Parallel and Distributed Computing

10-605

February 14, 2012

Announcement

- Assignment 3 is due tomorrow
- Filter bigrams that have stop words in them from your final results
- Aside: why are there stop words in the final results?

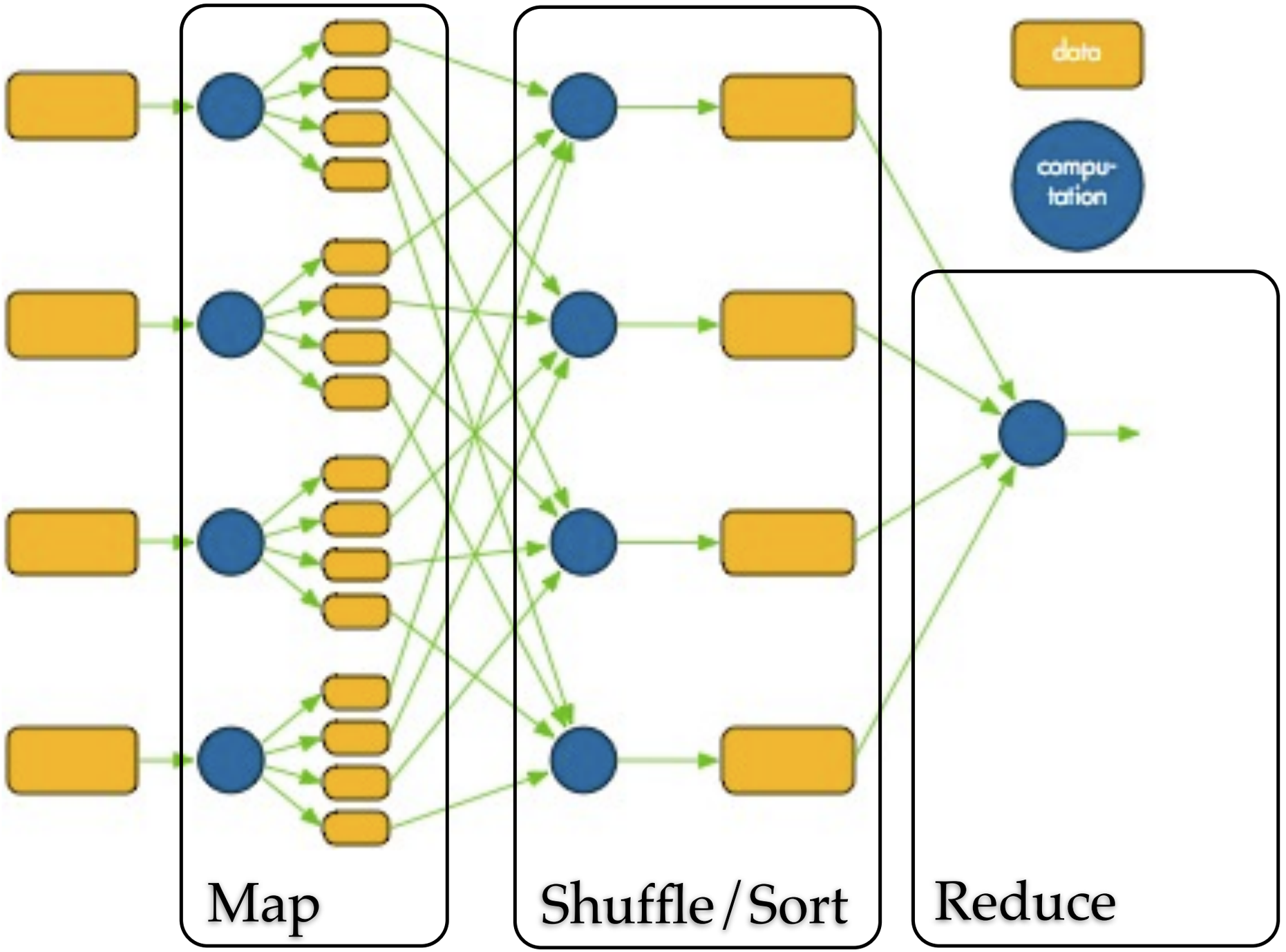
It's a Revolution, Y'all

- The Age of Big Data - NYT Sunday Review, Feb 11, 2012
- "GOOD with numbers? Fascinated by data? The sound you hear is opportunity knocking."
- "'It's a revolution,' says Gary King, director of Harvard's Institute for Quantitative Social Science."

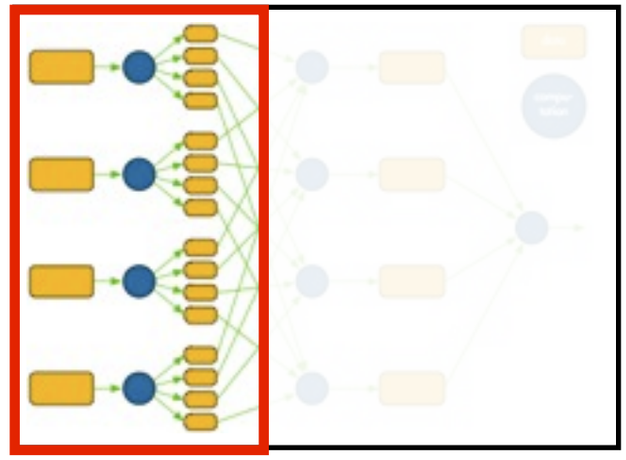
Outline

- MR review
- MR tips
- HDFS
- AWS/Hadoop demo
- Multithreaded programming
  - Parallel Lasso

Mapreduce: Review



MR: Map



Input

Joe likes toast

Map 1

Jane likes toast with jam

Map 2

Joe burnt the toast

Map 3

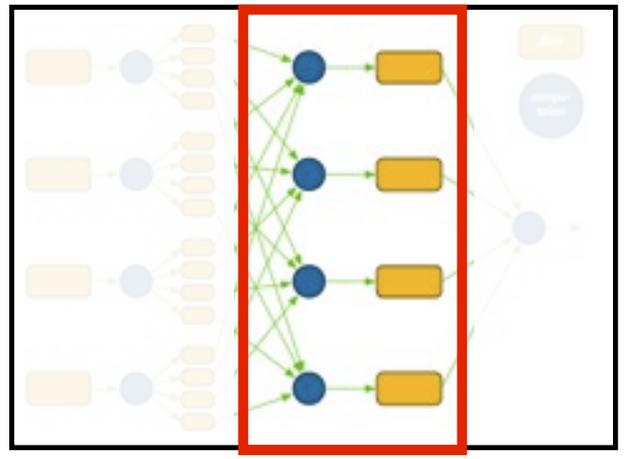
Output

Joe	1
likes	1
toast	1

Jane	1
likes	1
toast	1
with	1
jam	1

Joe	1
burnt	1
the	1
toast	1

MR: Sort



Input

Joe	1
likes	1
toast	1

Jane	1
likes	1
toast	1
with	1
jam	1

Joe	1
burnt	1
the	1
toast	1

Output

Joe	1
Joe	1

Jane	1

likes	1
likes	1

toast	1
toast	1
toast	1

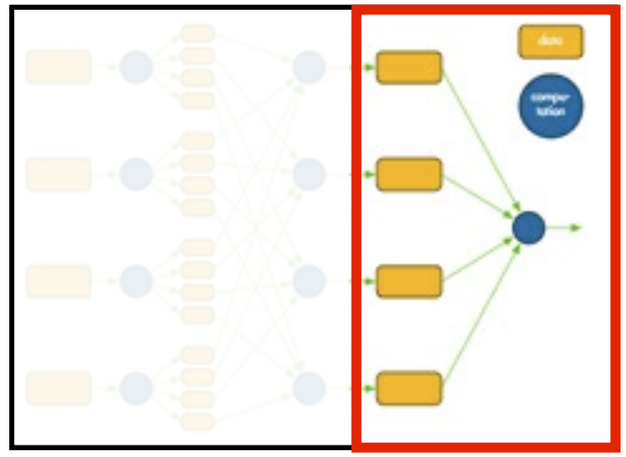
with	1

jam	1

burnt	1

the	1

MR: Reduce



Input

Joe 1 Joe 1	Reduce 1
Jane 1	Reduce 2
likes 1 likes 1	Reduce 3
toast 1 toast 1 toast 1	Reduce 4
with 1	Reduce 5
jam 1	Reduce 6
burnt 1	Reduce 7
the 1	Reduce 8

Output

Joe 2
Jane 1
likes 2
toast 3
with 1
jam 1
burnt 1
the 1

Mapreduce Tips

- Mapreduce is a simple idea
  - but there are still tricks

Mapreduce Tips

- Throw an exception when things go wrong.
  - Counter to typical software engineering practices
- Failing loudly and early will save you time and money
  - Exceptions are passed all the way up to the controller and you can see it in its stderr file.
  - Otherwise, debugging a mapreduce can be hard

Mapreduce Tips

- Input formats have keys.
- We totally ignored them in Word Count

```
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {

 String line = value.toString();
 StringTokenizer tokenizer = new StringTokenizer(line);
 while (tokenizer.hasMoreTokens()) {
     word.set(tokenizer.nextToken());
     context.write(word, one);
     }
 }
```

Mapreduce Tips

- TextInputFormat (for raw text files, for example)
  - Key
    - line number/file pos
    - LongWritable
  - Value
    - Text
- TextOutputFormat
  - Programmatically can be any type
  - On disk, will have tab separated key/values
  - key/values defined by the “write” method

Mapreduce Tips

- SequenceFileInputFormat (binary)
  - key
    - What you outputted from Reducer
    - Any writable type
  - Value
    - Any writable type
- SequenceFiles are more useful if they will be feeding into another mapreduce
- Runtime errors if types of key/values in input and types of Mapper<k,v,k,v> don't match

HDFS

- Hadoop operates on top of a distributed file system
- Your data is where you are
  - or rather, you are where your data is
- Ahead of time, data is replicated across many locations

HDFS

- Mostly transparent
- To access data on HDFS on AWS you need to look on s3://
  - E.g. uri = s3://bigml-shared/

```
FileSystem fileSystem;
 try {
    fileSystem = FileSystem.get(new URI(uri), context
            .getConfiguration());
} catch (URISyntaxException e) {
    e.printStackTrace();
    return;
}
         Path path = new Path(f);
if (!fileSystem.exists(path)) {
    System.out.println("File does not exist: " + f);
    return;
}
FSDataInputStream inf = fileSystem.open(path);
BufferedReader in = new BufferedReader(new InputStreamReader(inf));
```

Demo time

- Web interface
- http://docs.amazonwebservices.com/ElasticMapReduce/latest/DeveloperGuide/UsingtheHadoopUserInterface.html
- Progress
- Counters
- Failures

Multithreaded Programming

Not everything is a MR

- MRs are ideal for "embarrassingly parallel" problems
- Very little communication
- Easily distributed
- Linear computational flow

Parallel programming

- What if you need more synchronization?
  - Future results depend on past

Why synchronize

- We saw that in the perceptron algorithm, collecting results together is beneficial
- What if you need to communicate more often than once every pass through the data?

