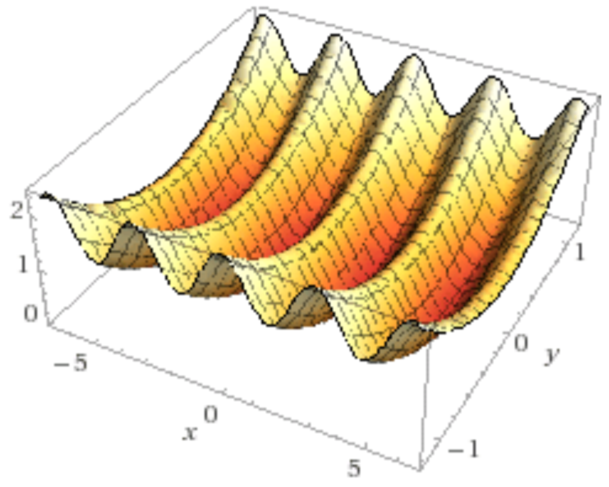
$f(x,y) = x^2 + y^2$

Convex!

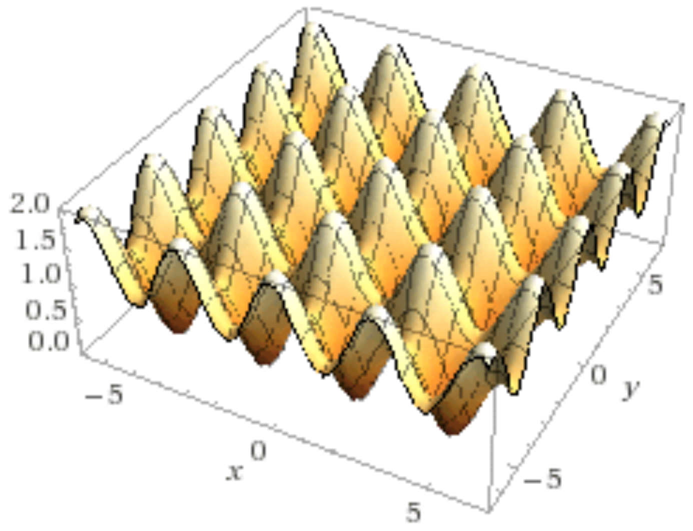


$f(x,y) = -x^2 - y^2$

Concave!



