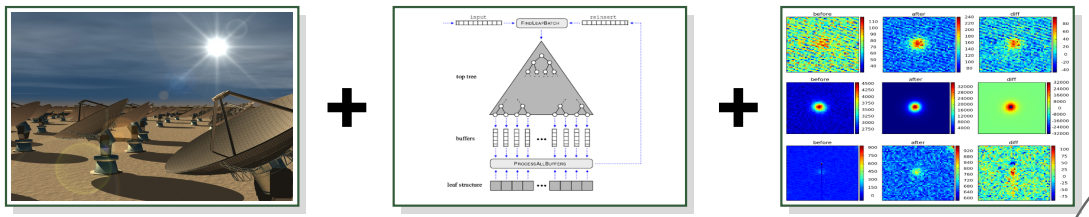
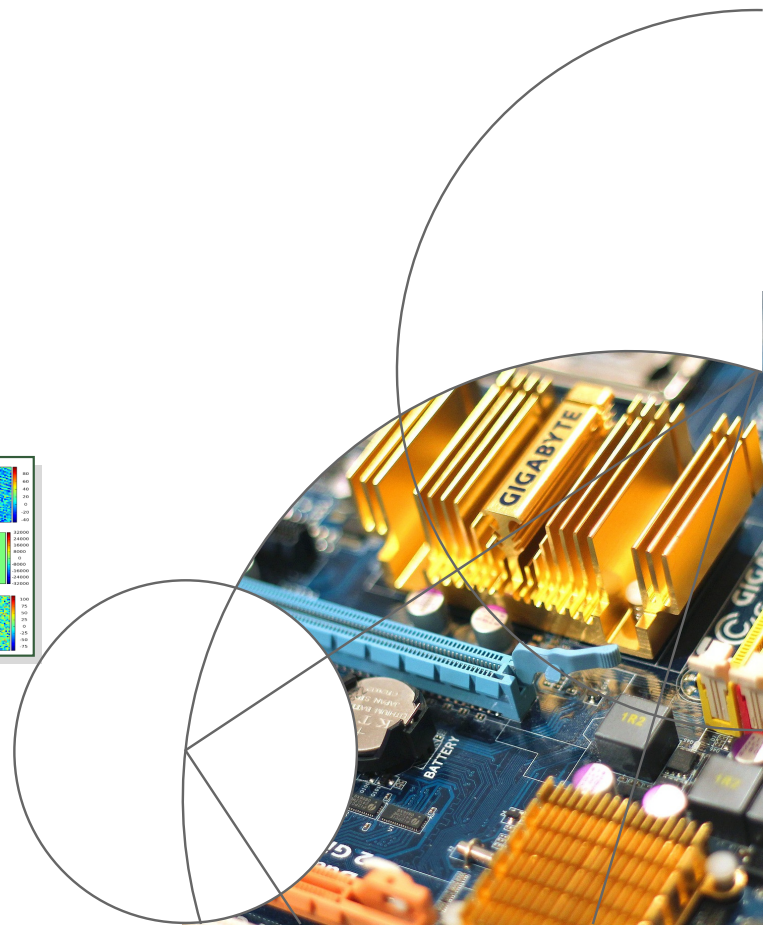
Large-Scale Machine Learning in Astronomy

Fabian Gieseke

Image Group
Department of Computer Science
University of Copenhagen

fabian.gieseke@di.ku.dk





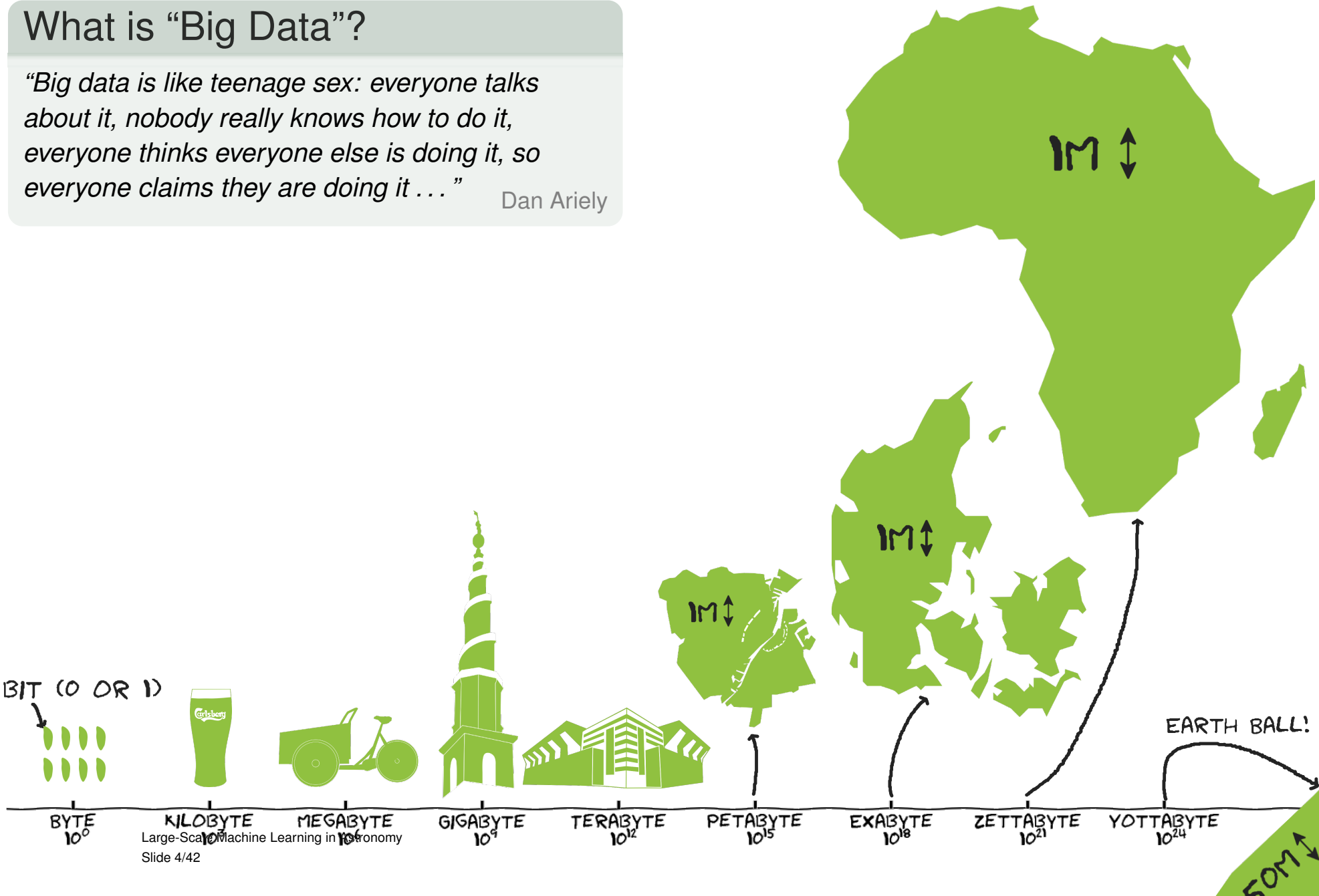
Outline

Outline

What is “Big Data”?

“Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it . . . ”

Dan Ariely



What is “Big Data”?

“Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, everyone claims they are doing it . . . ”

1X IMAGE

1X E-MAIL

BIT (0 OR 1)

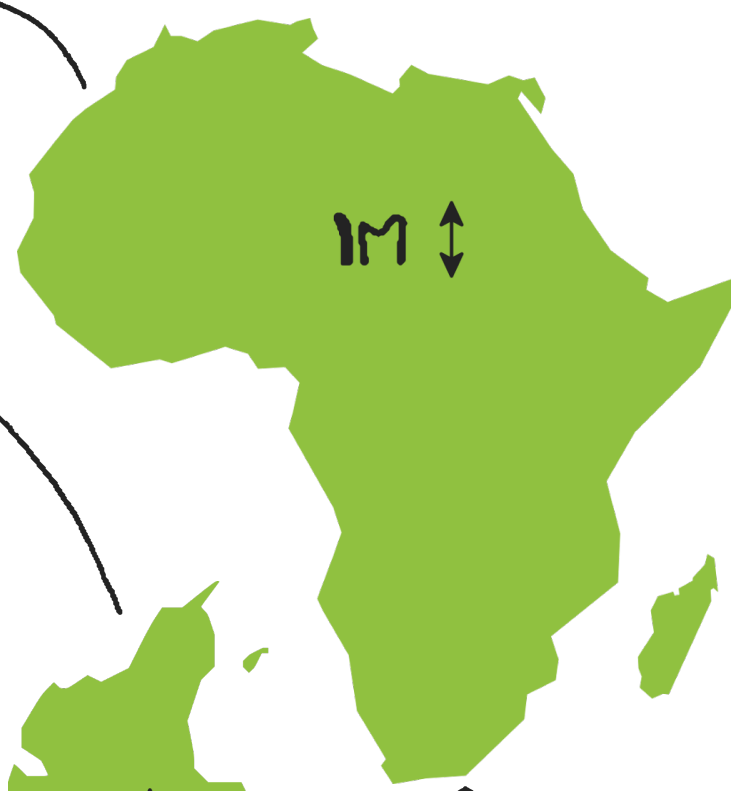
1X CD

1X LAPTOP

50,000 PHONES

TRAFFIC 2016
(WWW.CISCO.COM)

