18-646 - How to Write Fast Code II

Carnegie Mellon University

How to Write Fast Code?

Fast Platforms + **Good Techniques**

- **Multicore platforms**
- **Manycore platforms**
- **Cloud platforms**

- **Data structures**
- **Algorithms**
- **Software Architecture**

- **Course Goals**
  - To write fast code for your research/application, you should:
    1. **F**eel comfortable hacking up a solution
    2. **L**everage existing software building blocks
    3. **I**ndicate which platform is the best one to use
    4. **R**eason about why a piece of existing code is slow
    5. **T**ake care of potential performance bottlenecks

Course Staff

- **Instructor: Ian Lane**
  - Preferred contact method: Piazza
  - Office Hours: Wed 4:30-5:30pm ET
  - Email: ianlane@andrew.cmu.edu

- **Teaching Assistants**
  - Deepak Arumugam Sankara Subramanianh
  - Yunfan Jiang (Silicon Valley)

Experience

SOWG Associate Professor in AI Systems
Carnegie Mellon University
2006 – Present · 15 yrs
Pittsburgh, Pennsylvania, United States

Founder and CEO (Acq. Twilio)
Capio Inc (Acq. by Twilio)
2014 – Present · 7 yrs
Mountain View, California, United States

Board Member
Ichibel (Acq. by Sense Of Wonder Group)
2007 – 2017 · 10 yrs
Tokyo, Japan

Principle Research Scientist
Mobile Technologies (Acq. by Facebook)
2006 – 2011 · 5 yrs

Education

Kyoto University
PhD, Informatics
2002 – 2005

Kyoto University
Masters, Artificial Intelligence
2001 – 2002

Massey University
Bachelor of Informatics
1996 – 2000

Course Information

- **Lectures:**
  - Tuesday and Thursday 6:00pm-7:20pm ET
- **Office Hours:**
  - Instructor Office Hours: Wednesdays 4:30pm-5:30pm ET
  - TA Office Hours: TBD
- **Grading:**
  - 10% - Homeworks
  - 30% - Mini-Projects
  - 30% - Term Project
  - 30% - Final Exam
- **Course Links:**
  - **Canvas:** https://canvas.cmu.edu/courses/21510/pages/course-schedule
  - **Piazza:** https://piazza.com/class/kkmp02yc92h598
  - **Gradescope:** https://www.gradescope.com/courses/241050

https://canvas.cmu.edu/courses/21510/pages/course-schedule

canvas.cmu.edu/courses/21510/pages/course-schedule

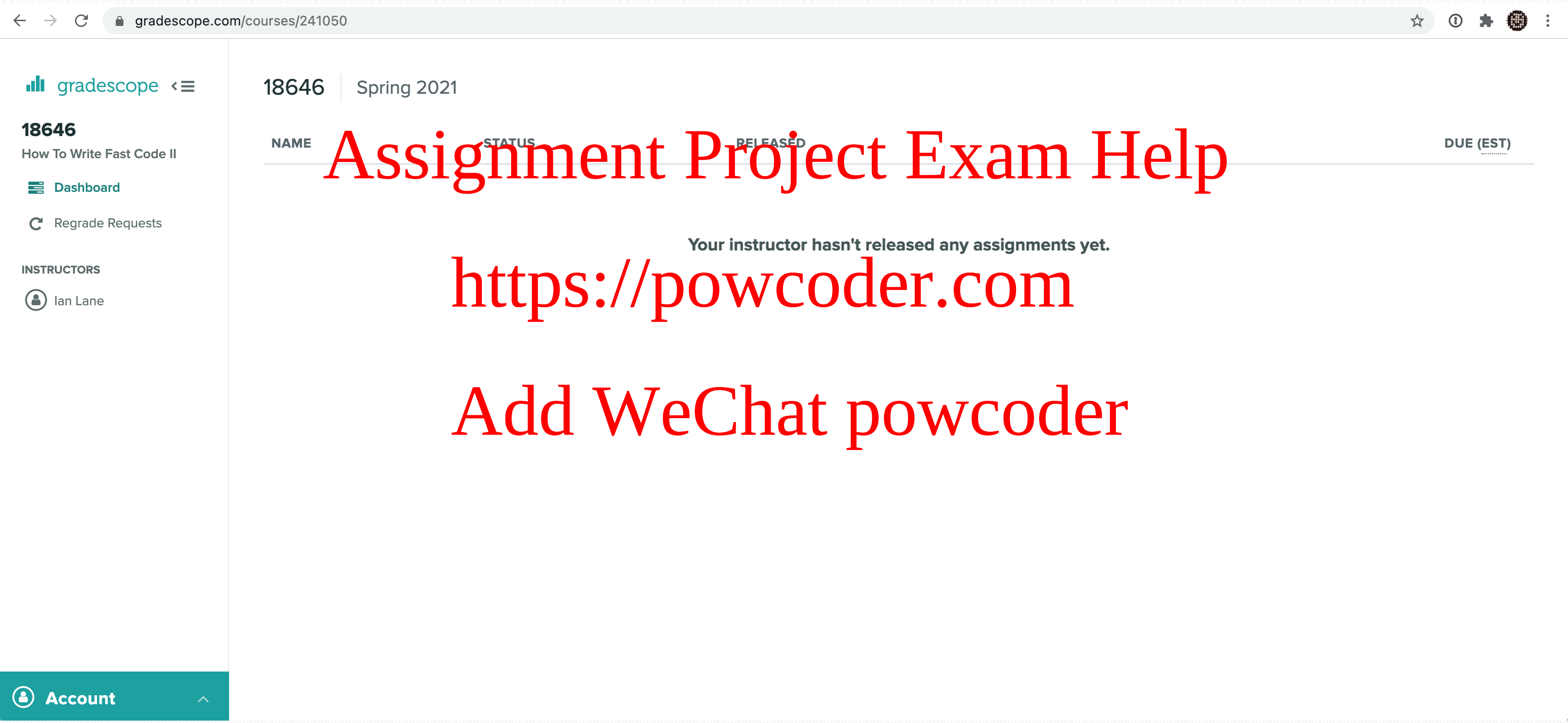
Carnegie Mellon University
Account
Dashboard
Courses
Calendar
Inbox
History
Commons
Help