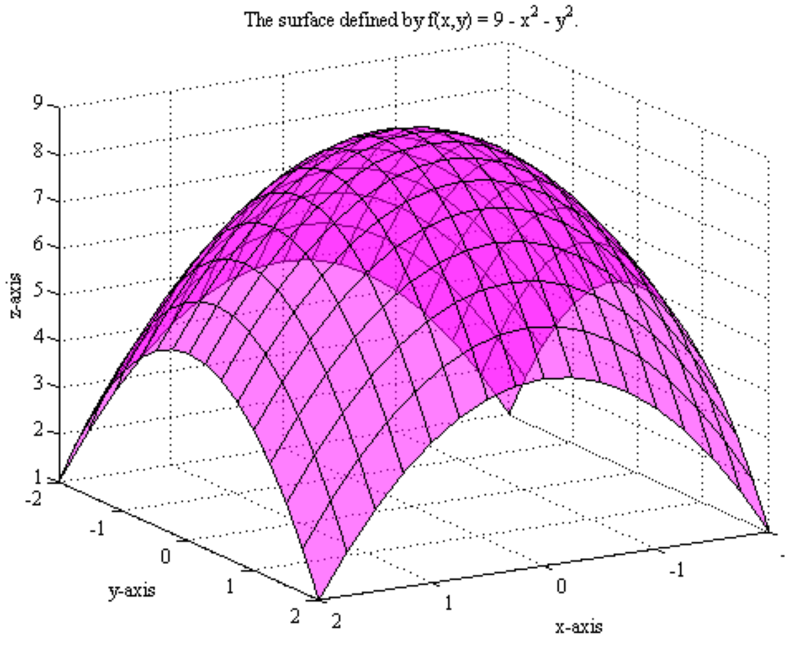
$f(x,y) = \cos^2(x) + y^2$

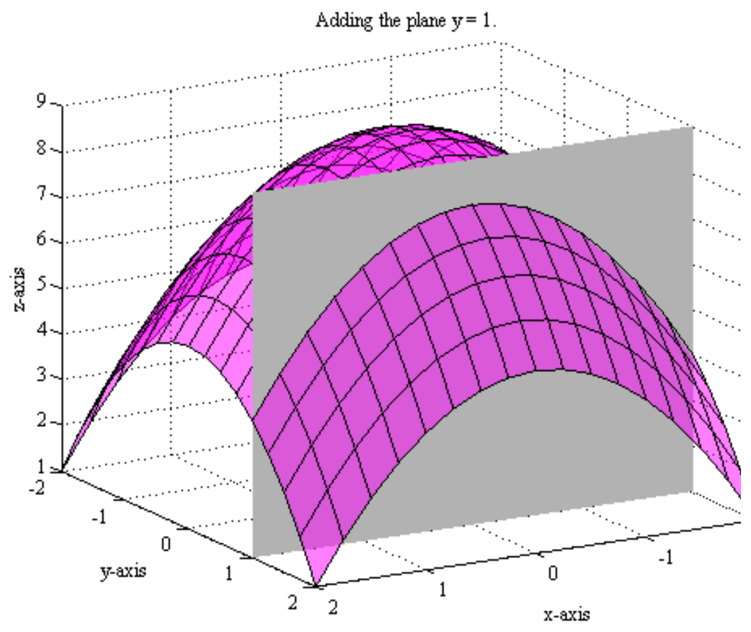


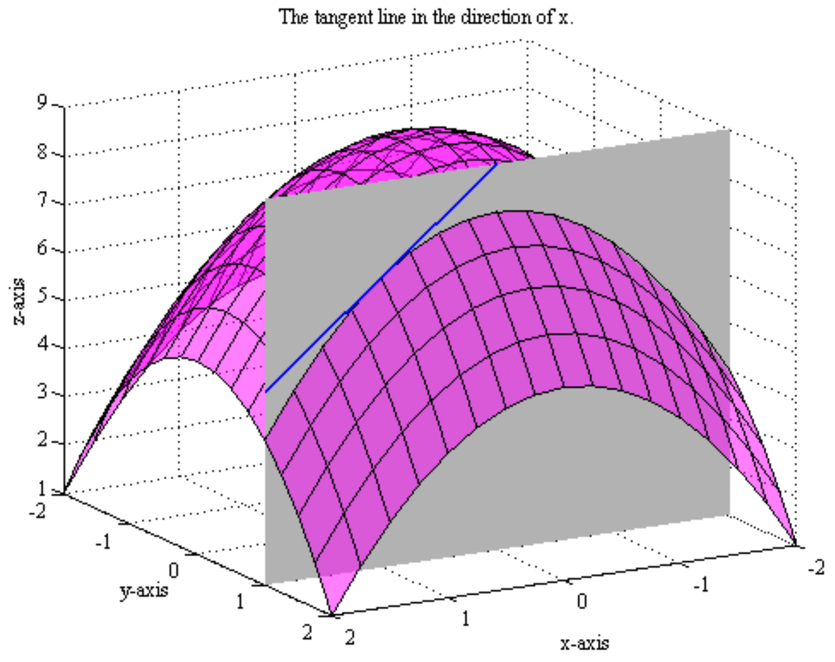
$f(x,y) = \cos^2(x) + \cos^2(y)$

Partial Derivative – Cont'd

To visualize the partial derivative for each of the dimensions x and y, we can imagine a plane that "cuts" our surface along the two dimensions and once again we get the slope of the tangent line.







surface: $f(x,y)=9-x^2-y^2$

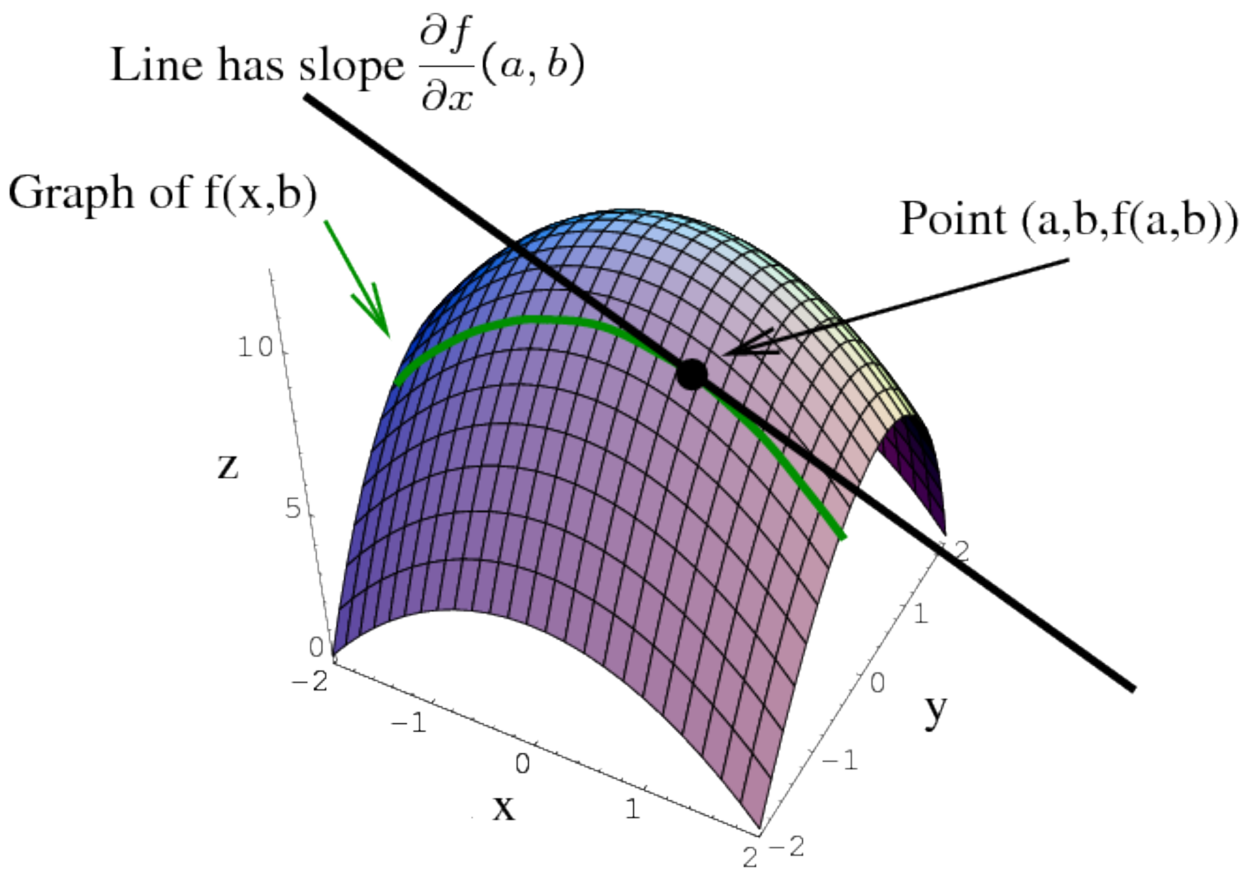
plane: $y=1$

cut: $f(x,1)=8-x^2$

slope / derivative of cut: $f'(x)=-2x$

Partial Derivative – Cont'd 2

If we partially differentiate a function with <u>respect to x</u>, we pretend <u>y is constant</u>