Why synchronize

- Synchronization is non-trivial without the proper constructs

Without Synchronization

- Thread A
  - Read variable X
  - compute X+1
  - Assign to X
- Thread B
  - Read variable X
  - compute X+1
  - Assign to X

Without Synchronization

- Depending on order, final value can be different
- A and B could both read the initial value of X and then overwrite each other
  - x = x+1
- A reads and writes, then B reads and writes
  - x = x + 2

Without Synchronization

- That's a race condition
  - and you will grow to hate them
    - if you don't already

How to Sync

- There are several ways to synchronize and avoid race conditions

How to Sync

- Simplest is mutex
  - mutually exclusive
- Usually associated with a variable or code block

```
mutex_t count_lock;

mutex_lock(&count_lock);

mutex_unlock(&count_lock);
```

Deadlock

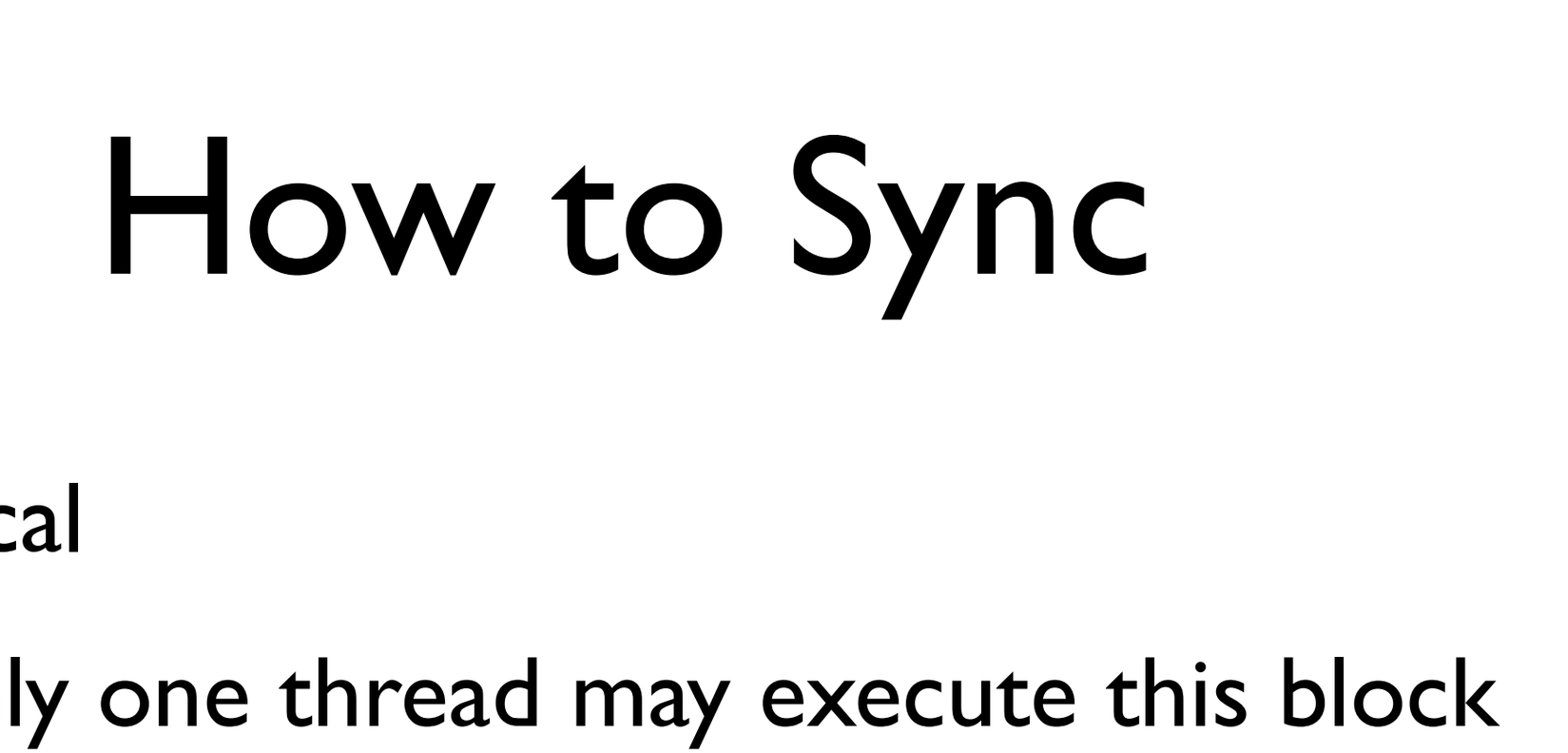
- Thread A and B both need locks for variables X and Y
  - A acquires mutex_X
  - B acquires mutex_Y
  - A waits for mutex_Y to be free
  - B waits for mutex_X to be free

Higher Level Syncing

- OpenMP has predefined locking paradigms
  - reduce the chance of deadlock
  - not foolproof
    - though possibly fool resistant

How to Sync

- Critical
  - Only one thread may execute this block at a time



How to Sync

- Atomic
  - The most basic unit of operation
  - Read-modify-write as one action

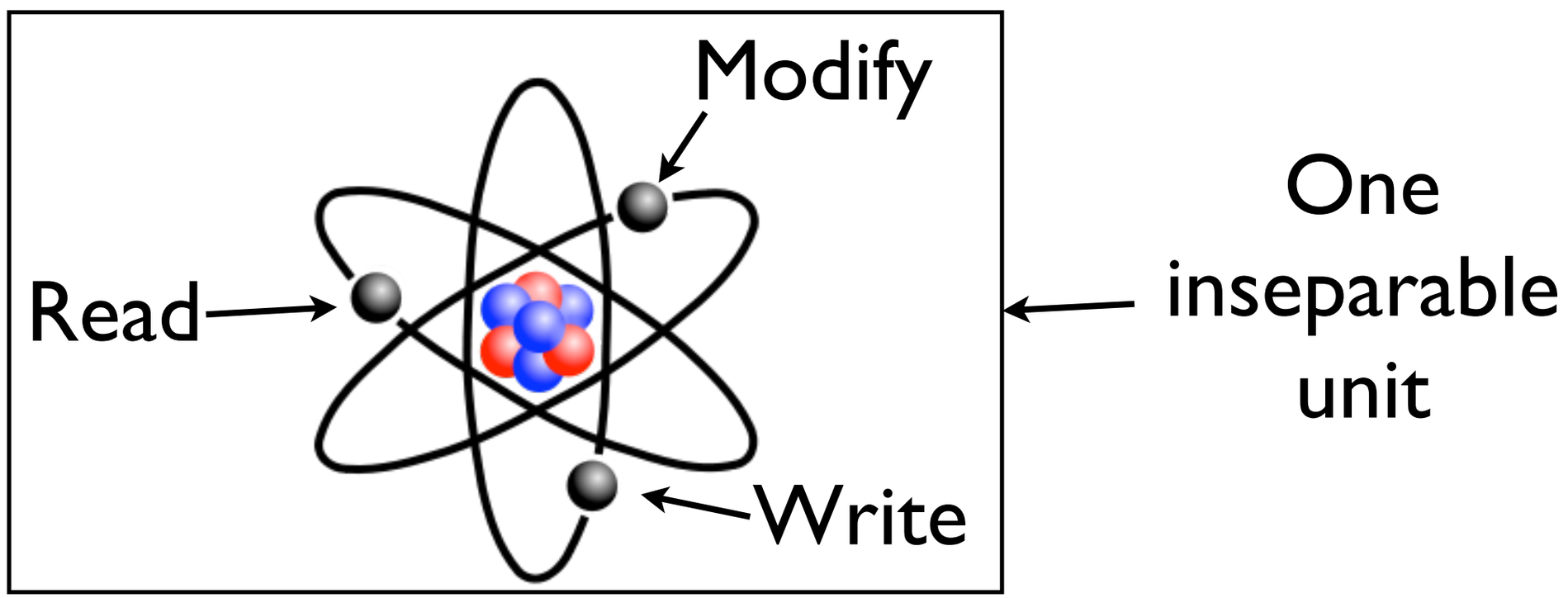


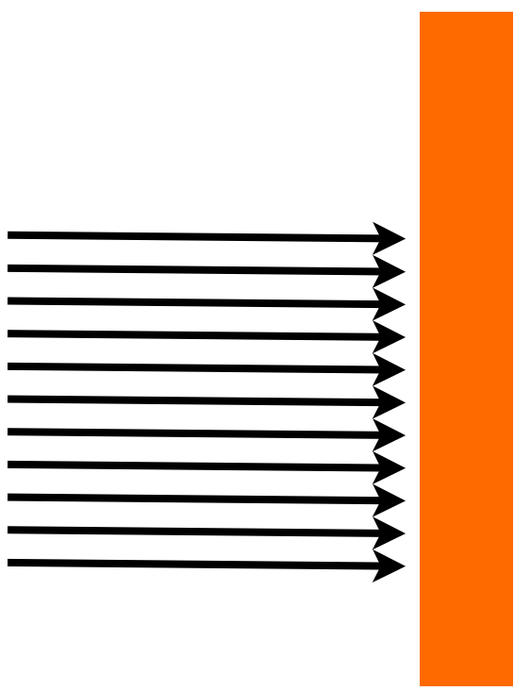
Image from wiktionary

How to Sync

- Barrier
  - You've already seen this in MR
  - Everybody waits for the last thread

How to Sync

- Barrier
  - You've already seen this in MR
  - Everybody waits for the last thread



How to Sync

- Barrier
  - You've already seen this in MR
  - Everybody waits for the last thread

How to Sync

- There are other tools for synchronization
  - There are also entire classes on parallel programming (e.g.15-846)
- More info:

  http://en.wikipedia.org/wiki/OpenMP

  (OpenMP is available on PSC's Blacklight)

Parallelism slow down

- Synchronization has a cost
  - sometimes greater than the speedup
- The more you talk, the more it costs

Parallel L1

- Based on:

  Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin.
  "Parallel Coordinate Descent for L1-Regularized Loss Minimization."
  In the 28th International Conference on Machine Learning (ICML), 2011.