Introduction and Multicore Programming		
Tue, Feb 2	Introduction to How to Write Fast Code II	
Thurs, Feb 4	Advanced Parallel Hardware Architectures	
Tue, Feb 9	Concurrency Opportunity Recognition	**Homework 1 Released**
Thur, Feb 11	Application Design for Multicore Programming	
Tue, Feb 16	Application Design for Manycore	**Mini-Project 1 Released**
Thur, Feb 18	Overview of Mini-Project 1	
Tue, Feb 23	Performance Analysis: Roofline Model	
Thur, Feb 25	No Class	**Homework 1 Due**
Manycore Programming		
Tue, Mar 2	Introduction to CUDA	**Homework 2 Released**
Thurs, Mar 4	Application Design for Manycore Programming	
Mon, Mar 8		**Mini-Project 1 Due**
Tue, Mar 9	Manycore Advanced Optimizations	**Project Proposals Due**
Thur, Mar 11	Mini-Project 1 Review	
Tue, Mar 16	No Class	**Mini-Project 2 Released**
Thur, Mar 18	Overview of Mini-Project 2	**Homework 2 Due**

https://piazza.com/class/kkmp02yc92h598

Sign Up Link: https://piazza.com/cmu/spring2021/18646

https://www.gradescope.com/courses/241050



Outline

- **Why Fast Code Now?**
- What does 100x speedup mean?
- Course structure and organization
- Problem solving when writing fast code

Why Fast Code?

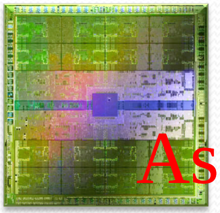
Need is driven by the applications…

…NOT by the availability of the platforms.

RMS Appears Everywhere

- **Recognition**: Interpretations of the world (with models)
- **Mining**: Understanding of the world (discover hidden patterns with model)
- **Synthesis**: Anticipate outcomes in the world (using the models to predict)

What is being valued today?



Hardware → **Commoditized**


```
int main(void) {
    mykernel<<<1,1>>>();
    printf("Hello World!\n");
    return 0;
}
```

Software → **Open Sourced**

Data → **King!**

Outline

- Why Fast Code Now?
- **What does 100x speedup mean?**
- Course structure and organization
- Problem solving when writing fast code

What does 100x Speedup Mean?

12 hours:	10x speedup →	1.2	hours
	100x speedup →	**7.2**	**minutes**
	1000x speedup →	43.2	seconds

- Game changing technology advances
  - Overnight jobs becomes interactive

Speech Analytics

Medical Imaging

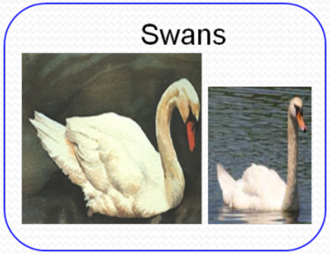
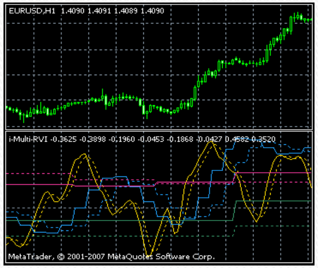