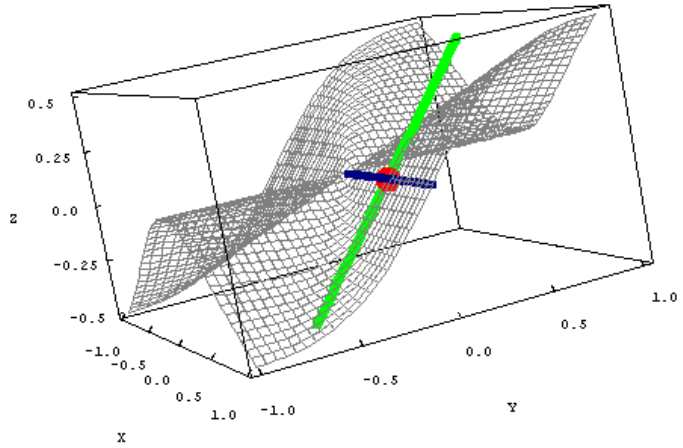
$$f(x,y)=9-x^2-y^2$$

$$f(x,y)=9-x^2-c^2$$

$$f_x=\partial f/\partial x=-2x$$

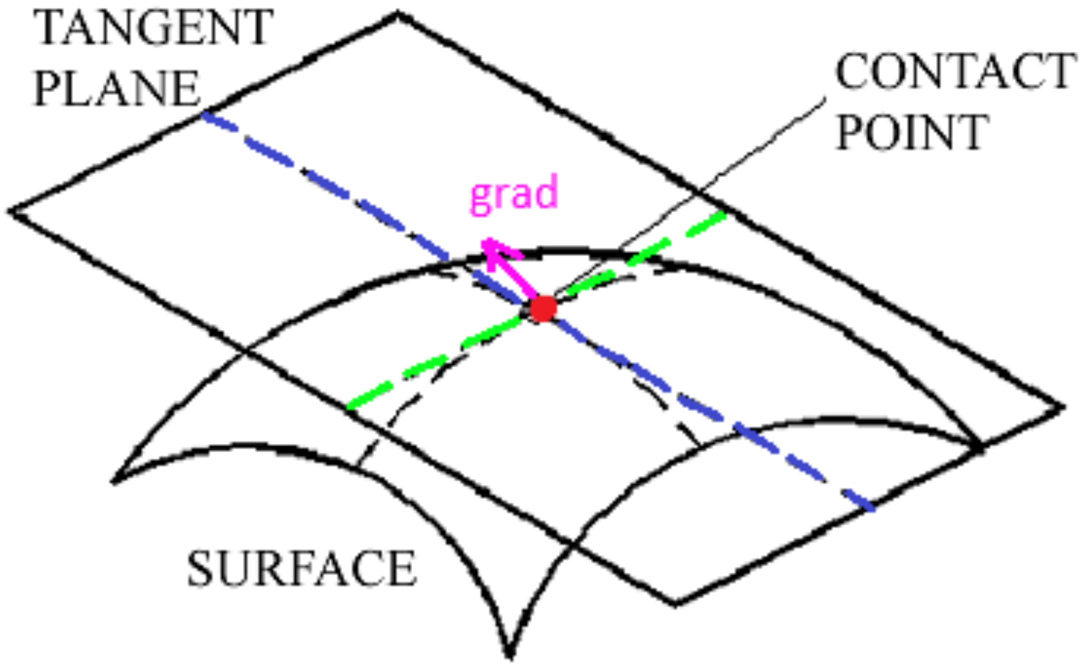
$$f(x,y)=9-c^2-y^2$$

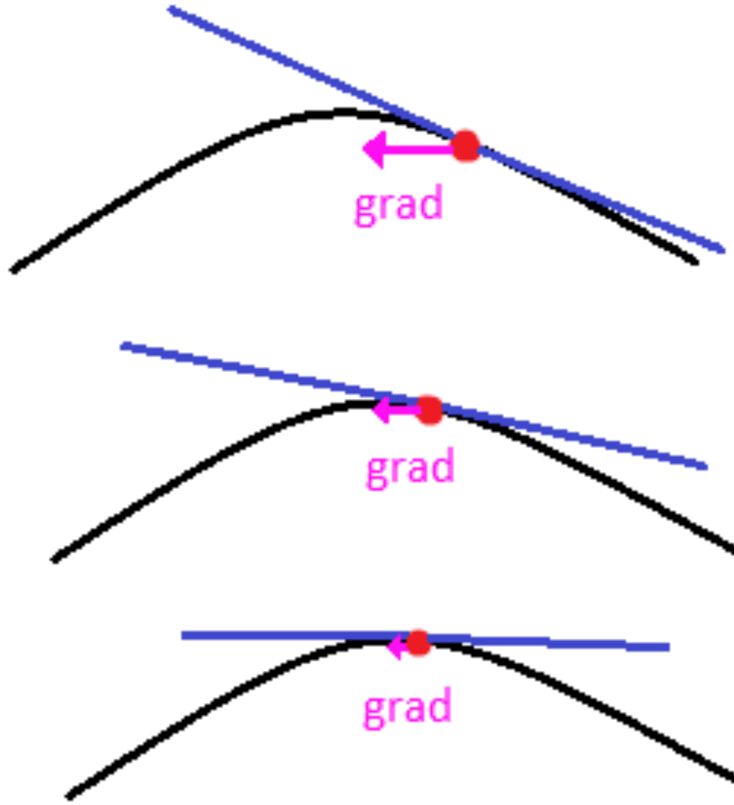
$$f_y=\partial f/\partial y=-2y$$



Partial Derivative – Cont'd 3

The two tangent lines that pass through a point, define the tangent plane to that point





Gradient Vector

- Is the vector that has as coordinates the partial derivatives of the function:

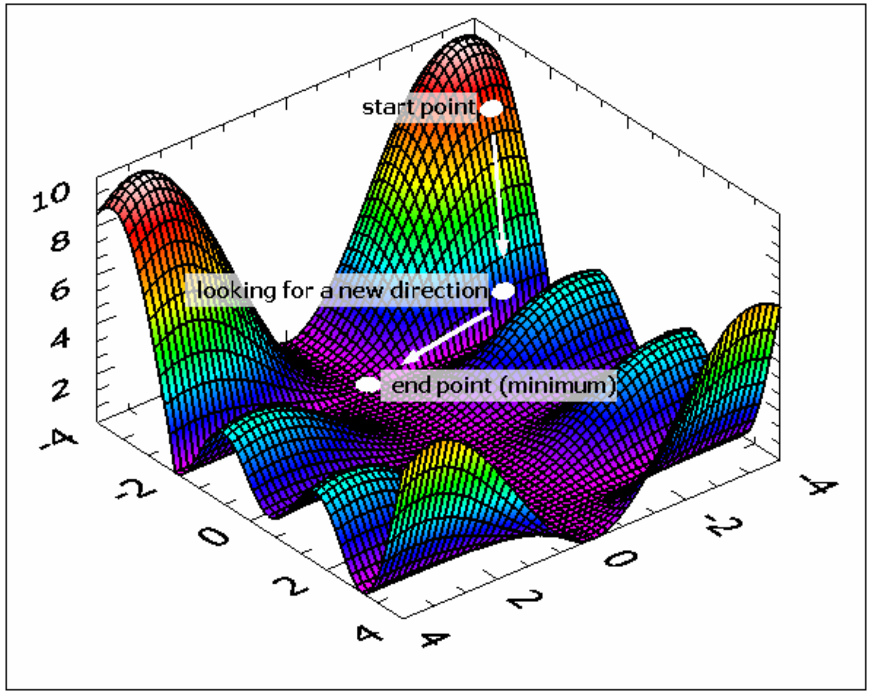
$$f(x,y)=9-x^2-y^2 \qquad \partial f/\partial x=-2x \quad \partial f/\partial y=-2y$$

$$\nabla f=\partial f/\partial x\, i+\partial f/\partial y\, j=(\partial f/\partial x, \partial f/\partial y)=(-2\mathrm{x}, -2\mathrm{y})$$

- **Note: Gradient Vector is not parallel to tangent surface**

Gradient Descent Algorithm & Walkthrough

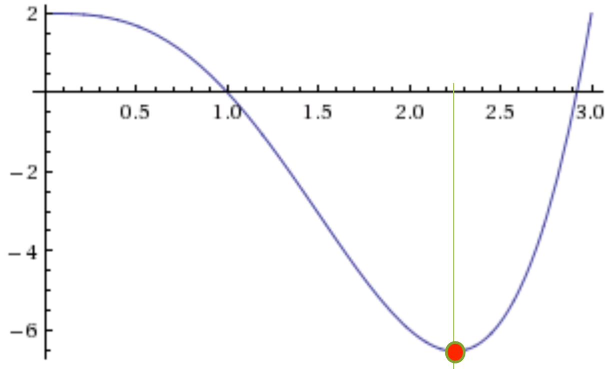
- Idea
  - Start somewhere
  - Take steps based on the gradient vector of the current position till convergence
- Convergence :
  - happens when change between two steps < ε



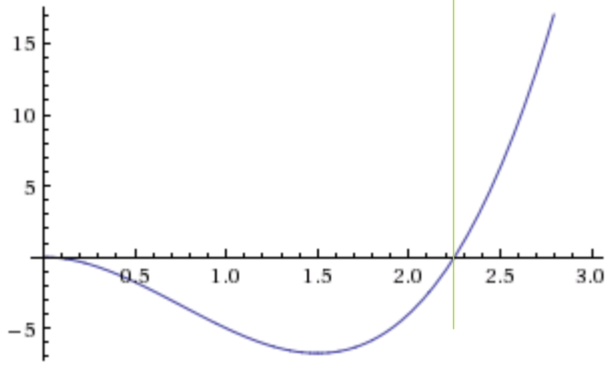
Gradient Descent Code (Python)

```
# From calculation, we expect that the local minimum occurs at x=9/4

x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_prime(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_prime(x_old)
print "Local minimum occurs at ", x_new
```