Parallel L1

- Reminder:

$$F(\beta) = \min_{\beta} \frac{1}{2} \| \beta \mathbf{x} - \mathbf{y} \|_2^2 + \lambda \| \beta \|_1$$

- Produces sparse solutions

Parallel L1

- Reminder:

$$F(\beta) = \min_{\beta} \frac{1}{2} \| \beta \mathbf{x} - \mathbf{y} \|_2^2 + \lambda \| \beta \|_1$$

Regularized

- Produces sparse solutions

Sparse solutions

- Useful in big data situations
  - especially if you think few features are actually relevant

Solving L1

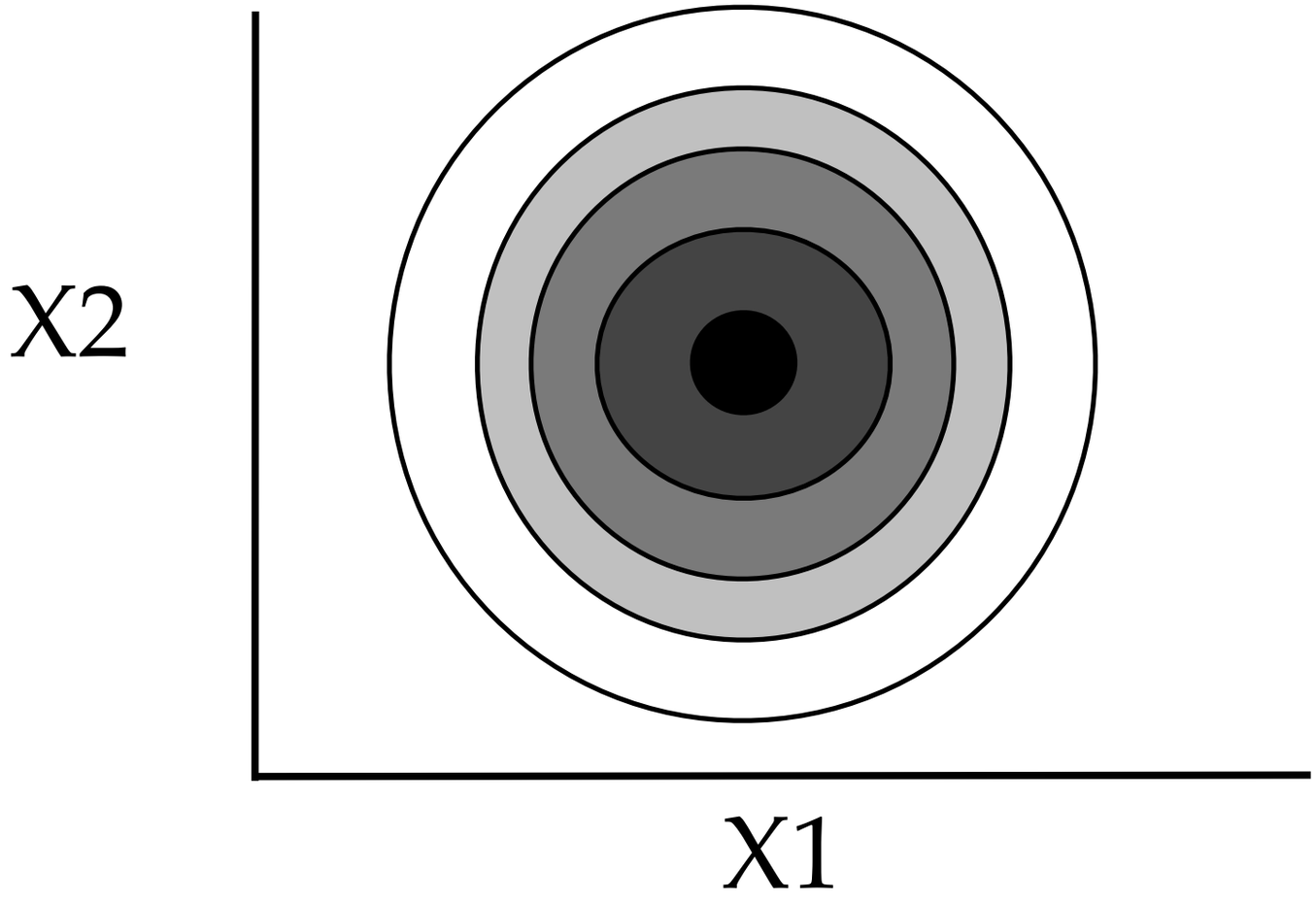
- No closed form solution
- Instead, do some kind of gradient descent to minimize $F(\beta)$

Solving L1

- Shooting (Wu, 1998)
  - Cycle through the dimensions, optimizing at each step

Solving L1

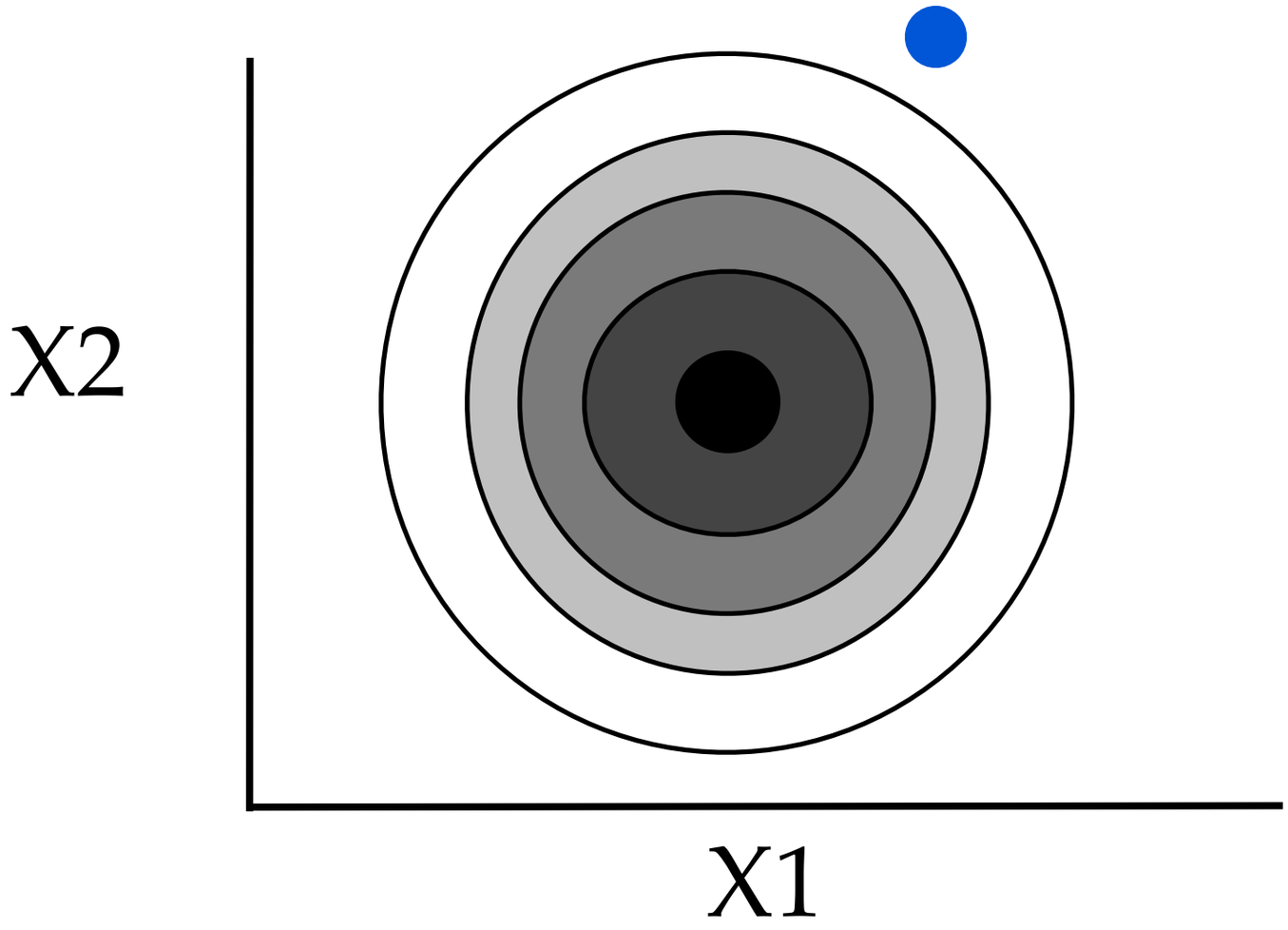
- Darker is lower (i.e. more optimal)