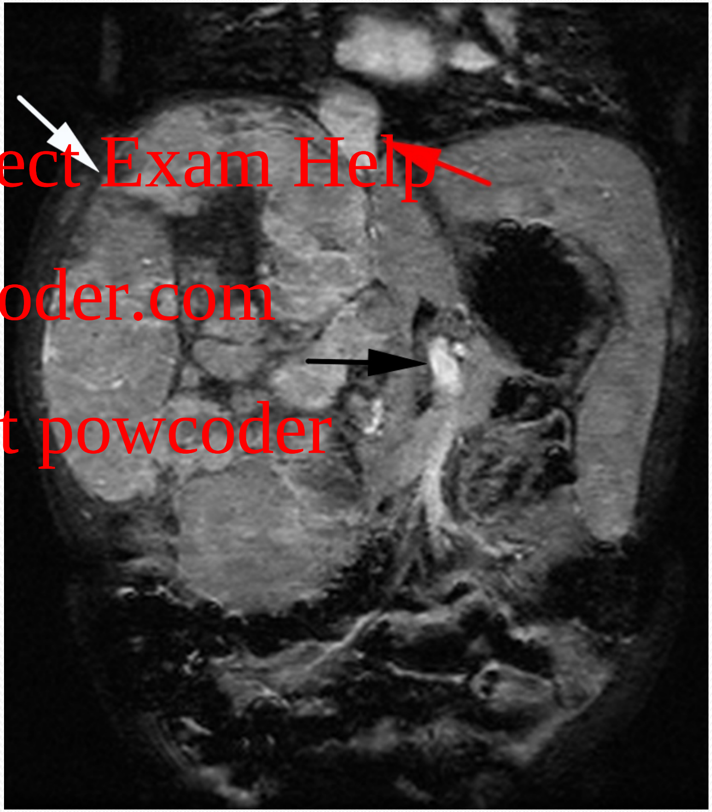
Image Recognition



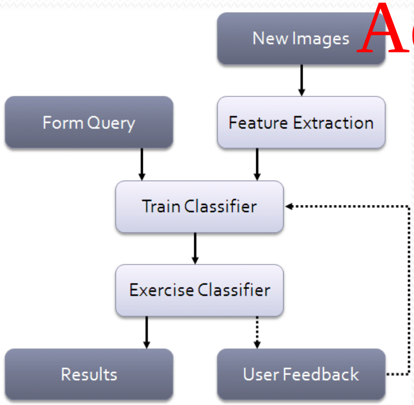
Computational Finance

Fast, Robust Pediatric MRI

- **100X** faster reconstruction
- Higher-quality, faster MRI
- This image: 8 month-old patient with cancerous mass in liver
  - 256 x 84 x 154 x 8 data size
  - Serial Recon: 1 hour
  - Parallel Recon: less than 1 minute
- Fast enough for clinical use
  - Software currently deployed at Lucile Packard Children's Hospital for clinical study of the reconstruction technique



Support-Vector Machines



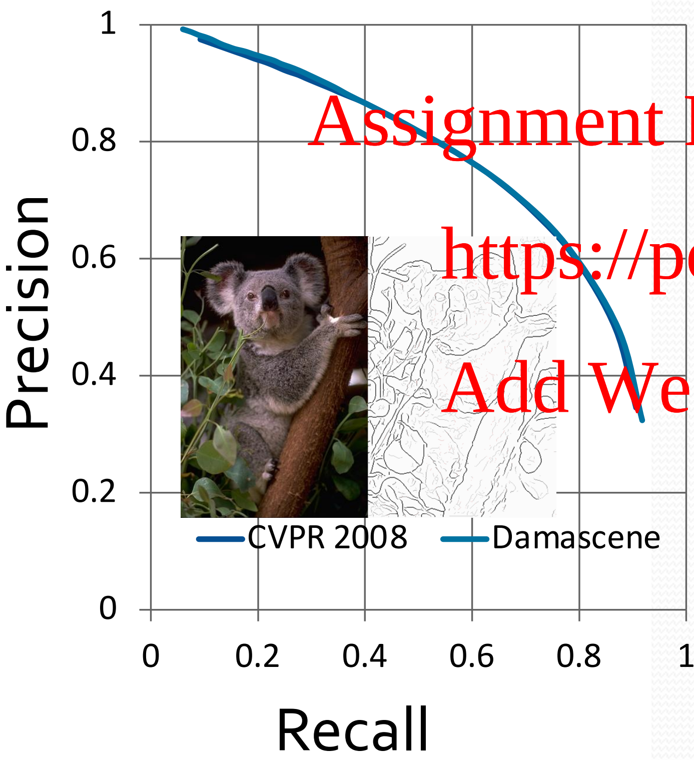
- Algorithmic changes and parallel implementation lead to performance speedup: Core 2 Duo versus G80

Computation	LIBSVM	Our algorithm2-core Parallel CPU	Our algorithm, 16-core Parallel GPU
SVM Training (geo-mean)	771.8 s	---	38.5 s
SVM Classification	41.42 s	4.21 s	0.38 s

100X speed-up

896 downloads since release in 10/2008

Image Contours Detection



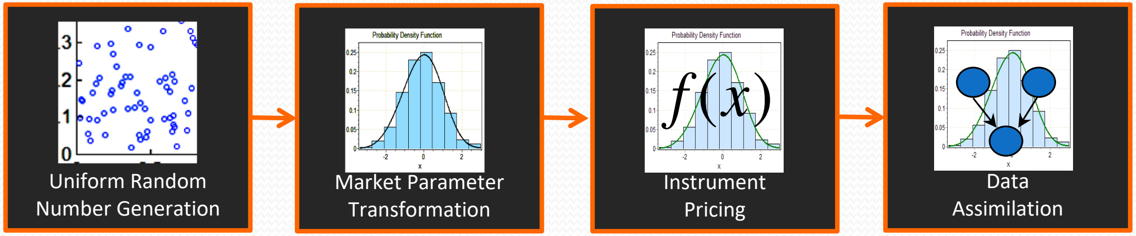
- We achieve equivalent accuracy on the Berkeley Segmentation Dataset
  - Comparing to human segmented "ground truth"
- F-measure 0.70 for both
- Human agreement = 0.79
- 3.8 minutes to 1.8 seconds:

  126x speedup
- 616 downloads since release in October 2009

Computational Finance

- **Value-at-Risk Computation** with **Monte Carlo Method**
- Summarizes a portfolio's vulnerabilities to market movements
- Important to algorithmic trading, derivative usage
- Improved implementation to run **60x faster** on a parallel microprocessor

Four Steps of Monte Carlo Method in Finance



NY Times TIFF to PDF

- In 2007, the New York Times decided to make all the public domain articles from 1851-1922 available free of charge
  - Needed to convert from TIFF to PDF

Data set:	4TB of raw image TIFF to 11 million PDF
Compute Instances:	100 Amazon EC2 instances
Time taken:	24 hours
Cost:	$240

The New York Times

http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/

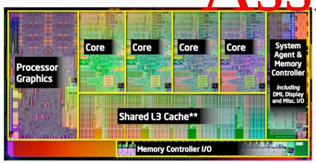
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Philosophy on Platforms

Provide theoretical background and hands-on practices…
…to innovate with multicore/manycore/cloud-based platforms.



Intel Sandy Bridge Multicore Processor (Core i7-2600K)

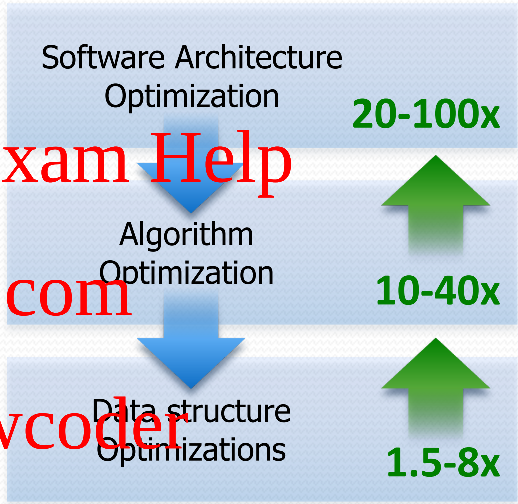
NVIDIA Fermi Manycore Processor - GTX580

Yahoo! Hadoop Cluster ~2000 nodes in one cluster

- ***Significant scientific advances*** will be empowered by multicore/manycore/cloud-based platforms over the next decade
- Knowledge of these new computing capabilities will give you an ***advantage*** over your peers in developing innovative techniques to solve challenging problems

Philosophy on Techniques

- Efficient ***software architecture*** is the most important to designing fast code
  - Software design patterns
- Understanding the implementation platform will help reason about application performance bottlenecks
- ***Hands-on experience*** provide confidence for you to effectively use these technologies for your research and development needs



Outline

- Why Fast Code Now?
- What does 100x speedup mean?
- Course structure and organization
- Problem solving when writing fast code

https://canvas.cmu.edu/courses/21510/quizzes/55032

Question 1 — 0 pts

Have you taken 18-645 "**How to Write Fast Code I**" in a prior semester?

- ○ Yes
- ○ No
- ○ I am taking that course this semester
- ○ Have never heard of that course

Question 2 — 0 pts

Which degree program are you in (i.e. MS-ECE, MS-INI)?

Question 3 — 0 pts

Where are you located this semester?

- ○ Pittsburgh
- ○ Silicon Valley
- ○ Somewhere else in the US
- ○ Overseas

Outline

- Why Fast Code Now?
- What does 100x speedup mean?
- **Course structure and organization**
- Problem solving when writing fast code

Organization of This Course

- **Lectures**
  - Tuesdays and Thursdays
  - All **mini-project introductions** held on Tuesdays, **mini-project reviews** on Thursdays
  - Keep an eye on the course schedule for changes

- **Canvas** - https://canvas.cmu.edu/courses/21510/pages/course-schedule
  - Course content (slides and videos)
  - Homework and Mini-Project Descriptions with links to Gradescope

- **Piazza** - https://piazza.com/class/kkmp02yc92h598
  - Course Announcements
  - Course Questions and Discussions

- **Gradescope** - https://www.gradescope.com/courses/241050
  - Assignment Submissions
  - Final Exam

Organization of This Course

- During class:
  - Lectures and Q&A sessions

- Outside class:
  - 3 homework assignments
  - 3 mini homework projects
  - 1 term project (focused on your own research or interests)
  - Lots of coding ;)

Grading and Expectations

- GRADING
  - 10% Homework assignments
  - 30% Homework projects
  - 30% Term Project
  - 30% Final examination
- EXPECTATIONS
  - Attend majority of lectures
  - Hand in all assignments and mini-projects
  - Complete a term project
  - Complete the final exam

Software Reuse and Plagiarism

- **Question**: Can I learn to use code from the web?
- **Answer**: Yes! But only under certain circumstances…

Plagiarism	Acceptable SW Reuse
Collusion Unacknowledged code from a third-party	File-level Reuse Third-party code factored out to a separate file
Reverse Engineering Unacknowledged re-use of some code abstraction	Acknowledgement in Documentation Third-party code clearly distinguished in documentation
Translation Unacknowledged re-use by translating functions from one language to another	Adequate Testing Third-party code must be tested to one's own requirements
Code Generation Unacknowledged assistance by using some code generators	
Reuse Without Testing Reuse third-party code without testing against one's own requirements	