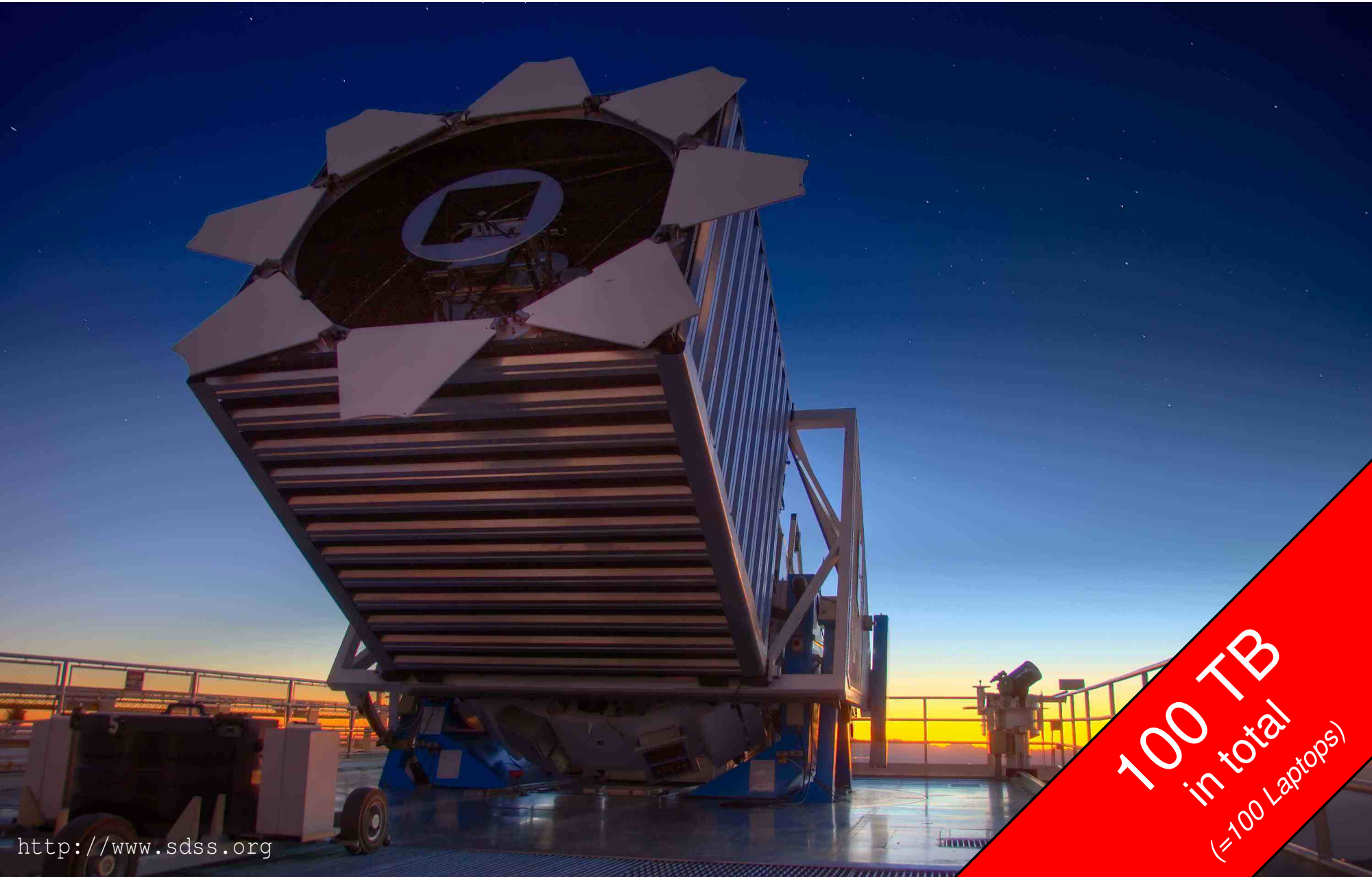
FACEBOOK:
1XPB PER DAY



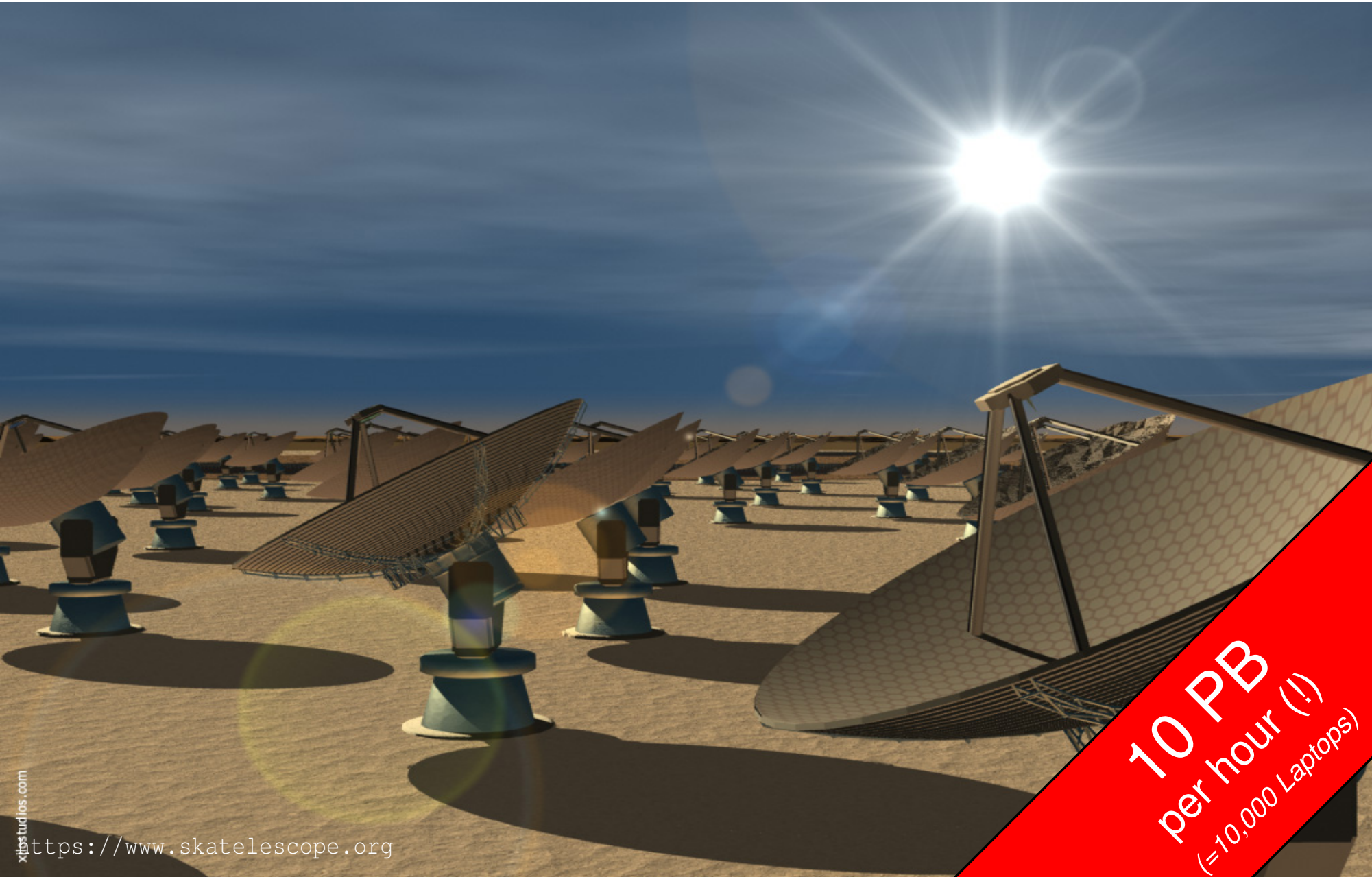
EARTH BALL!

BYTE 10^0 — KILOBYTE 10^3 — MEGABYTE 10^6 — GIGABYTE 10^9 — TERABYTE 10^{12} — PETABYTE 10^{15} — EXABYTE 10^{18} — ZETTABYTE 10^{21} — YOTTABYTE 10^{24}

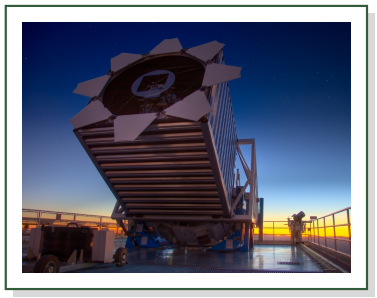
Today: Telescopes (2000–2017)



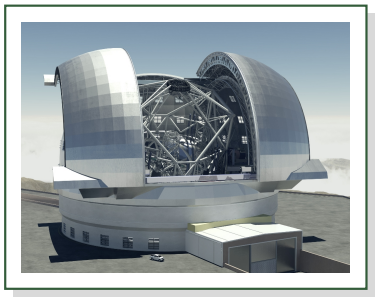
Tomorrow: Telescopes (2020+)



Astronomy and Machine Learning?

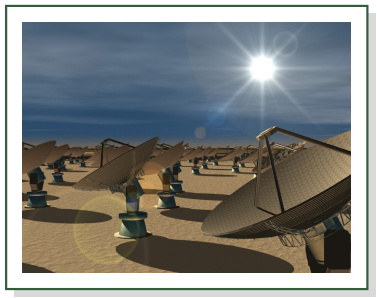


2015 → SDSS
(in total: 100TB)



2024 → EELT
(per night: 2TB)

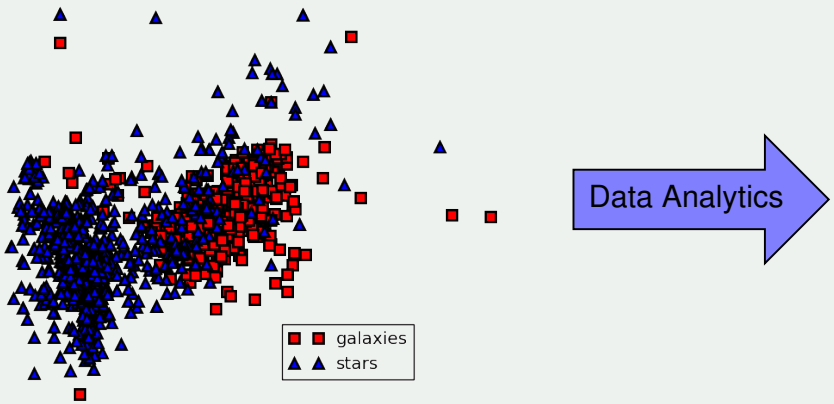
2021 → LSST
(per night: 30TB)



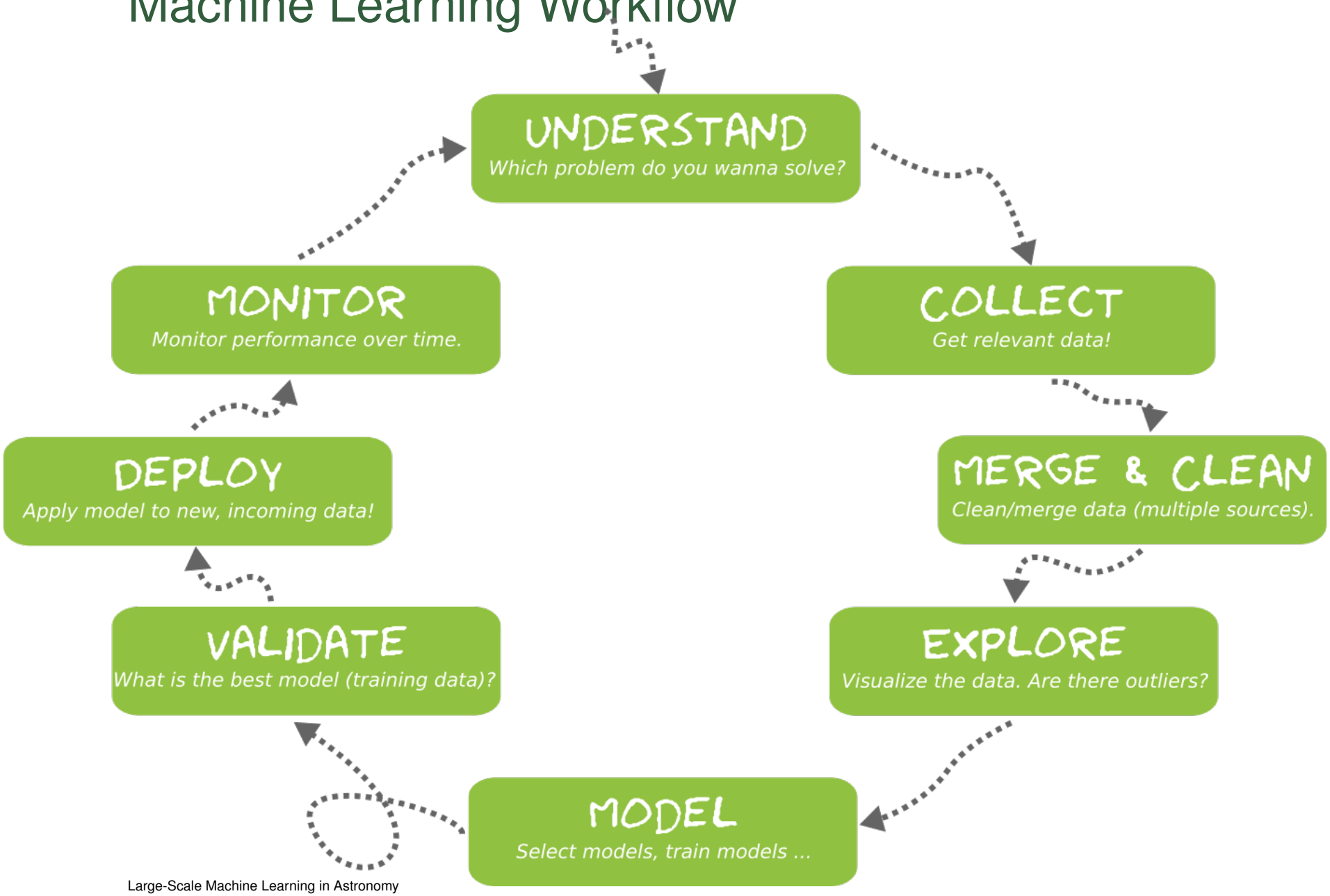
2025 → SKA
(per hour: 10PB)