$f'(x)=4x^3-9x^2$

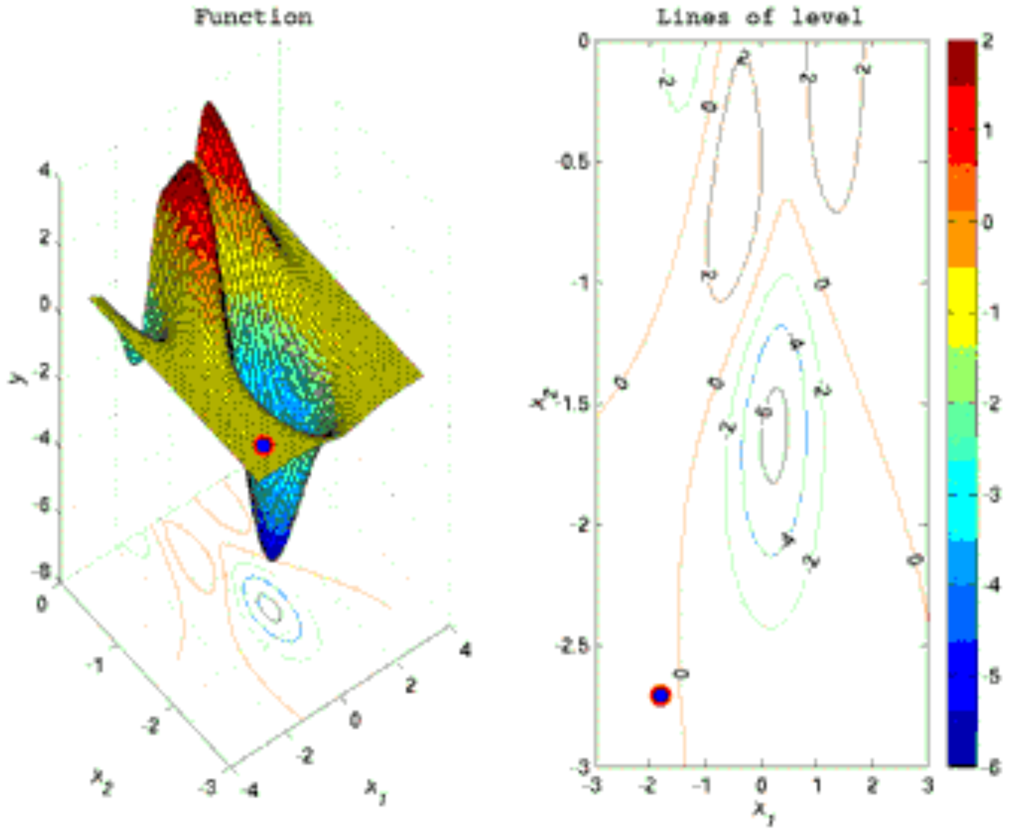


$f(x)=x^4-3x^3+2$



$f'(x)=4x^3-9x^2$

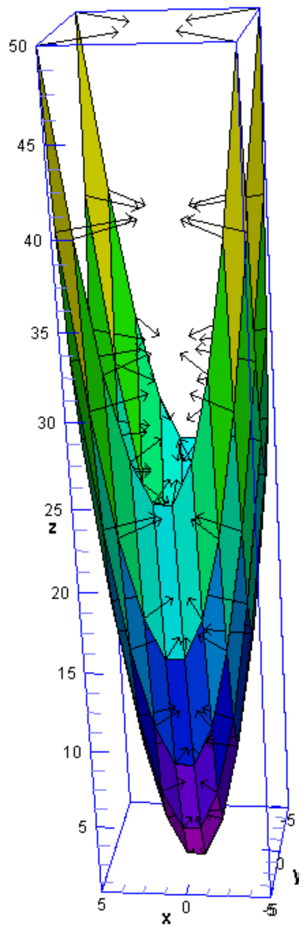
Gradient Descent Algorithm & Walkthrough

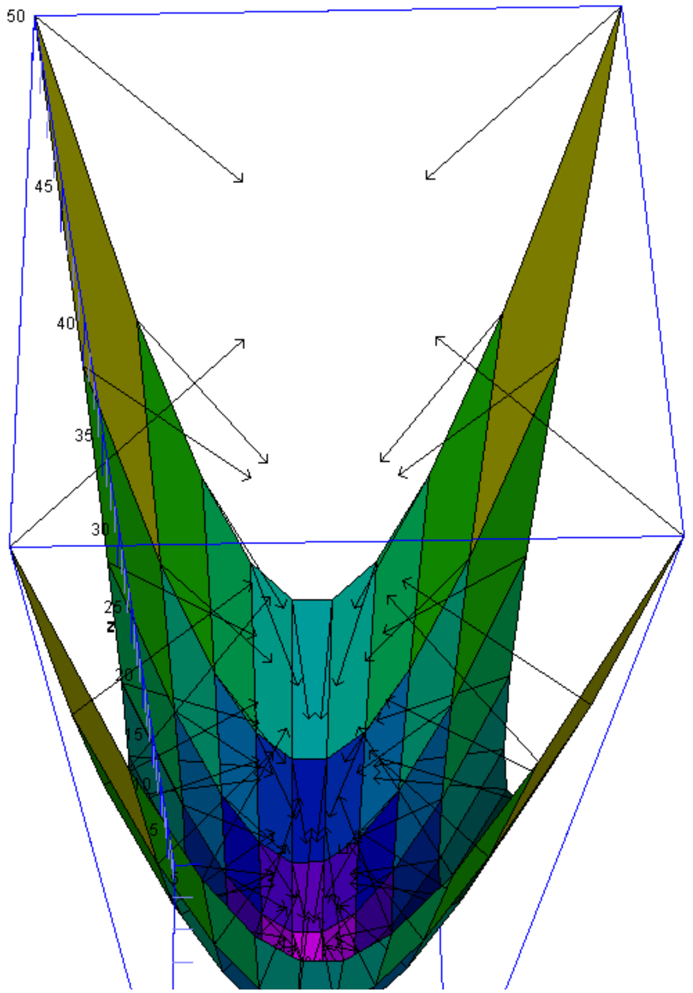


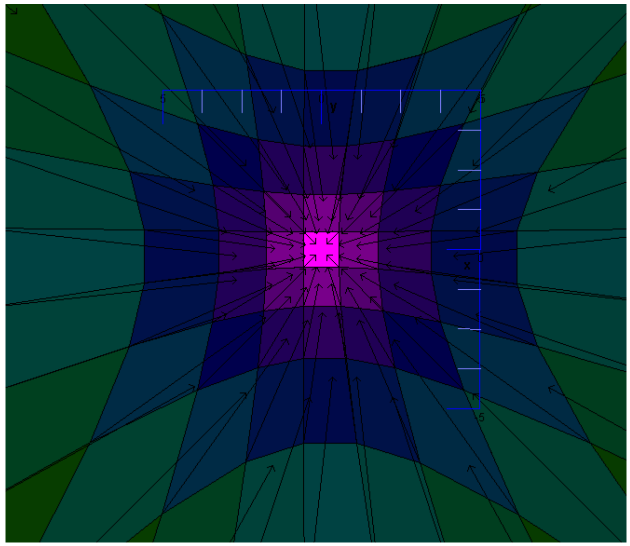
Potential issues of gradient descent - Convexity

We need a convex function
→ so there is a global minimum:

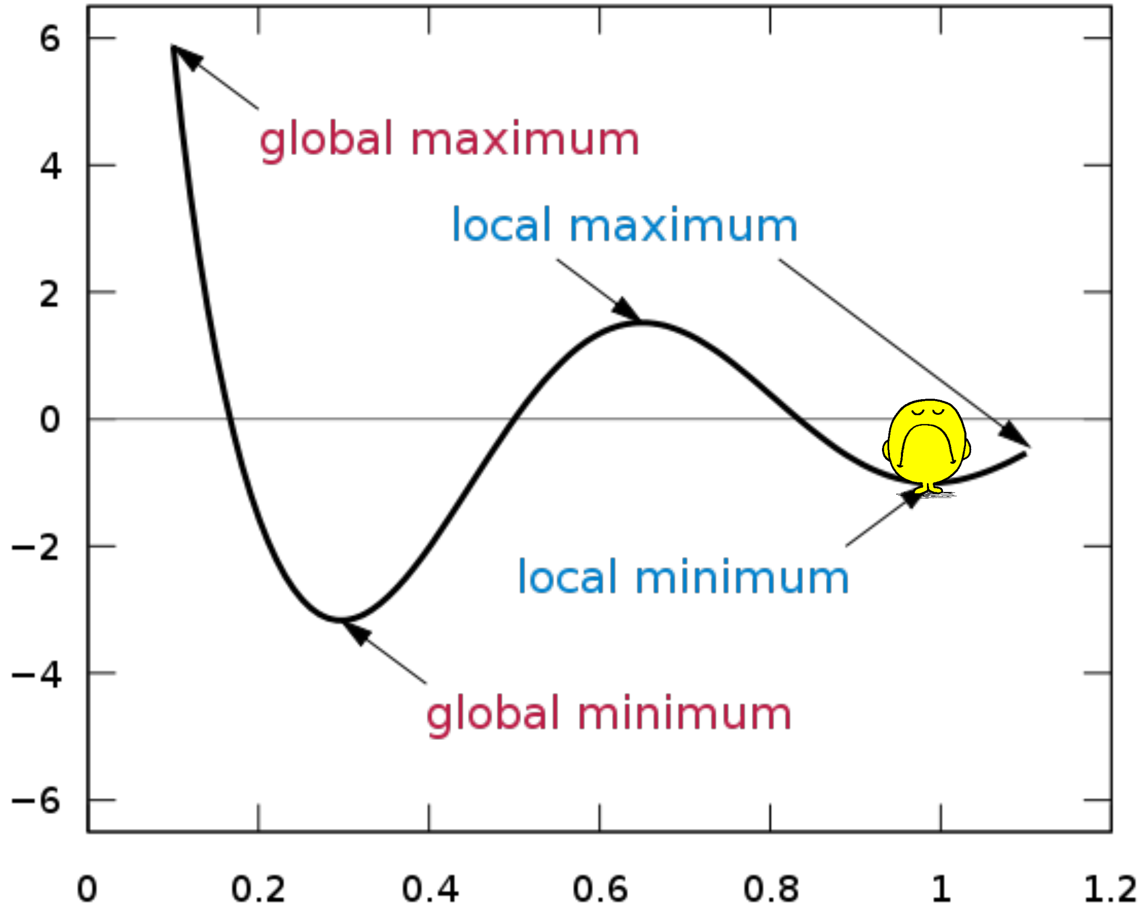
$$f(x,y)=x^2+y^2$$







Potential issues of gradient descent – Convexity (2)



Potential issues of gradient descent – Step Size

- As we saw before, one parameter needs to be set is the step size
- Bigger steps leads to faster convergence, right?

Alternative algorithms

- Newton’s Method
  - Approximates a polynomial and jumps to the min of that function
  - Needs Hessian
- BFGS
  - More complicated algorithm
  - Commonly used in actual optimization packages

Stochastic Gradient Descent

- Motivation
  - One way to think of gradient descent is as a minimization of a sum of functions:
    - $w = w - \alpha \nabla L(w) = w - \alpha \sum \nabla L_i(w)$
      - (L_i is the loss function evaluated on the i-th element of the dataset)
    - On large datasets, it may be computationally expensive to iterate over the whole dataset, so pulling a subset of the data may perform better
    - Additionally, sampling the data leads to “noise” that can avoid finding “shallow local minima.” This is good for optimizing non-convex functions. (Murphy)

Stochastic Gradient descent

- Online learning algorithm
- Instead of going through the entire dataset on each iteration, randomly sample and update the model

```
Initialize w and α
Until convergence do:
  Sample one example i from dataset //stochastic portion
  w = w - $\alpha \nabla L_i(w)$
return w
```

Stochastic Gradient descent (2)

- Checking for convergence after each data example can be slow
- One can simulate stochasticity by reshuffling the dataset on each pass:

```
Initialize w and α
Until convergence do:
  shuffle dataset of n elements //simulating stochasticity
  For each example i in n:
    w = w - $\alpha \nabla L_i(w)$
return w
```

- This is generally faster than the classic iterative approach ("noise")
- However, you are still passing over the entire dataset each time
- An approach in the middle is to sample "batches", subsets of the entire dataset
  - This can be parallelized!