Gibson, "Software Reuse and Plagiarism: A Code of Practice", In Proceedings of ITiCSE'2009. pp.55-59

Structure of Lectures

- **Module 1: Background and Multicore Programming**
  - Hardware architectures, applications
  - Application design with OpenMP
- **Module 23: Manycore Programming**
  - Manycore architectures
  - Application design with CUDA
- **Module 3: Cluster Programming**
  - Distributed architectures
  - Application design with Hadoop

Homework & Mini-Projects (40%)

- Three mini projects throughout the semester
    1. Multicore project with OpenMP
    2. Manycore project with CUDA
    3. Cloud project with Hadoop

- Projects are done in teams of two or three students

- Programming assignments will start with template code
- Accelerate them with techniques discussed in class

- Will submit code, a project write-up and selected teams will present during class

- Homeworks are done in preparation for the Mini-Projects
    - Internalize concepts discussed in class
    - Set up the project environment
    - Discussion among peers is allowed and encouraged. However, tasks and write-ups must be completed individually

Homework & Mini-Project Schedule

- **Homeworks (Project Setup)** **10%**
  - Module 1 – Multicore **Thursday, Feburary 25th**
  - Module 2 – Manycore **Thursday, March 18th**
  - Module 3 – Cluster Computing **Thursday, April 15th**

- **Mini-Projects (Coding and Evaluation)** **30%**
  - Module 1 – Multicore **Monday, March 8th**
  - Module 2 – Manycore **Monday, April 5th**
  - Module 3 – Cluster Computing **Monday, April 26th**

Term Project (30%)

- **<u>Term project</u>**
  - An application area of your choice
  - Term project introduction on Tuesday next week (1/9)
  - Projects are done in teams
  - Does not need to be the same team as for the Mini-Project

Term Project Schedule

- **Term (Team) Project** **30%**
  - Project Proposal (10%) **Tuesday, March 9^{th}**
  - Poster Presentations (40%) **Tuesday, May 4^{th}**
  - Final Report (50%) **Friday, May 14^{th}**

Exam (30%)

- **Final Exam – TBD (Week of May 10th-14th)**

Outline

- Why Fast Code Now?
- What does 100x speedup mean?
- Course structure and organization
- **Problem solving when writing fast code**

Problem Solving for Fast Code

- Writing fast code is a process coherent with

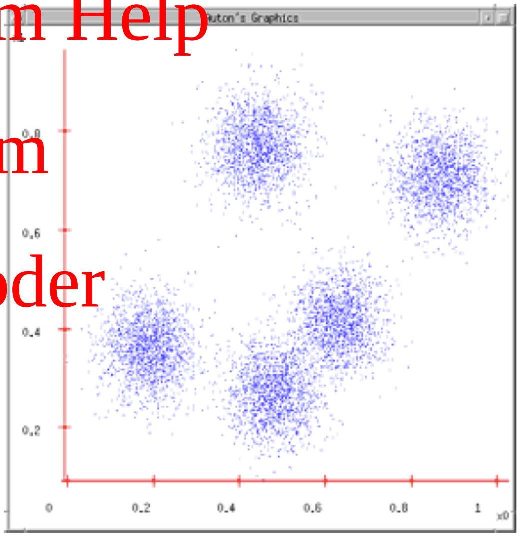
“general problem solving behavior”

Newell and Simon, Human Problem Solving (1972), pp. 72-73

- The process of problem solving involves:
  1. Understand the **current state**
  2. Observe the **internal representation**
  3. **Search** among alternatives
  4. Select from a set of **choices**

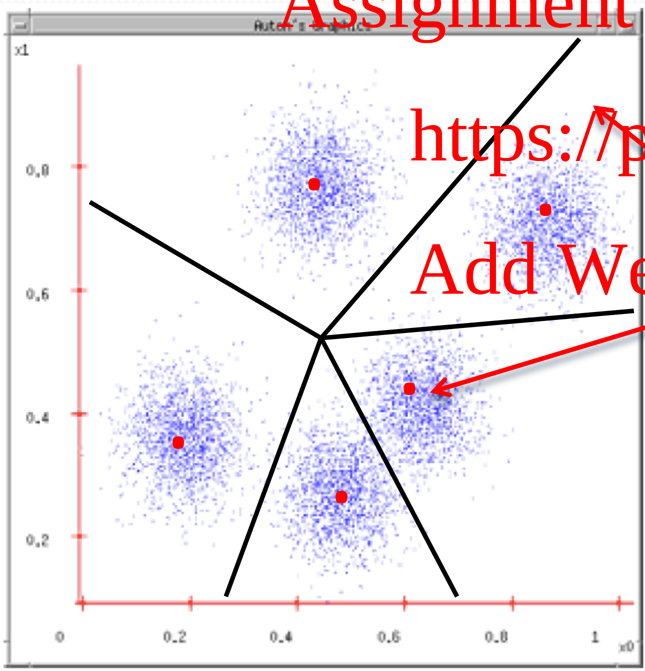
The *k*-means Problem

- Find *k* cluster centers that minimize the distance from each data point to a cluster center
- Important algorithm in machine learning:
  - Statistical data analysis
  - Vector quantization (Speech Recognition)
- NP-hard for arbitrary input
- ***k*-means algorithm** frequently finds a reasonable solutions quickly
- Issues:
  - Worst case running time is super-polynomial
  - Approximation can be arbitrarily bad