- Co-ordinate descent

Solving L1

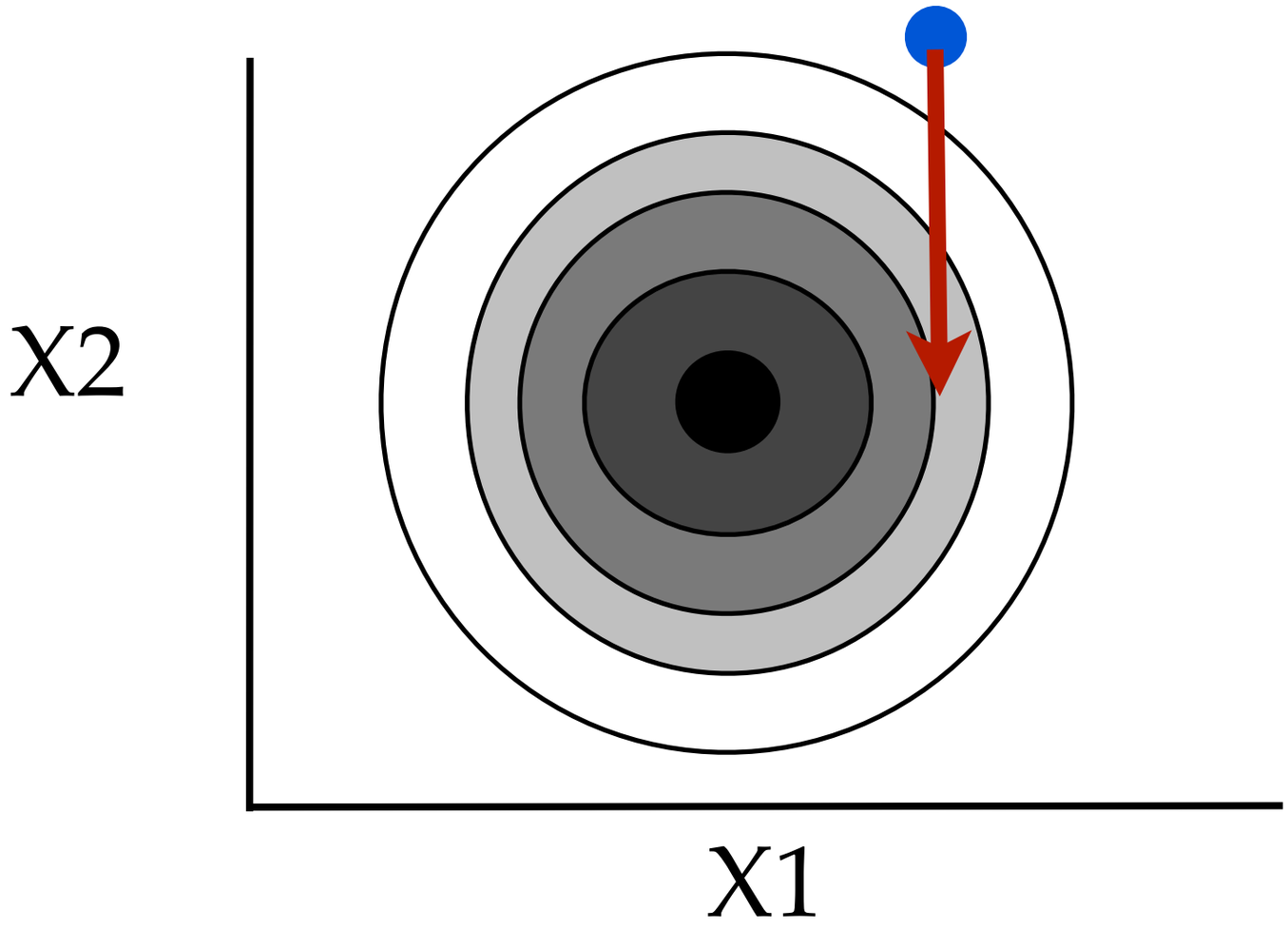
- Darker is lower (i.e. more optimal)



- Co-ordinate descent

Solving L1

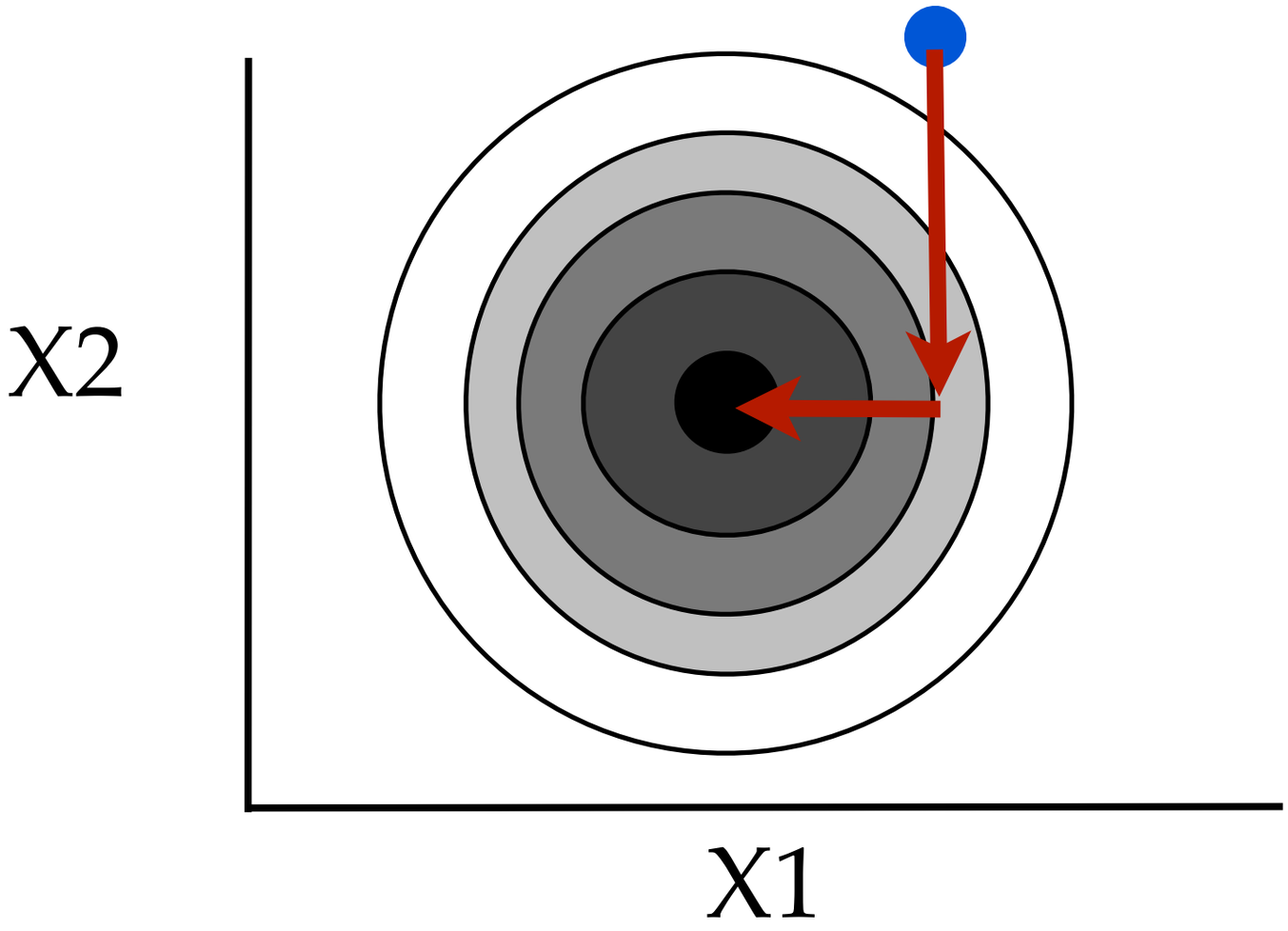
- Darker is lower (i.e. more optimal)



- Co-ordinate descent

Solving L1

- Darker is lower (i.e. more optimal)



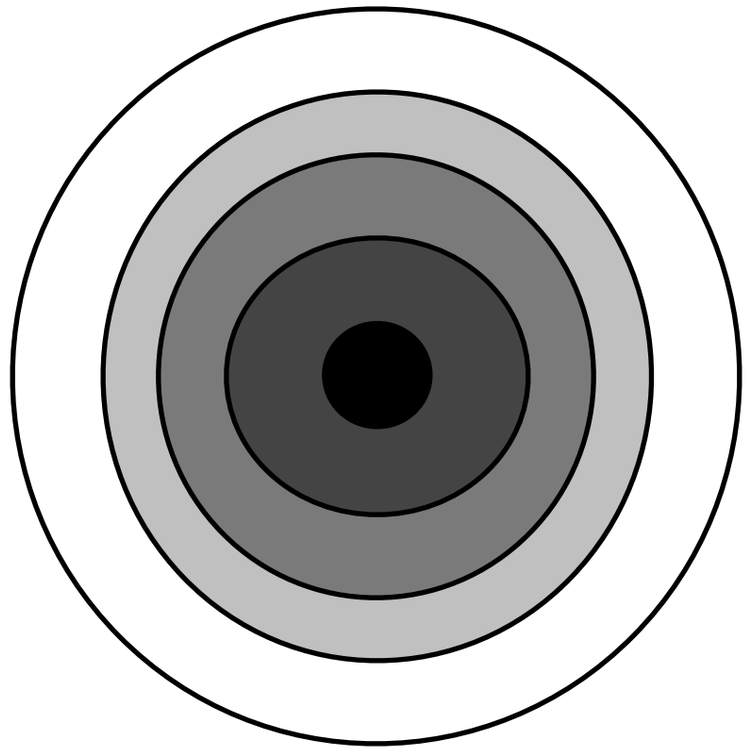
- Co-ordinate descent

Solving L1

- Stochastic co-ordinate descent (SCD)
  - Choose a dimension at random, and minimize

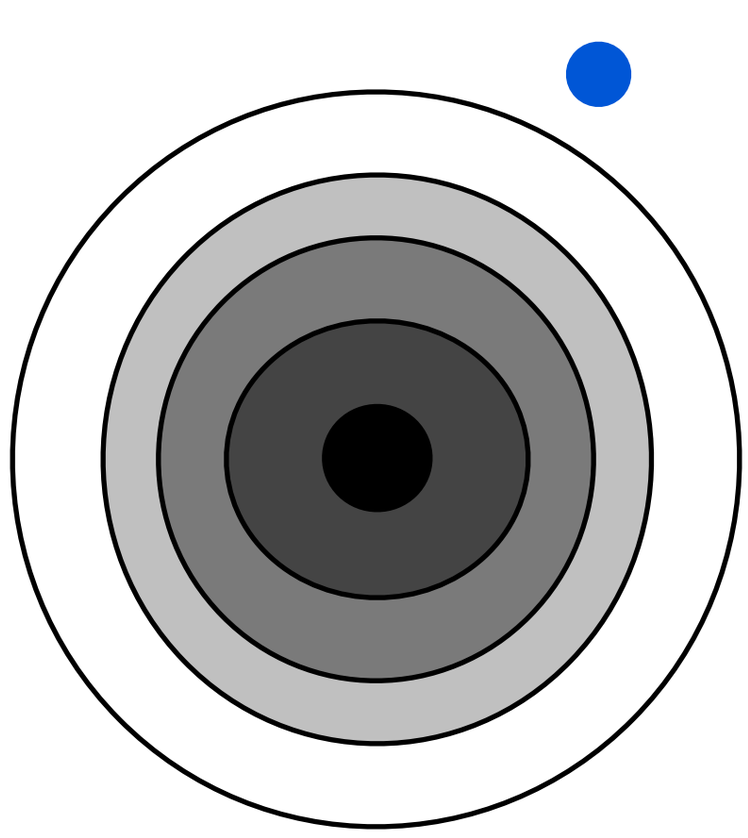
Solving L1

- How can we speed this up?
- What if we took both steps at the same time?