Parallel Gradient descent

- Training data is chunked into batches and distributed

```
Initialize w and α
Loop until convergence:
    generate randomly sampled chunk of data m
    on each worker machine v:
        $\nabla L_v(w) = sum(\nabla L_i(w))$ // compute gradient on batch
    $w = w - \alpha * sum(\nabla L_v(w))$ //update global w model
return w
```

HOGWILD! (Niu, et al. 2011)

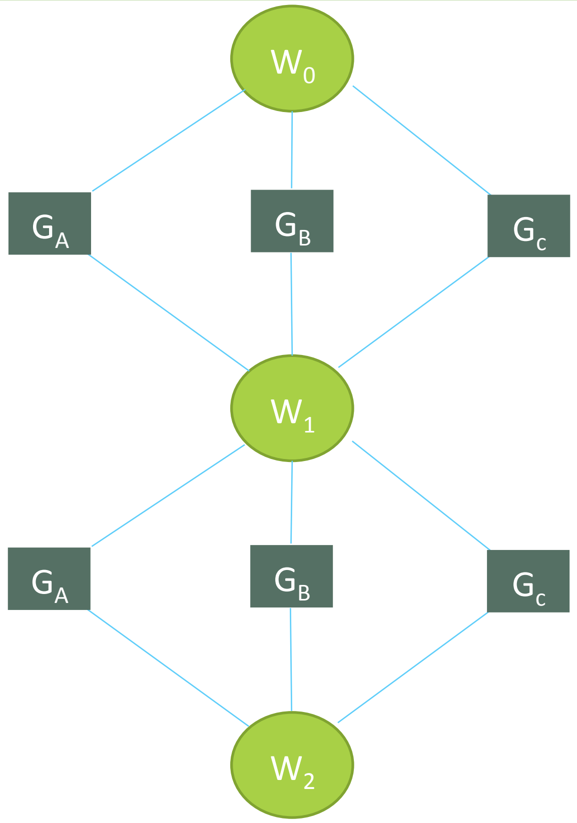
- Unclear why it is called this
- Idea:
  - In Parallel SGD, each batch needs to finish before starting next pass
  - In HOGWILD!, share the global model amongst all machines and update on-the-fly
    - No need to wait for all worker machines to finish before starting next epoch
    - Assumption: component-wise addition is atomic and does not require locking

HOGWILD! - Pseudocode

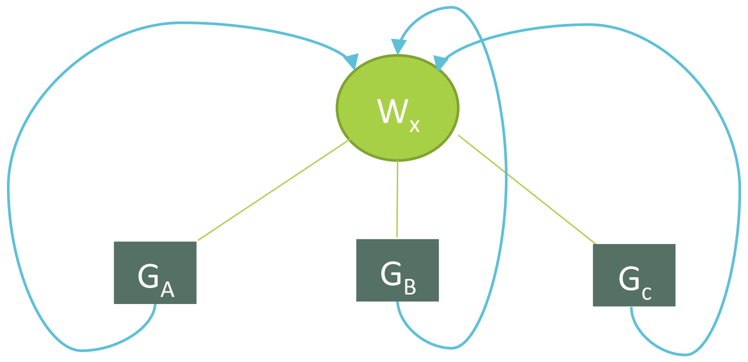
```
Initialize global model w
On each worker machine:
     loop until convergence:
          draw a sample e from complete dataset E
          get current global state w and compute $\nabla L_e(w)$
          for each component i in e:
               $w_i = w_i - \alpha b_v^T \nabla L_e(w)$ // $b_v$ is vth std. basis component
          update global w
return w
```

Comparison

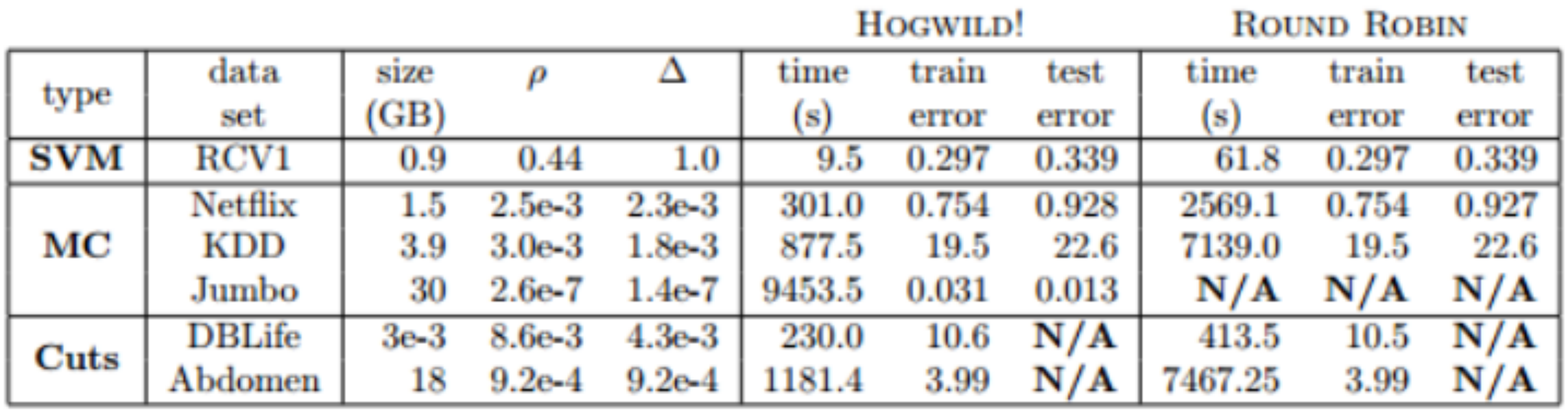
Parallel SGD



HOGWILD!



Comparison



					Hogwild!			Round Robin		
type	data set	size (GB)	ρ	Δ	time (s)	train error	test error	time (s)	train error	test error
SVM	RCV1	0.9	0.44	1.0	9.5	0.297	0.339	61.8	0.297	0.339
MC	Netflix	1.5	2.5e-3	2.3e-3	301.0	0.754	0.928	2569.1	0.754	0.927
	KDD	3.9	3.0e-3	1.8e-3	877.5	19.5	22.6	7139.0	19.5	22.6
	Jumbo	30	2.6e-7	1.4e-7	9453.5	0.031	0.013	**N/A**	**N/A**	**N/A**
Cuts	DBLife	3e-3	8.6e-3	4.3e-3	230.0	10.6	**N/A**	413.5	10.5	**N/A**
	Abdomen	18	9.2e-4	9.2e-4	1181.4	3.99	**N/A**	7467.25	3.99	**N/A**

Figure 2: Comparison of wall clock time across of Hogwild! and RR. Each algorithm is run for 20 epochs and parallelized over 10 cores.