Challenges

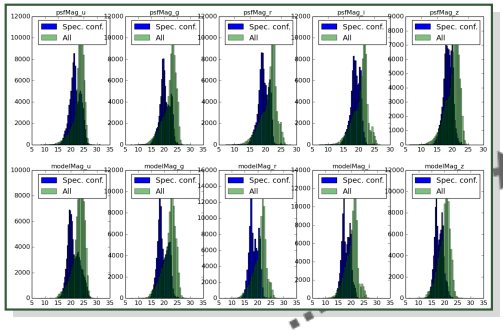
Find interesting objects such as distant galaxies or very rare stars! Combine various data sources! Handle billions of objects per night → Process and analyze all the data efficiently and at low cost!



Machine Learning Workflow

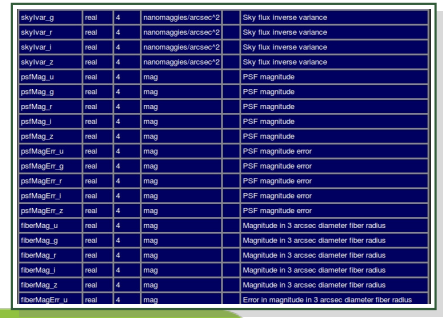


Machine Learning Workflow



UNDERSTAND

Which problem do you wanna solve?



skyIvar_g	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyIvar_r	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyIvar_i	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyIvar_z	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
psfMag_u	real	4	mag	PSF magnitude
psfMag_g	real	4	mag	PSF magnitude
psfMag_r	real	4	mag	PSF magnitude
psfMag_i	real	4	mag	PSF magnitude
psfMag_z	real	4	mag	PSF magnitude
psfMagErr_u	real	4	mag	PSF magnitude error
psfMagErr_g	real	4	mag	PSF magnitude error
psfMagErr_r	real	4	mag	PSF magnitude error
psfMagErr_i	real	4	mag	PSF magnitude error
psfMagErr_z	real	4	mag	PSF magnitude error
fiberMag_u	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_g	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_r	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_i	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_z	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMagErr_u	real	4	mag	Error in magnitude in 3 arcsec diameter fiber radius

COLLECT

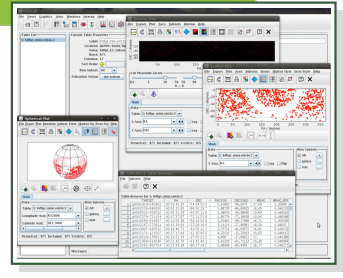
Get relevant data!

MERGE & CLEAN

Clean/merge data (multiple sources).

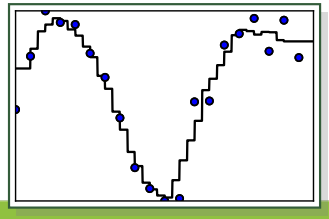
EXPLORE

Visualize the data. Are there outliers?



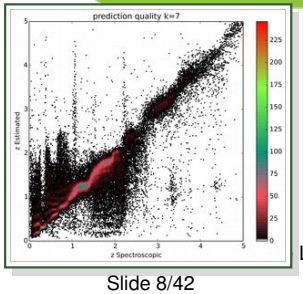
MODEL

Select models, train models ...



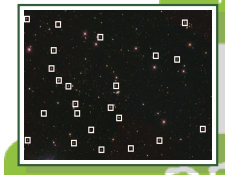
VALIDATE

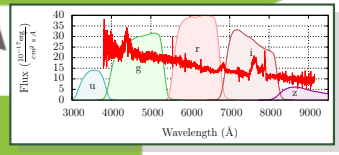
What is the best model (training data)?



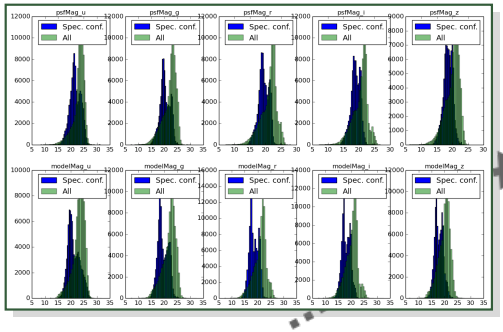
DEPLOY

Apply model to new, incoming data!



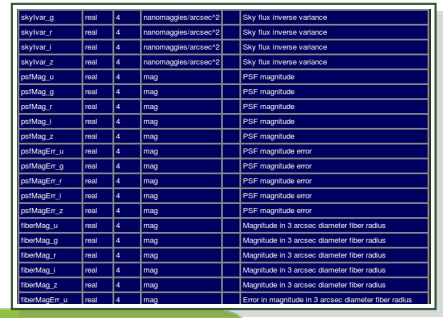


Machine Learning Workflow



UNDERSTAND

Which problem do you wanna solve?



skyivar_g	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyivar_r	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyivar_i	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
skyivar_z	real	4	nanomaggies/arcsec^2	Sky flux inverse variance
psfMag_u	real	4	mag	PSF magnitude
psfMag_g	real	4	mag	PSF magnitude
psfMag_r	real	4	mag	PSF magnitude
psfMag_i	real	4	mag	PSF magnitude
psfMag_z	real	4	mag	PSF magnitude
psfMagErr_u	real	4	mag	PSF magnitude error
psfMagErr_g	real	4	mag	PSF magnitude error
psfMagErr_r	real	4	mag	PSF magnitude error
psfMagErr_i	real	4	mag	PSF magnitude error
psfMagErr_z	real	4	mag	PSF magnitude error
fiberMag_u	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_g	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_r	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_i	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMag_z	real	4	mag	Magnitude in 3 arcsec diameter fiber radius
fiberMagErr_u	real	4	mag	Error in magnitude in 3 arcsec diameter fiber radius

COLLECT

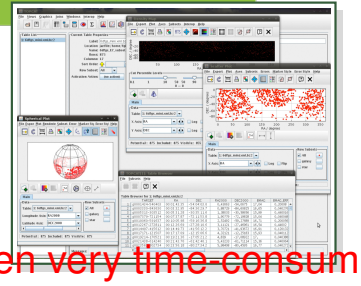
Get relevant data!