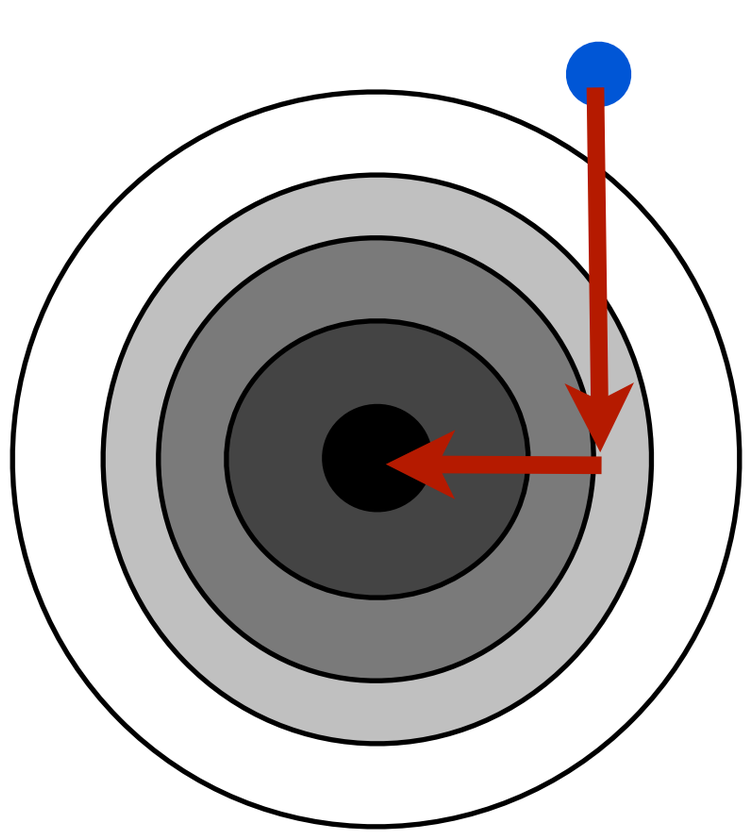
Solving L1

- How can we speed this up?
- What if we took both steps at the same time?



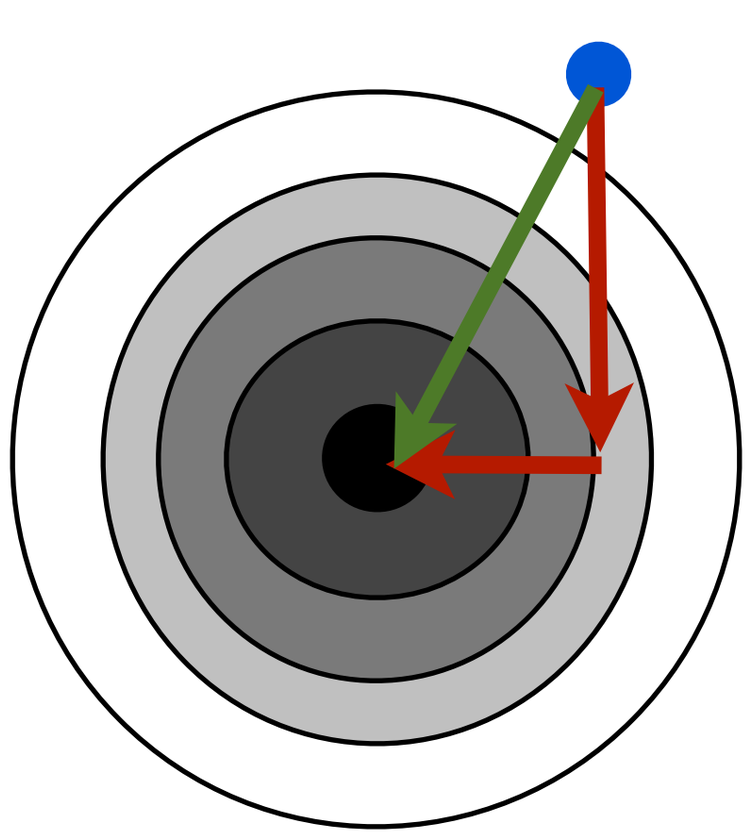
Solving L1

- How can we speed this up?
- What if we took both steps at the same time?



Solving L1

- How can we speed this up?
- What if we took both steps at the same time?



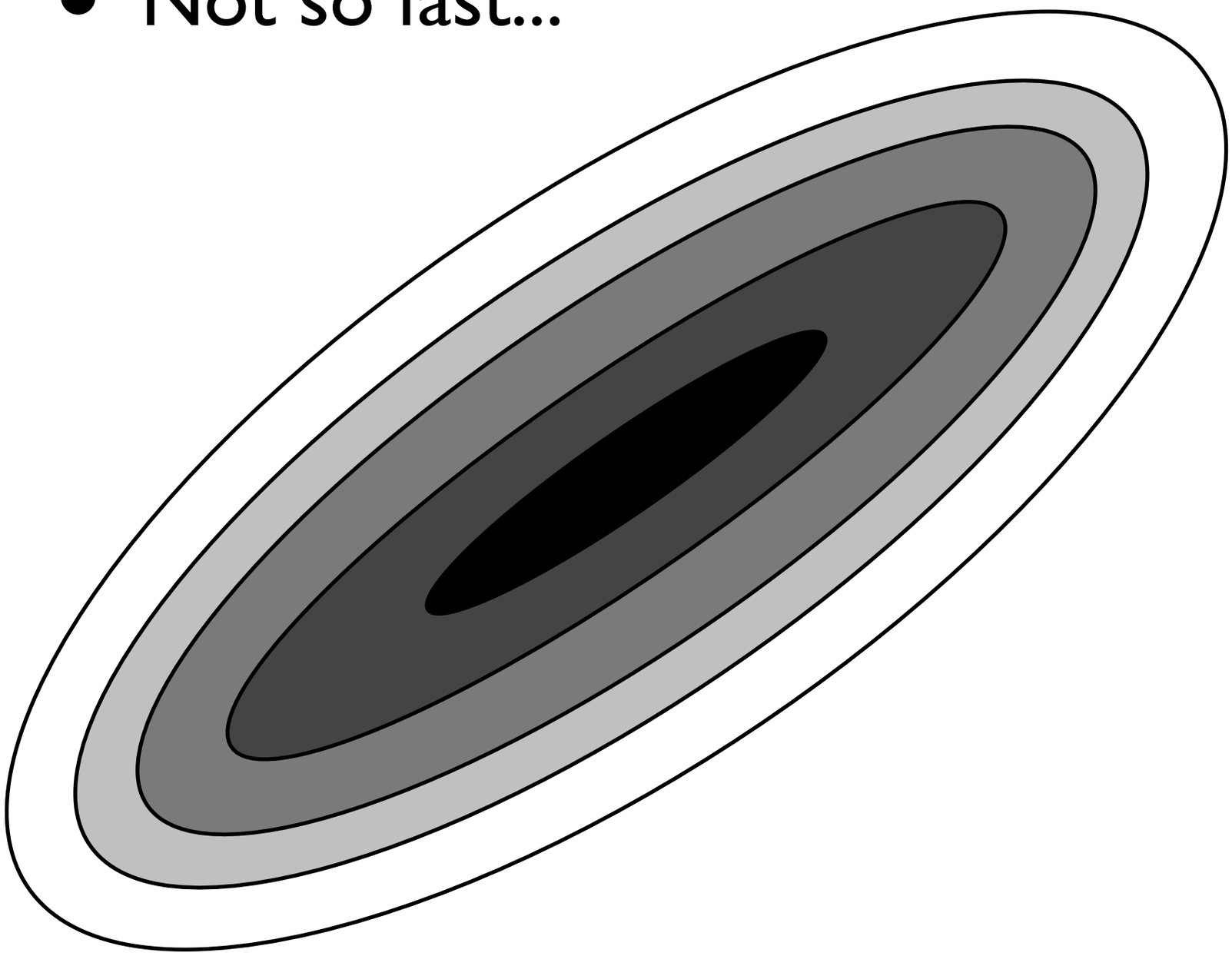
Equivalent to this!

AWESOME!

- Not so fast...