The *k*-means Problem

- Find *k* cluster centers that minimize the distance from each data point to a cluster center



Cluster

Cluster center (centroid)

k: Number of clusters (defined a-priori)
Cluster: Assignment of data points to a class
Cluster Center: μ of data points in a cluster

k-means Algorithm (*"Lloyd's algorithm"*)

- Given an initial set of k means $\mathbf{m}_1^{(1)},\ldots,\mathbf{m}_k^{(1)}$
- **Expectation Step:** Assign each observation to the cluster with the closest mean

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \big\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \big\| \leq \big\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \big\| \text{ for all } i^* = 1, \ldots, k \right\}$$

- **Maximization Step:** Calculate the new means to be the centroid of the observations in the cluster.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

- Iterate until convergence or stopping criteria met

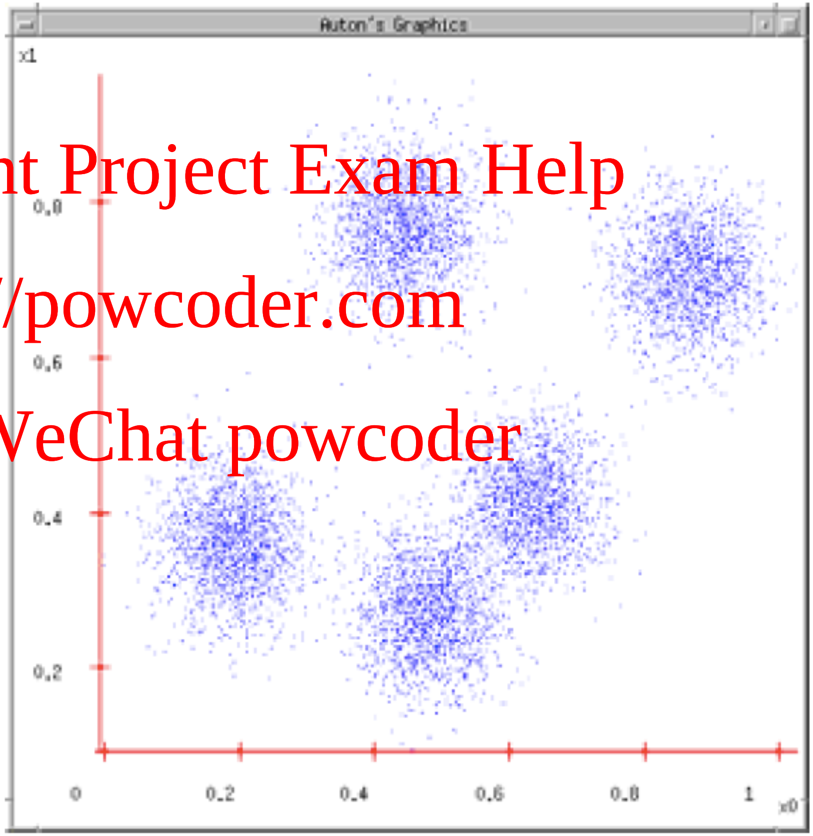
The Algorithm

Example:

k=5

Distance metric=euclidean

Dimensions=2



1. Randomly select k cluster Centers
2. Assign closest Center to each data point
3. Update Centers based on assignments from (2)
4. Re-iterate steps 2-3 until convergence or stopping criteria met

The Algorithm

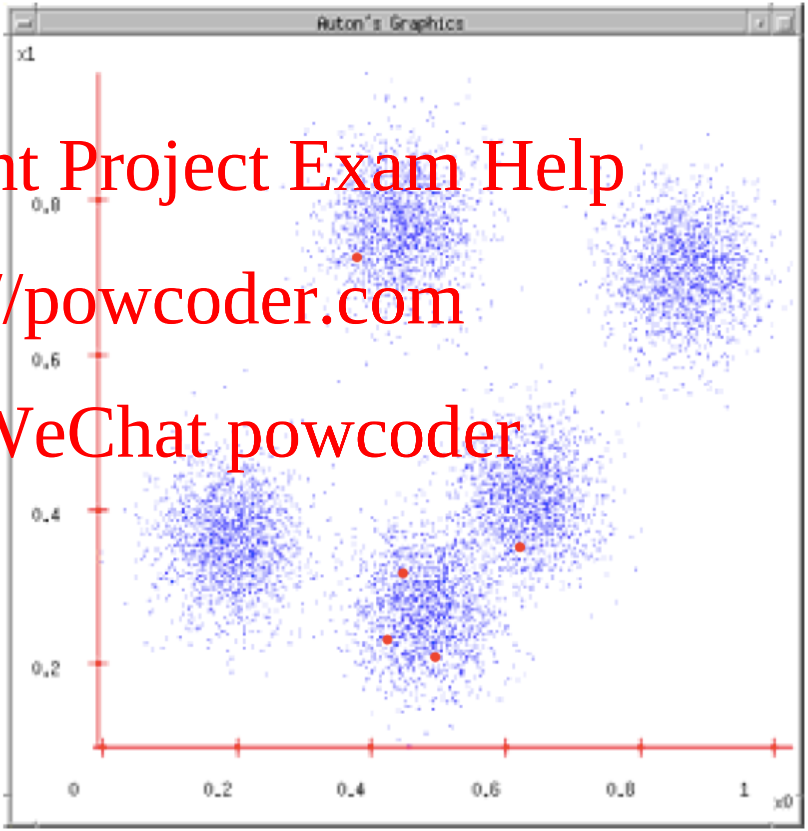
Example:

k=5

Distance metric=euclidean

Dimensions=2

1. Randomly select k cluster Centers
2. Assign closest Center to each data point
3. Update Centers based on assignments from (2)
4. Re-iterate steps 2-3 until convergence or stopping criteria met