MERGE & CLEAN

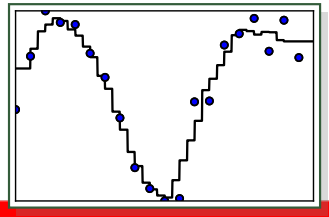
Clean/merge data (multiple sources).

EXPLORE

Visualize the data. Are there outliers?



Often very time-consuming!

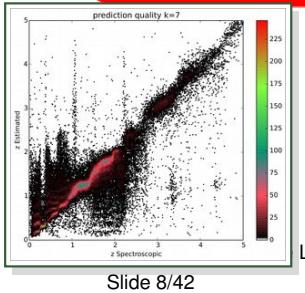


MODEL

Select models, train models ...

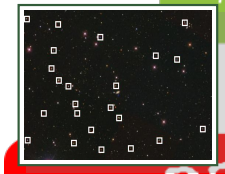
VALIDATE

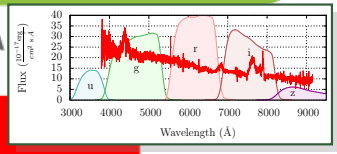
What is the best model (training data)?



DEPLOY

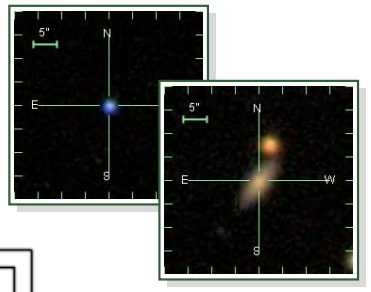
Apply model to new, incoming data!

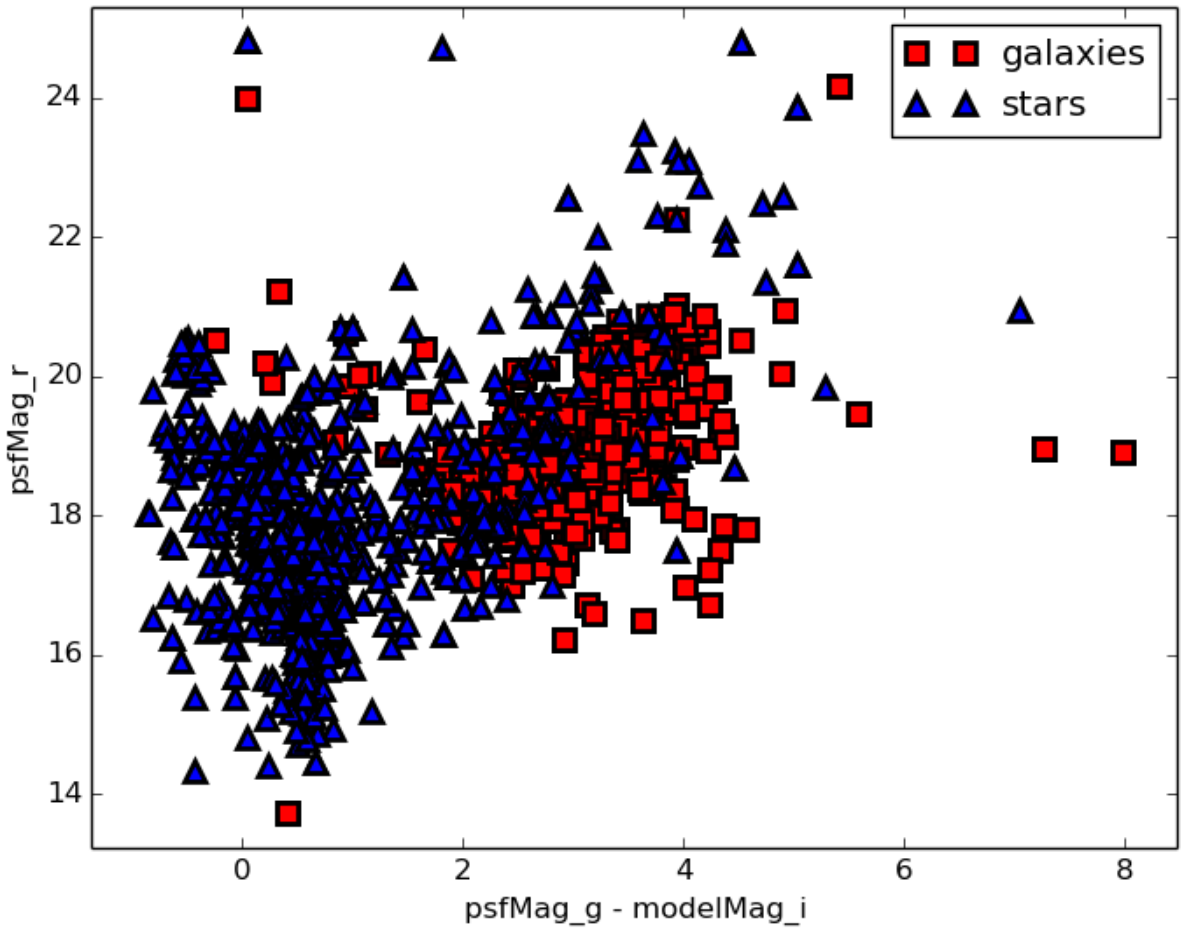




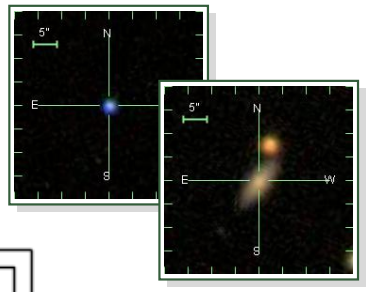
Outline

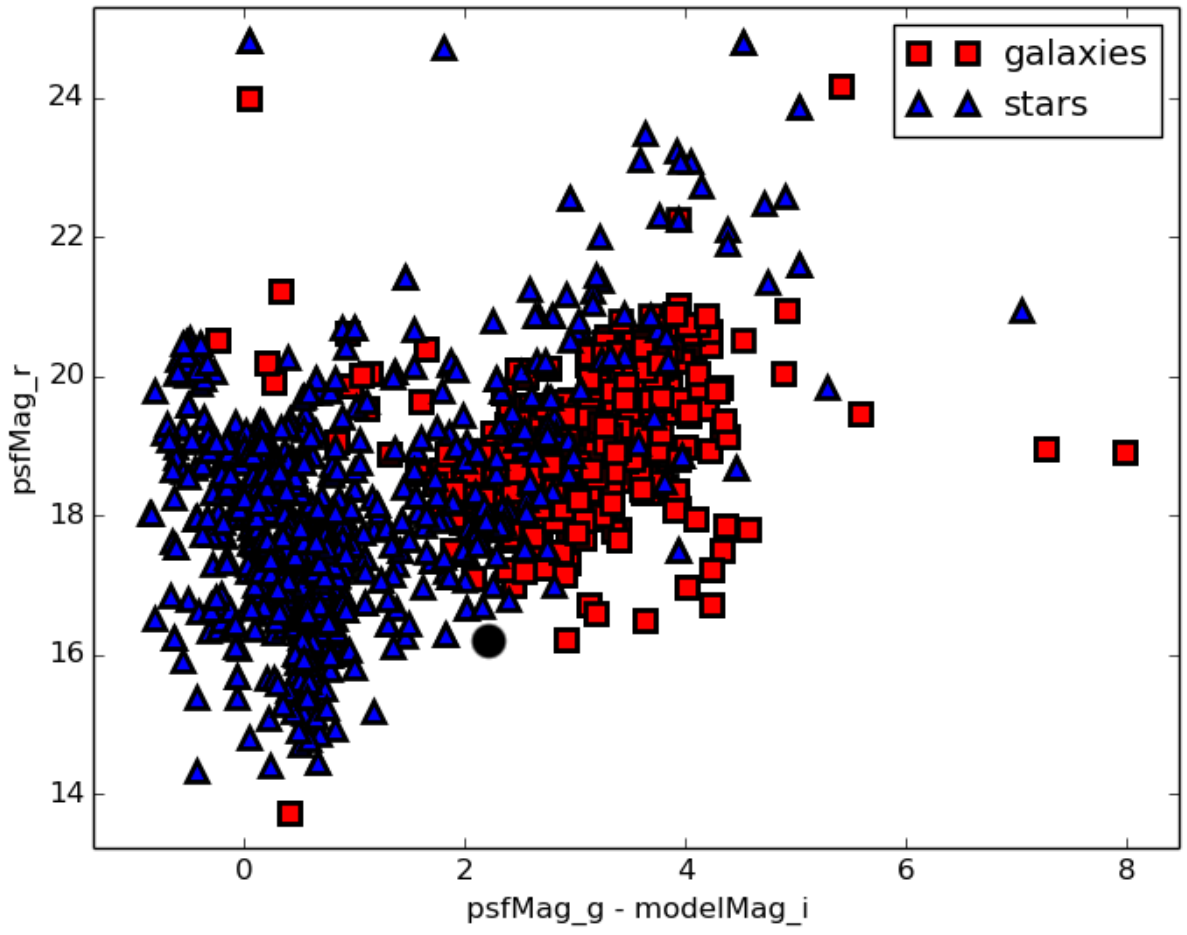
A Simple Task (?)