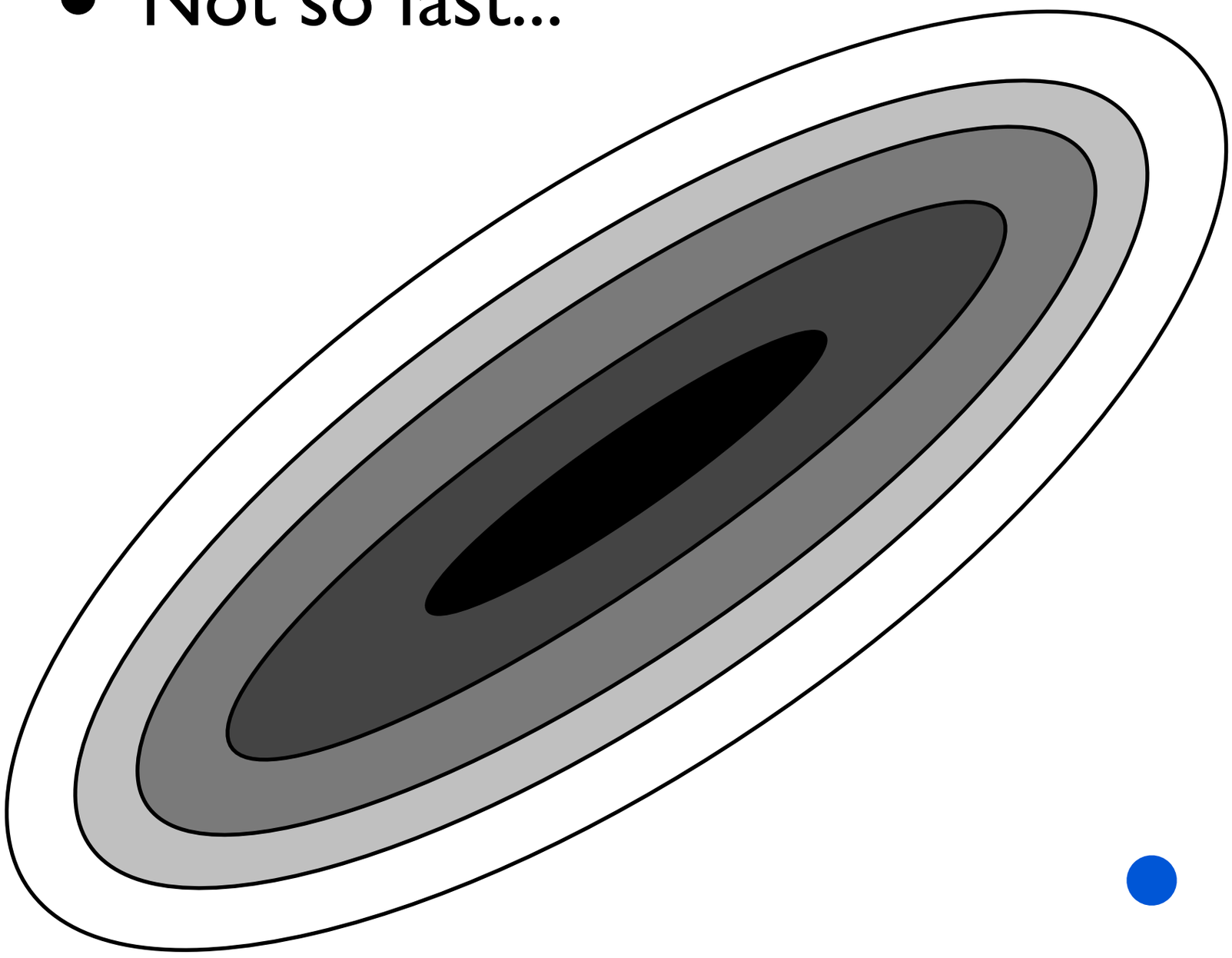
AWESOME!

- Not so fast...



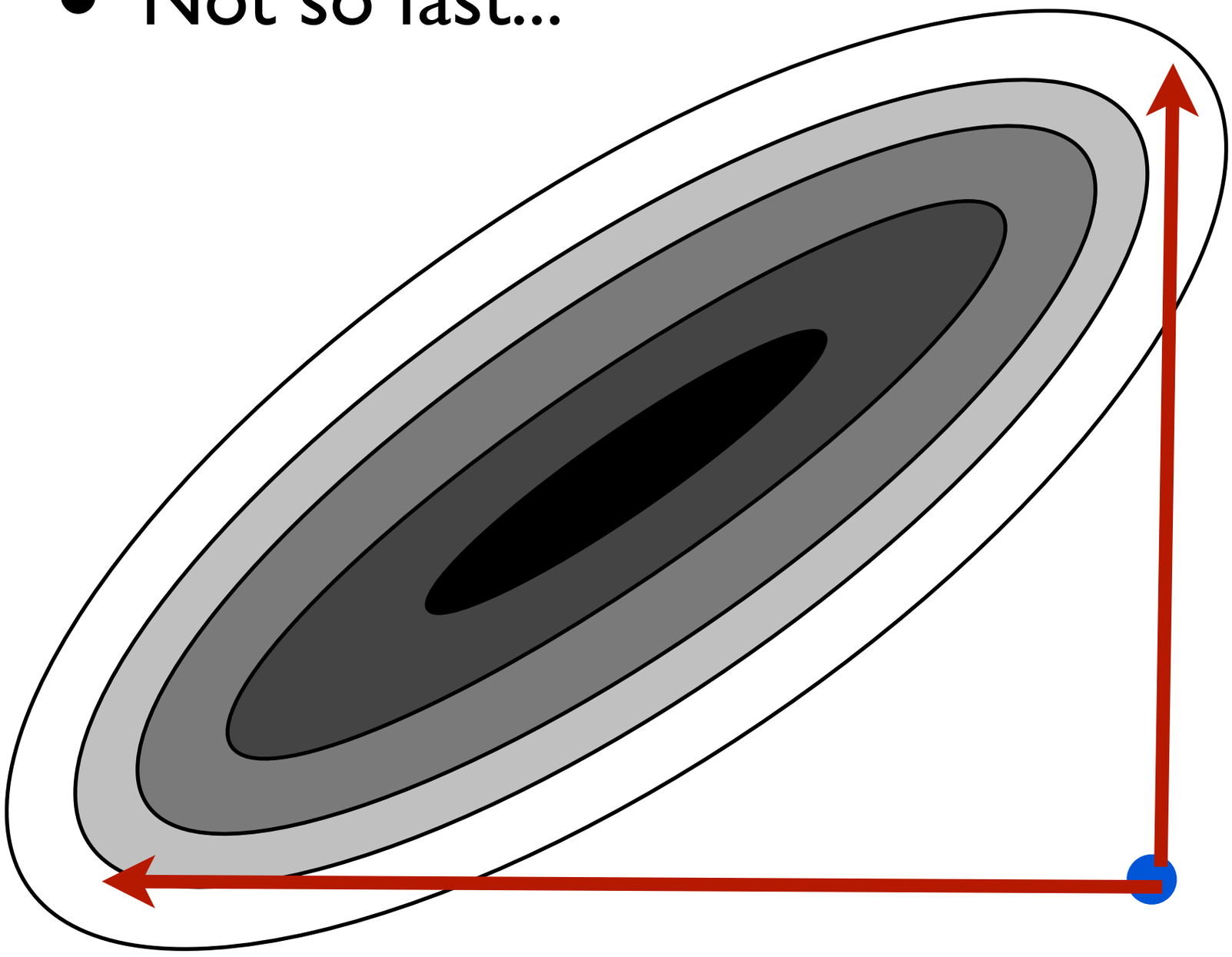
AWESOME!

- Not so fast...



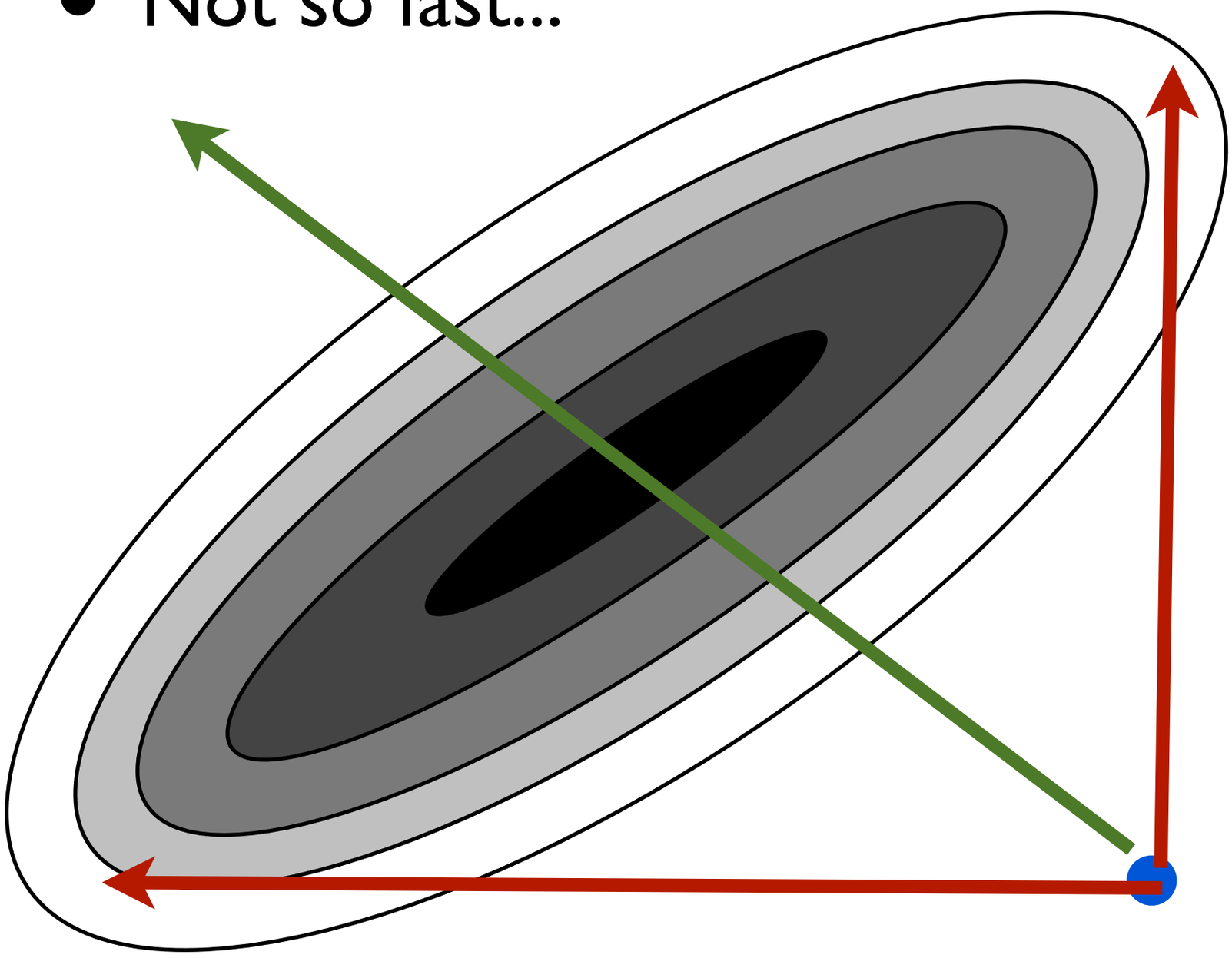
AWESOME!