The Algorithm

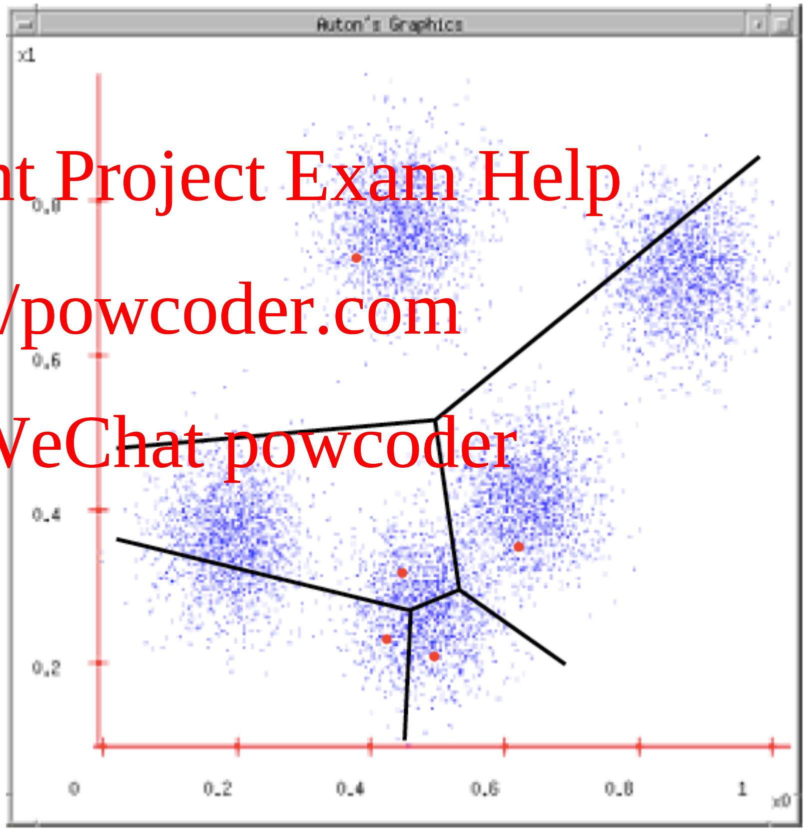
Example:

k=5

Distance metric=euclidean

Dimensions=2

1. Randomly select k cluster Centers
2. Assign each data point to closest Center
3. Update Centers based on assignments from (2)
4. Re-iterate steps 2-3 until convergence or stopping criteria met



The Algorithm

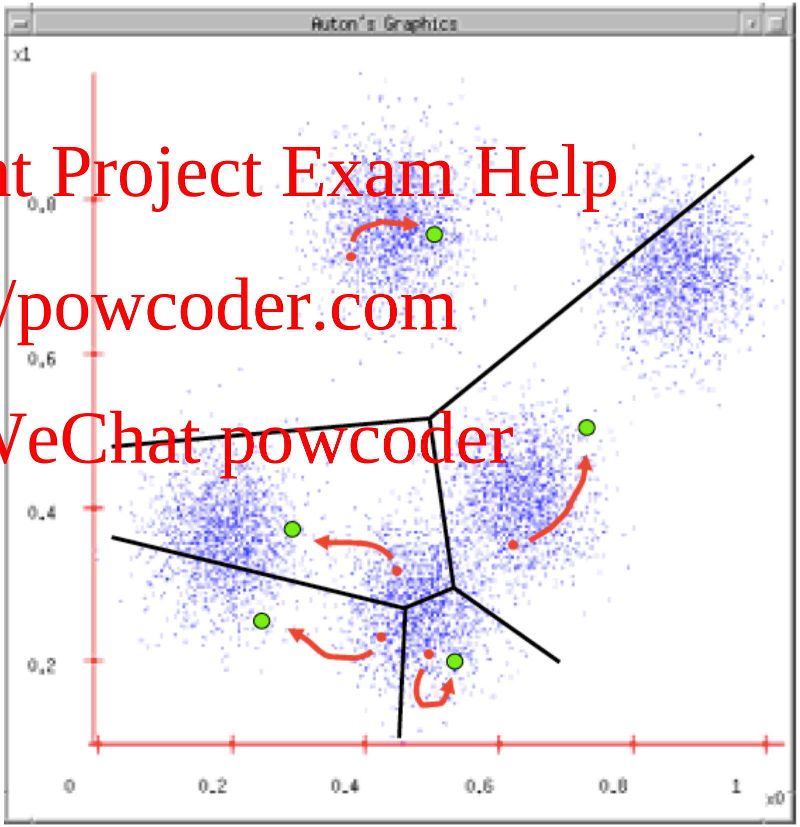
Example:

k=5

Distance metric=euclidean

Dimensions=2

1. Randomly select k cluster Centers
2. Assign closest Center to each data point
3. Update Centers based on assignments from (2)
4. Re-iterate steps 2-3 until convergence or stopping criteria met