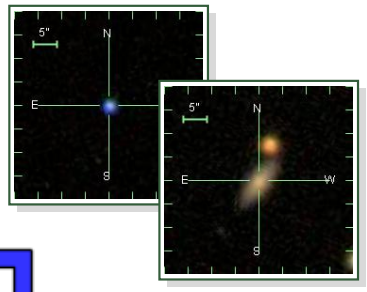


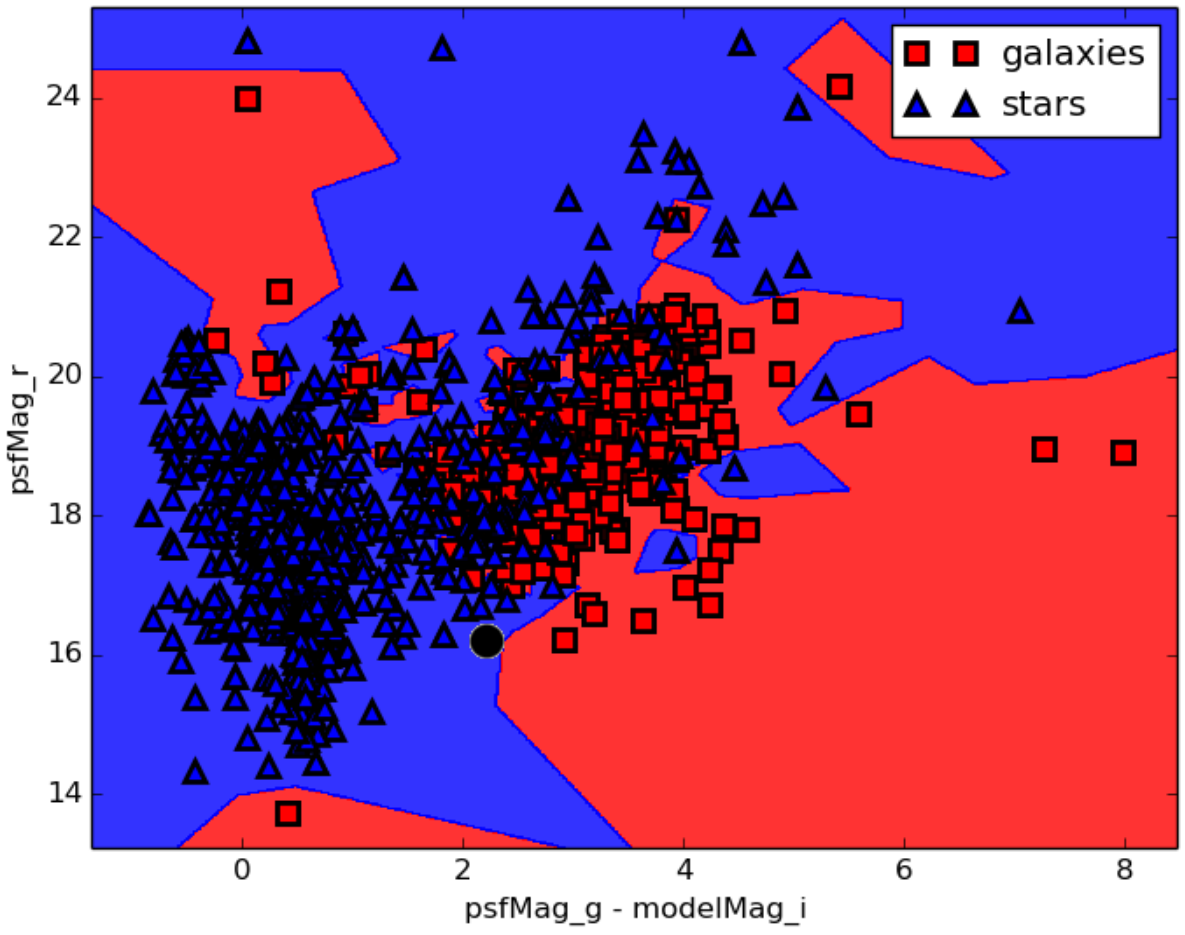
A Simple Task (?)



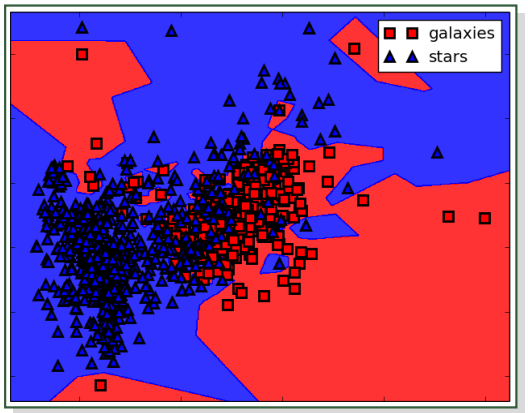


A Simple Task (?)

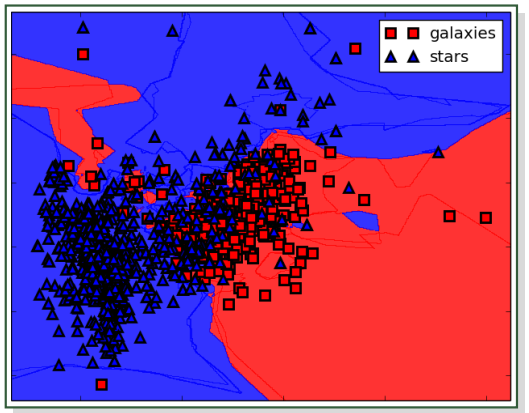




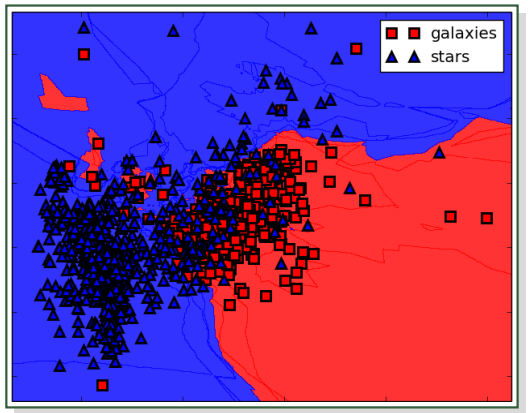
Example: Nearest Neighbor Classification



$k = 1$



$k = 5$



$k = 10$

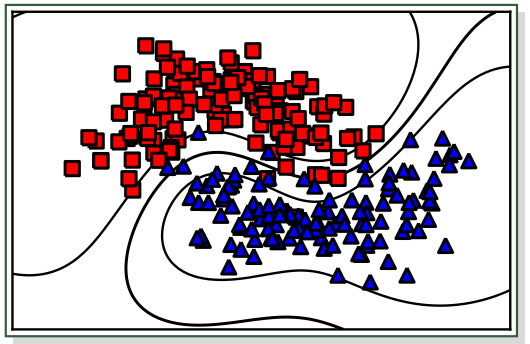
The k-nearest neighbor classification algorithm

Require: Let $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \{1, \ldots, C\}$ be the set of training examples and k be the number of nearest neighbors.