- Not so fast...



AWESOME!

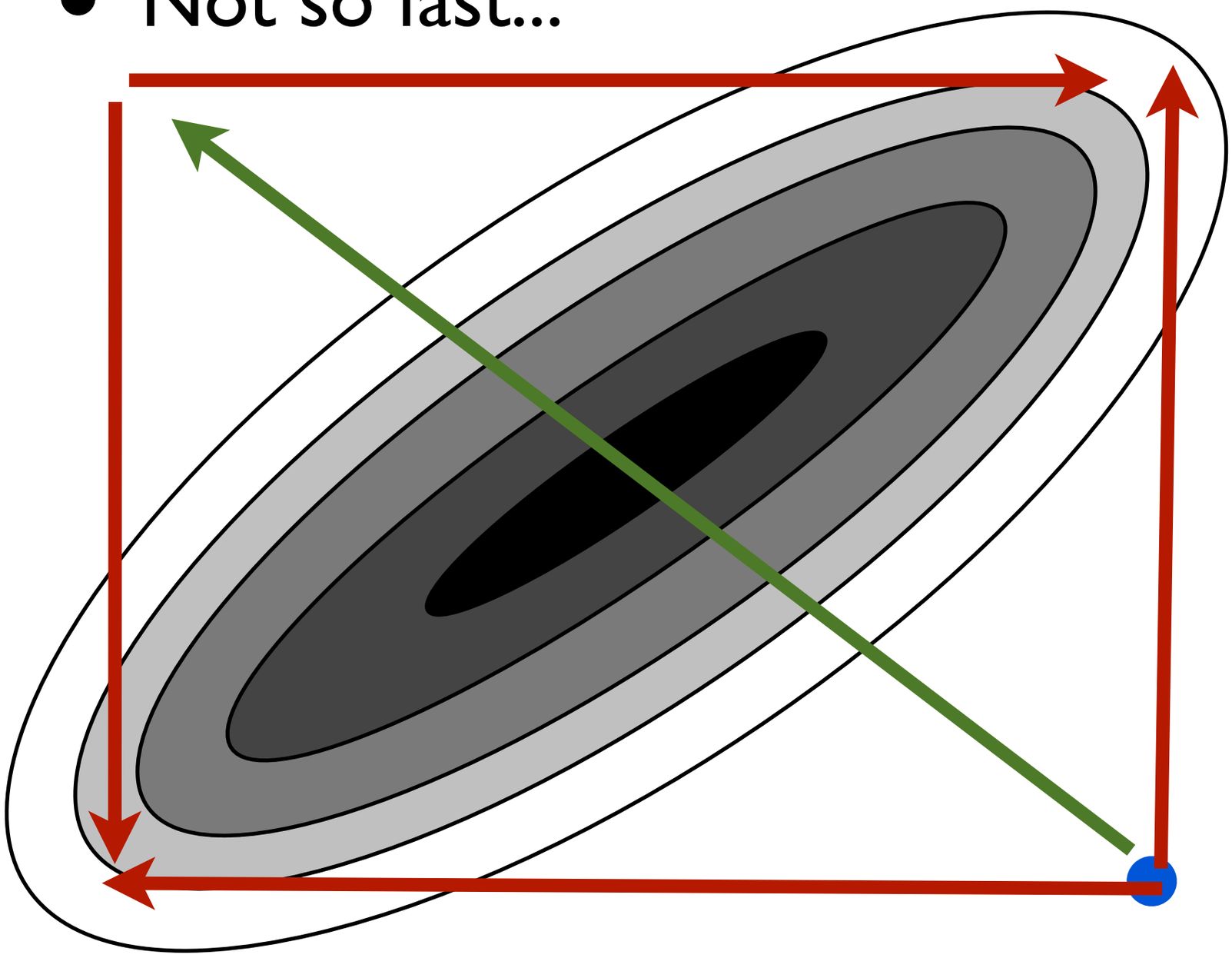
- Not so fast...



Equivalent to this
:(

AWESOME!

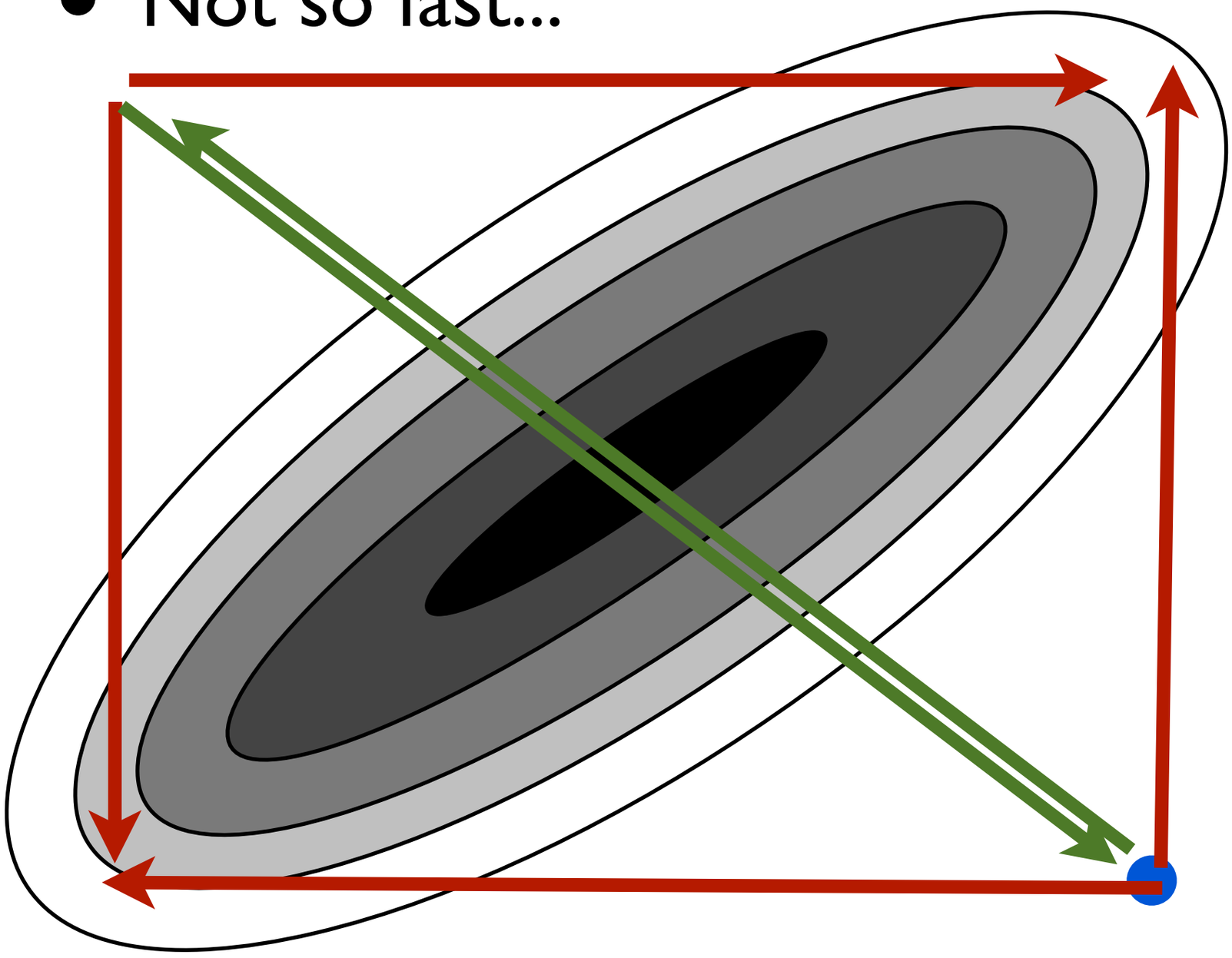
- Not so fast...



Equivalent to this :(

AWESOME!

- Not so fast...



Equivalent to this
:(

What are the chances?

- If variables are correlated, parallel updates will interfere

Loss update

$$F(\beta + \Delta\beta) - F(\beta)$$

$$\leq -\tfrac{1}{2} \sum_{i_j \in \mathcal{P}} \left(\delta\beta_{i_j} \right)^2 + \tfrac{1}{2} \sum_{\substack{i_j, i_k \in \mathcal{P}, \\ j \neq k}} \left(X^{\mathbf{T}} X \right)_{i_j, i_k} \delta\beta_{i_j} \delta\beta_{i_k}$$

Loss update

Change in Loss

$$F(\beta+\Delta\beta)-F(\beta)$$

$$\leq -\tfrac{1}{2}\sum_{i_j\in\mathcal{P}}\left(\delta\beta_{i_j}\right)^2+\tfrac{1}{2}\sum_{\substack{i_j,i_k\in\mathcal{P},\\ j\neq k}}\left(X^{\mathbf{T}}X\right)_{i_j,i_k}\delta\beta_{i_j}\delta\beta_{i_k}$$

Loss update

Change in Loss

$$F(\beta+\Delta\beta)-F(\beta)$$

$$\leq -\frac{1}{2}\sum_{i_j\in\mathcal{P}}\left(\delta\beta_{i_j}\right)^2+\frac{1}{2}\sum_{\substack{i_j,i_k\in\mathcal{P},\\ j\neq k}}\left(X^{\mathbf{T}}X\right)_{i_j,i_k}\delta\beta_{i_j}\delta\beta_{i_k}$$