The Algorithm

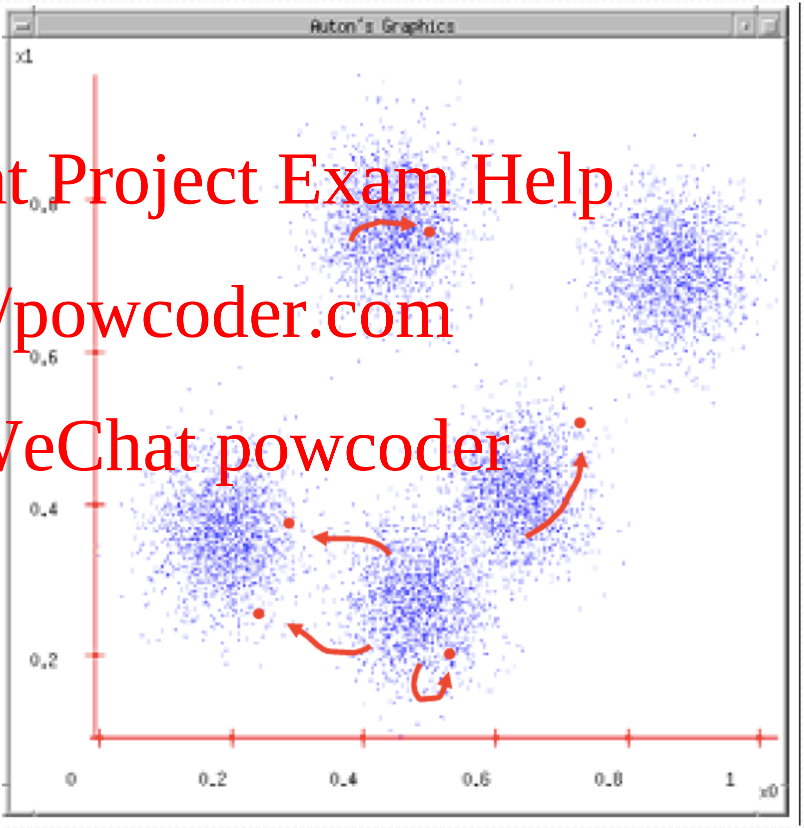
Example:

k=5

Distance metric=euclidean

Dimensions=2

1. Randomly select k cluster Centers
2. Assign closest Center to each data point
3. Update Centers based on assignments from (2)
4. Re-iterate steps 2-3 until convergence or stopping criteria met



The Phases

1. **Initialization:** Randomly select *k* cluster centers
   - Select *k* samples from data as initial centers [Forgy Partition]

2. **Expectation:** Assign each data point go closest center
   - Compare each data point (*N*) to each cluster center (*k*)

   - Distance Metric: Euclidean distance (*D* dimensions)
3. **Maximization:** Update centers based on assignments

   - For each cluster (*k*) compute mean (D dimensions) from data points assigned to that cluster
4. **Evaluate:** Re-iterate steps 2-3 until convergence or stopping criteria met
   - Percentage of data points re-assigned
   - Number of iterations (2-3)

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

A Fast Implementation of *k*-means

- Writing fast code is a process coherent with

"general problem solving behavior"

Newell and Simon, Human Problem Solving (1972), pp. 72-73

- The process of problem solving involves:

1. Understand the **current state**

2. Observe the **internal representation**
3. **Search** among alternatives
4. Select from a set of **choices**

A Fast Implementation of *k*-means

- Following the process of problem solving with *k*-means:
  1. Understand the **current state**
     - Running on a platform
     - Using a specific set of resources
     - Achieving a specific performance
     - Meeting a specific criteria/requirement
  2. Observe the **internal representation**
  3. **Search** among alternatives
  4. Select from a set of **choices**

Assumption:
Starting from a functionally correct reference implementation

Implication:
Must observe the *current state* and *implementation requirements* before starting to solve a problem

A Fast Implementation of *k*-means

- Following the process of problem solving with *k*-means:
  1. Understand the **current state**
  2. Observe the **internal representation**
     - Application structure
       - Identified four phases of execution
     - Implementation concerns
       - Task considerations
       - Data representations
       - Concurrency opportunities
  3. **Search** among alternatives
  4. Select from a set of **choices**

A Fast Implementation of *k*-means

- Following the process of problem solving with *k*-means:
  1. Understand the **current state**
  2. Observe the **internal representation**
     - Application structure
       - Identified four phases of execution
     - Implementation concerns
       - Task considerations
       - Data representations
       - Concurrency opportunities
  3. **Search** among alternatives
  4. Select from a set of **choices**

A Fast Implementation of *k*-means

- Following the process of problem solving with *k*-means:
  1. Understand the **current state**
  2. Observe the **internal representation**
     - Application structure
       - Identified four phases of execution
     - Implementation concerns
       - Task considerations
       - Data representations
       - Concurrency opportunities
  3. **Search** among alternatives
  4. Select from a set of **choices**