1: **for** each test example **x do**
2: Compute the distance $D(\mathbf{x}, \mathbf{x}_i)$ to each training example $\mathbf{x}_i$.
3: Select $N_{\mathbf{x}} \subset T$, the set of the k closest training examples to $\mathbf{x}$.
4: $f(\mathbf{x}) = \text{argmax}_c \sum_{(\mathbf{x}_i, y_i) \in N_{\mathbf{x}}} \mathbb{I}(c = y_i)$
5: **end for**

Can you deal high-dimensional data?
Can you handle millions of training/test points?

Example: Support Vector Machines



$$\underset{\beta\in[0,C]^n}{\text{maximize}} \sum_{i=1}^{n}\beta_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle$$

$$\text{s.t.} \sum_{i=1}^{n}\beta_i y_i = 0$$

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x}$$

$$\text{s.t. } \mathbf{G}\mathbf{x} \leq \mathbf{g}$$

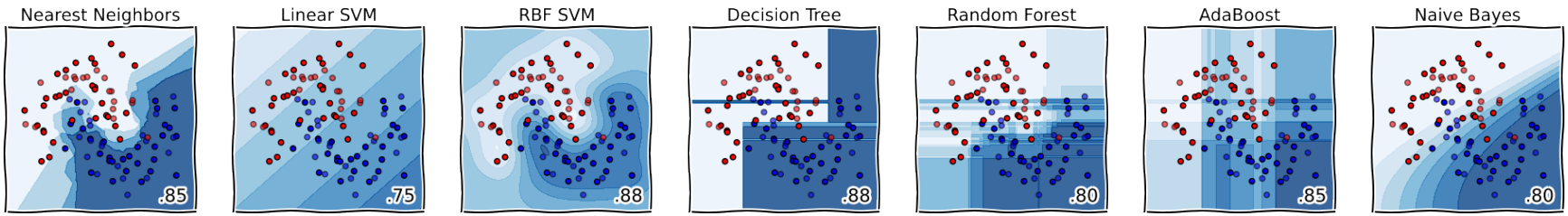
$$\mathbf{A}\mathbf{x} = \mathbf{a}$$

1. $\mathbf{Q} = \mathbf{K} \odot \mathbf{y}\mathbf{y}^\mathsf{T} \in \mathbb{R}^{n\times n}$ and $\mathbf{c} = (-1,\ldots,-1)^\mathsf{T} \in \mathbb{R}^n$
2. $\mathbf{G} = \begin{pmatrix} -\mathbf{I} \\ \mathbf{I} \end{pmatrix}$ with $\mathbf{I} \in \mathbb{R}^{n\times n}$ and $\mathbf{g} = (0,\ldots,0,C,\ldots,C)^\mathsf{T} \in \mathbb{R}^{2n}$
3. $\mathbf{A} = \mathbf{y}^\mathsf{T} \in \mathbb{R}^{1\times n}$ and $\mathbf{a} = 0 \in \mathbb{R}^1$

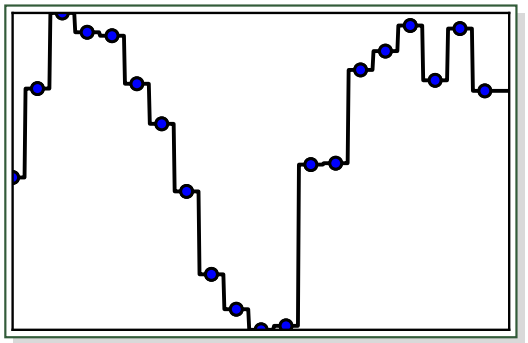
Here, $\odot$ denotes the elementwise product, $\mathbf{y} = (y_1,\ldots,y_n)^\mathsf{T} \in \mathbb{R}^n$, and kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ with $\mathbf{K}_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j\rangle$.

Solving such a quadratic program generally takes $\mathcal{O}(n^3)$ time ...

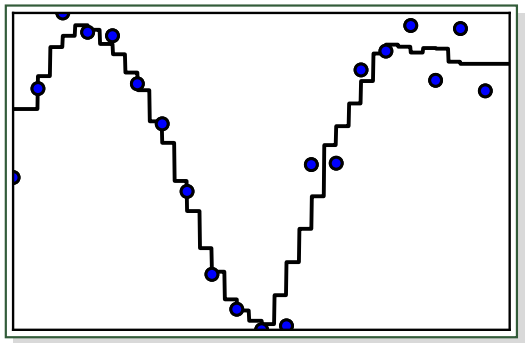
More Classifiers!



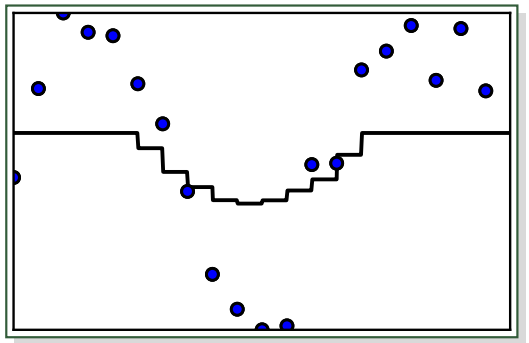
Example: Nearest Neighbor Regression



$k = 1$



$k = 5$



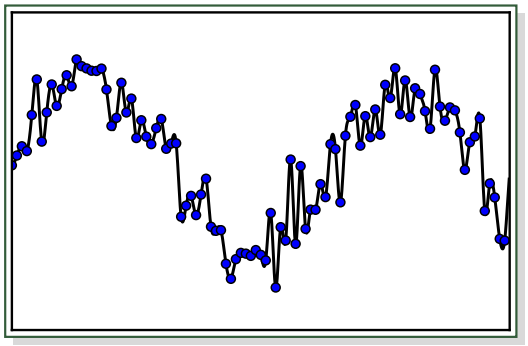
$k = 10$

The k-nearest neighbor regression algorithm

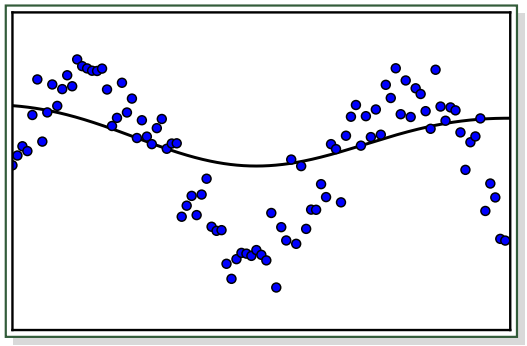
Require: Let $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$ be the set of training examples and k be the number of nearest neighbors.