Sequential progress

Loss update

Change in Loss

$$F(\beta+\Delta\beta)-F(\beta)$$

$$\leq -\tfrac{1}{2}\sum_{i_j\in\mathcal{P}}\left(\delta\beta_{i_j}\right)^2+\tfrac{1}{2}\sum_{\substack{i_j,i_k\in\mathcal{P},\\ j\neq k}}\left(X^{\mathbf{T}}X\right)_{i_j,i_k}\delta\beta_{i_j}\delta\beta_{i_k}$$

Sequential progress

Interference due to correlation

Loss update

- Parallel updates will only help if variables are sufficiently uncorrelated

$$F(\beta + \Delta\beta) - F(\beta)$$

$$\leq -\tfrac{1}{2}\sum_{i_j \in \mathcal{P}} \left(\delta\beta_{i_j}\right)^2 + \tfrac{1}{2} \sum_{\substack{i_j, i_k \in \mathcal{P}, \\ j \neq k}} \left(X^{\mathbf{T}} X\right)_{i_j, i_k} \delta\beta_{i_j} \delta\beta_{i_k}$$

Loss update

- Parallel updates will only help if variables are sufficiently uncorrelated

$$F(\beta + \Delta\beta) - F(\beta)$$

$$\leq -\frac{1}{2}\sum_{i_j \in \mathcal{P}} \left(\delta\beta_{i_j}\right)^2 + \frac{1}{2} \sum_{\substack{i_j, i_k \in \mathcal{P}, \\ j \neq k}} \left(X^{\mathbf{T}}X\right)_{i_j, i_k} \delta\beta_{i_j}\delta\beta_{i_k}$$

zero if x_j,x_k uncorrelatied,
non-zero otherwise

Bounding interference

- Interference is related to correlation
- Choose the amount of parallelism based on amount of correlation

Spectral radius

- definition:

$$\rho(X^TX) = \max_i(|\lambda_i|)$$

where λ are eigenvalues of X^TX

PCA reminder

- For the covariance or correlation matrix, the eigenvectors correspond to principal components and the eigenvalues to the variance explained by the principal components. Principal component analysis of the correlation matrix provides an orthonormal eigen-basis for the space of the observed data: In this basis, the largest eigenvalues correspond to the principal-components that are associated with most of the covariability among a number of observed data.

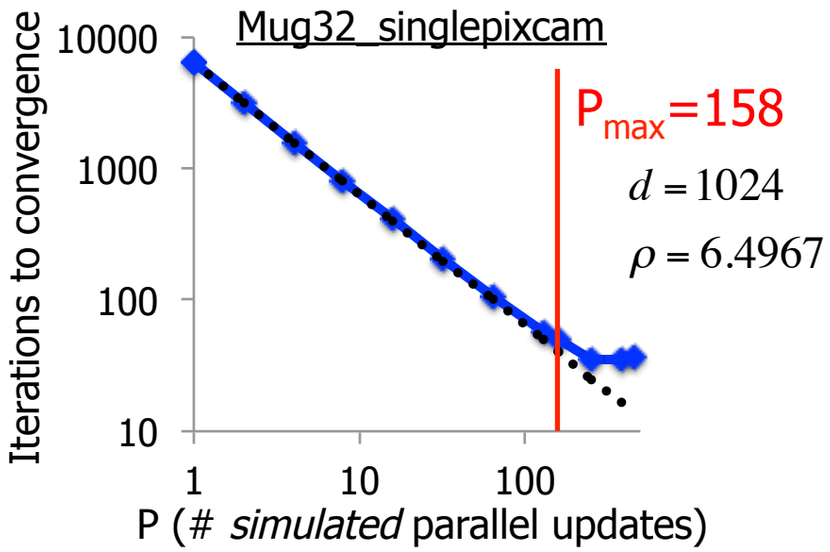
Wikipedia: Eigenvalue,_eigenvector_and_eigenspace

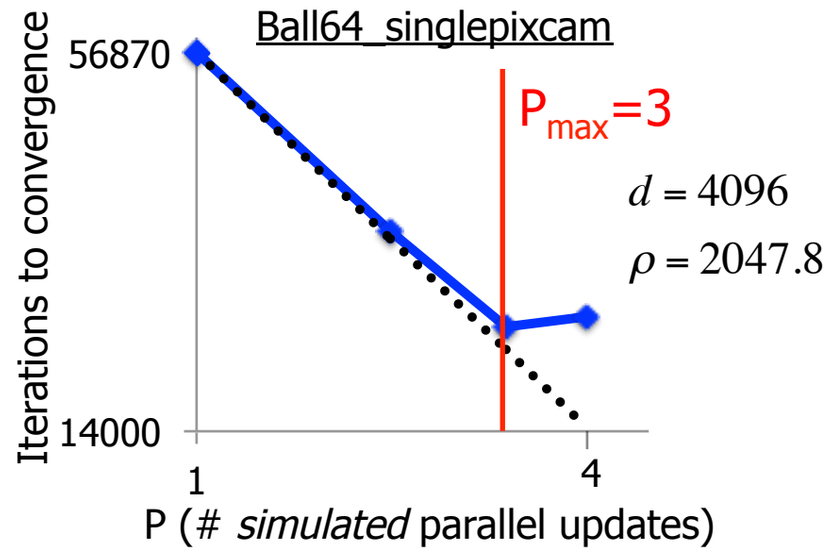
Parallelism and Eigenvalues

- We can do as many as $P < d/\rho + 1$ parallel updates (d = number of variables)
- if $\rho = 1$ (uncorrelated) we can perform all updates in parallel
- if $\rho = d$ (degenerate data) then we revert to shooting method ($P = 1$)

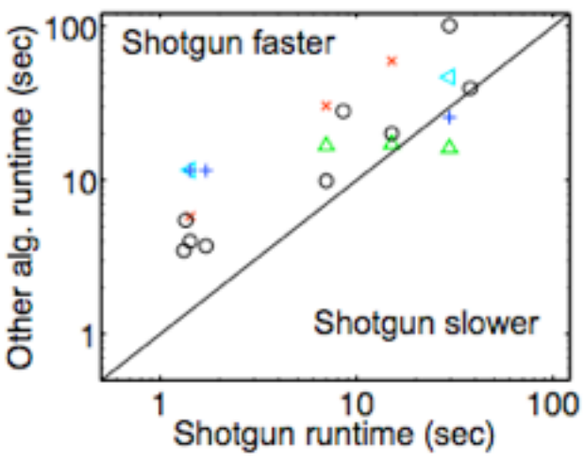
Experiments match theory

- Speedup is near-linear until the P_{max} threshold



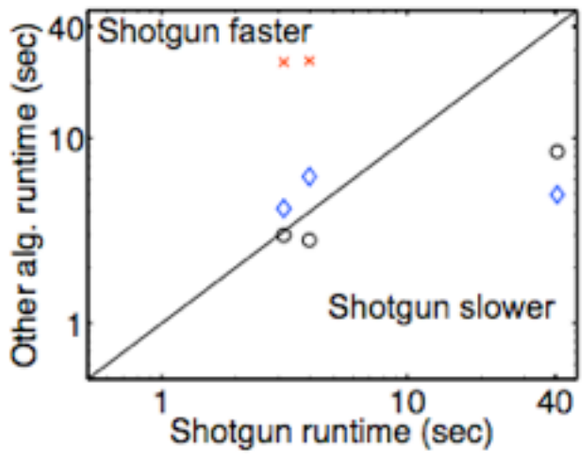


Comparing run time



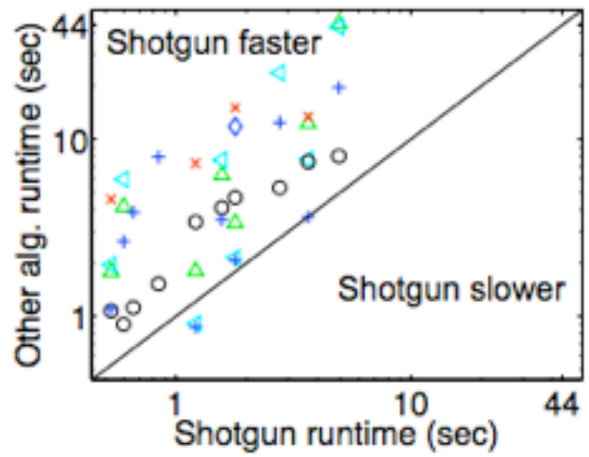
(a) Sparco

$\mathbf{P}^* \in [1, 8683]$, avg 1493



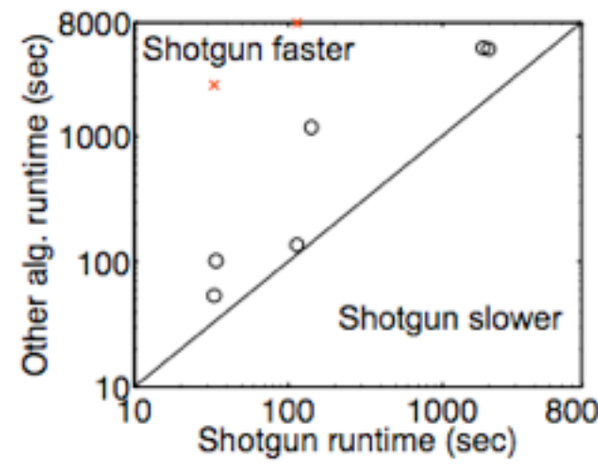
(b) Single-Pixel Camera

$\mathbf{P}^* = 1$



(c) Sparse Compressed Img.

$\mathbf{P}^* \in [1432, 5889]$, avg 3844



(d) Large, Sparse Datasets

$\mathbf{P}^* \in [107, 1036]$, avg 571

Shooting L1_LS FPC_AS GPSR_BB Hard_l0 SpaRSA

Conclusions

- Not everything is a mapreduce
- Libraries exist to make good use of multicore environments
- Correlation hinders parallel updates in co-ordinate descent
  - Calculate the max number of parallel updates via the spectral radius

Done!

- Questions?

- I'll be having special office hours
  - Friday Feb 16, 10-11, outside GHC 8009
  - Tuesday Feb 22, 12-1 via Skype/Google hangout
  - Monday Feb 27, 10-11 outside GHC 8009