- RR – Round Robin
  - Each machine updates x as it comes in. Wait for all before starting next pass
- AIG
  - Like Hogwild but does fine-grained locking of variables that are going to be used

Comparison (2)

SVM Graph Cuts

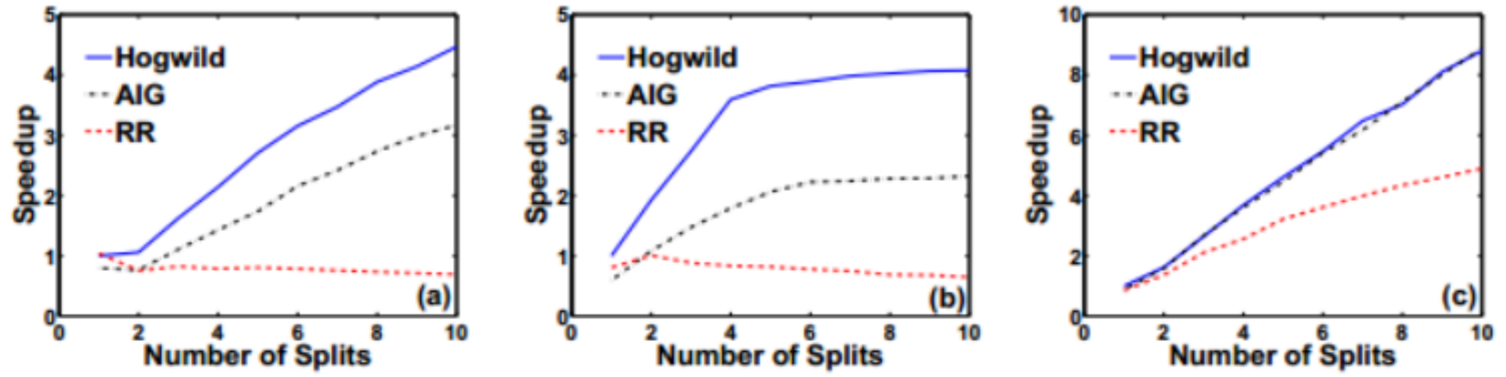


Figure 3: Total CPU time versus number of threads for (a) RCV1, (b) Abdomen, and (c) DBLife.

Matrix Completion

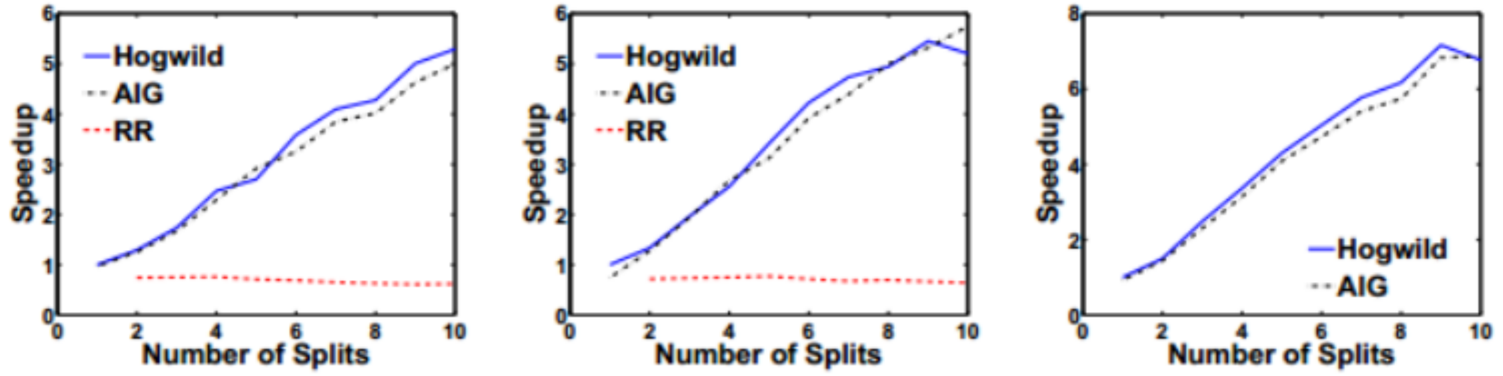


Figure 4: Total CPU time versus number of threads for the matrix completion problems (a) Netflix Prize, (b) KDD Cup 2011, and (c) the synthetic Jumbo experiment.

Moral of the story

- Having an idea of how gradient descent works informs your use of others' implementations
- There are very good implementations of the algorithm and other approaches to optimization in many languages
- Packages:
  - Python
    - NumPy/SciPy
  - Matlab
    - Matlab Optimization toolbox
    - Pmtk3
  - R
    - General-purpose optimization: optim()
    - R Optimization Infrastructure (ROI)
  - TupleWare
    - Coming soon….

Resources

Partial Derivatives:

- http://msemac.redwoods.edu/~darnold/math50c/matlab/pderiv/index.xhtml
- http://mathinsight.org/nondifferentiable_discontinuous_partial_derivatives
- http://www.sv.vt.edu/classes/ESM4714/methods/df2D.html
- Gradients Vector Field Interactive Visualization: http://dlippman.imathas.com/g1/Grapher.html from https://www.khanacademy.org/math/calculus/partial_derivatives_topic/gradient/v/gradient-1
- http://simmakers.com/wp-content/uploads/Soft/gradient.gif

Gradient Descent:

- http://en.wikipedia.org/wiki/Gradient_descent
- http://www.youtube.com/watch?v=5u4G23_OohI (Stanford ML Lecture 2)
- http://en.wikipedia.org/wiki/Stochastic_gradient_descent
- Murphy, *Machine Learning, a Probabilstic Perspective*, 2012, MIT Press
- Hogwild paper: http://pages.cs.wisc.edu/~brecht/papers/hogwildTR.pdf