1: **for** each test example **x do**
2: Compute the distance $D(\mathbf{x}, \mathbf{x}_i)$ to each training example $\mathbf{x}_i$.
3: Select $N_{\mathbf{x}} \subset T$, the set of the k closest training examples to $\mathbf{x}$.
4: $f(\mathbf{x}) = \frac{1}{k} \sum_{(\mathbf{x}_i, y_i) \in N_{\mathbf{x}}} y_i$
5: **end for**

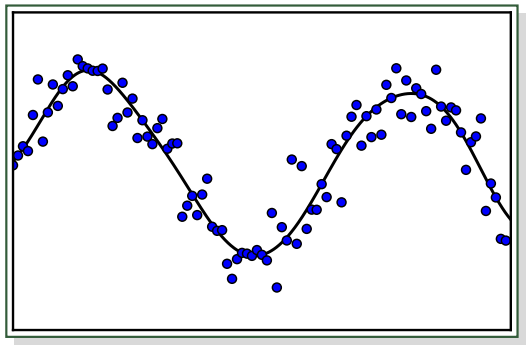
Example: Regularized Least-Squares



$\lambda = \text{small}$



$\lambda = \text{large}$



$\lambda = \text{middle}$

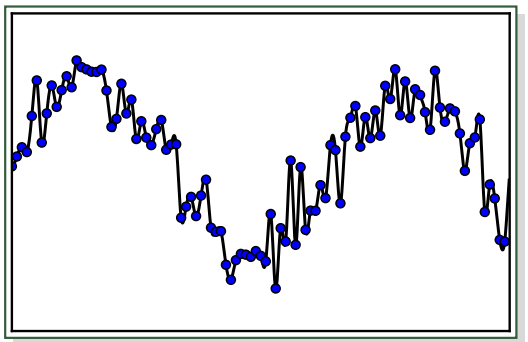
Models

Let $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$ be the set of training examples, $\mathbf{X} \in \mathbb{R}^{n \times d}$ containing the patterns $\mathbf{x}_i$ as rows, and $\lambda > 0$. Goal: Find a model $f \in \mathcal{H}$ in a hypothesis space $\mathcal{H}$ that minimizes

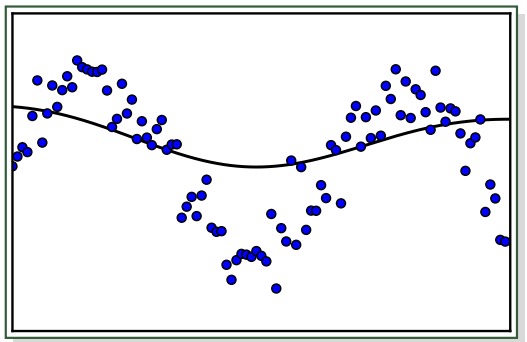
$$\underset{f \in \mathcal{H}}{\text{minimize}} \underbrace{\sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2}_{\text{Small Loss}} + \underbrace{\lambda \|f\|^2}_{\text{Small Complexity}} \tag{1}$$

Here: Models of the form $f(\mathbf{x}) = \sum_{j=1}^{n} c_j K(\mathbf{x}_j, \mathbf{x})$, where $\mathbf{c} \in \mathbb{R}^n$ and $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a so-called kernel (many machine learning techniques are based on this formulation!).

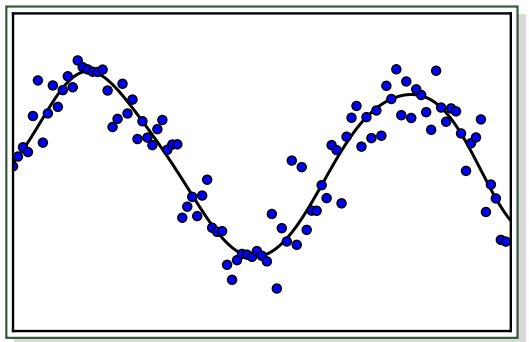
Example: Regularized Least-Squares



$\lambda =$ small



$\lambda =$ large



$\lambda =$ middle

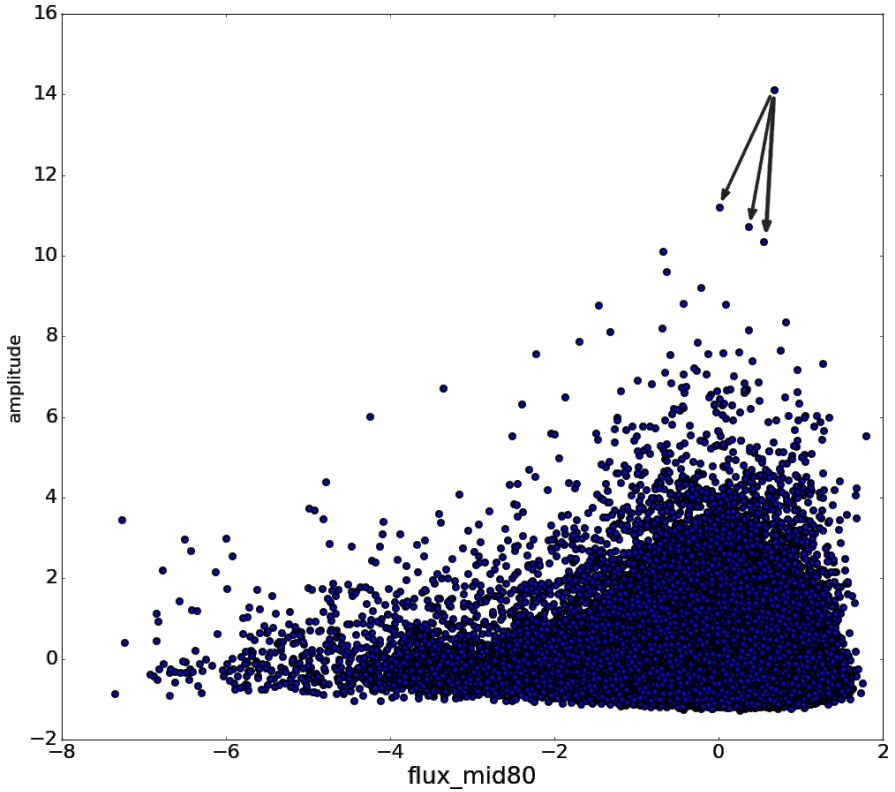
Computational Complexities

1. Training: One can compute optimal coefficients in $\mathcal{O}(n^3)$ operations. In addition, one needs to compute the kernel matrix beforehand (typically $\mathcal{O}(\cdot n^2)$ time and space).
2. Testing: Once the coefficients $\mathbf{c}^*$ have been computed, one can apply the model f to new $\mathbf{x} \in \mathbb{R}^d$ via $f(\mathbf{x}) = \sum_{j=1}^{n} c_j K(\mathbf{x}_j, \mathbf{x})$. This takes $\mathcal{O}(n)$ time per instance!

Space Reduction & Speed-Ups: We need $\mathcal{O}(n^2)$ space to store $\mathbf{K}$. Various ways exist to reduce the runtime/space consumption. A prominent one is to approximate the kernel matrix (e.g., via $\tilde{\mathbf{K}} = \mathbf{K}_R^T (\mathbf{K}_{R,R})^{-1} \mathbf{K}_R \in \mathbb{R}^{n \times n}$ with R rows/columns being randomly selected).

Can you train a model for one million training instances?
Can you deal with one billion test points?

Example: Unsupervised Learning

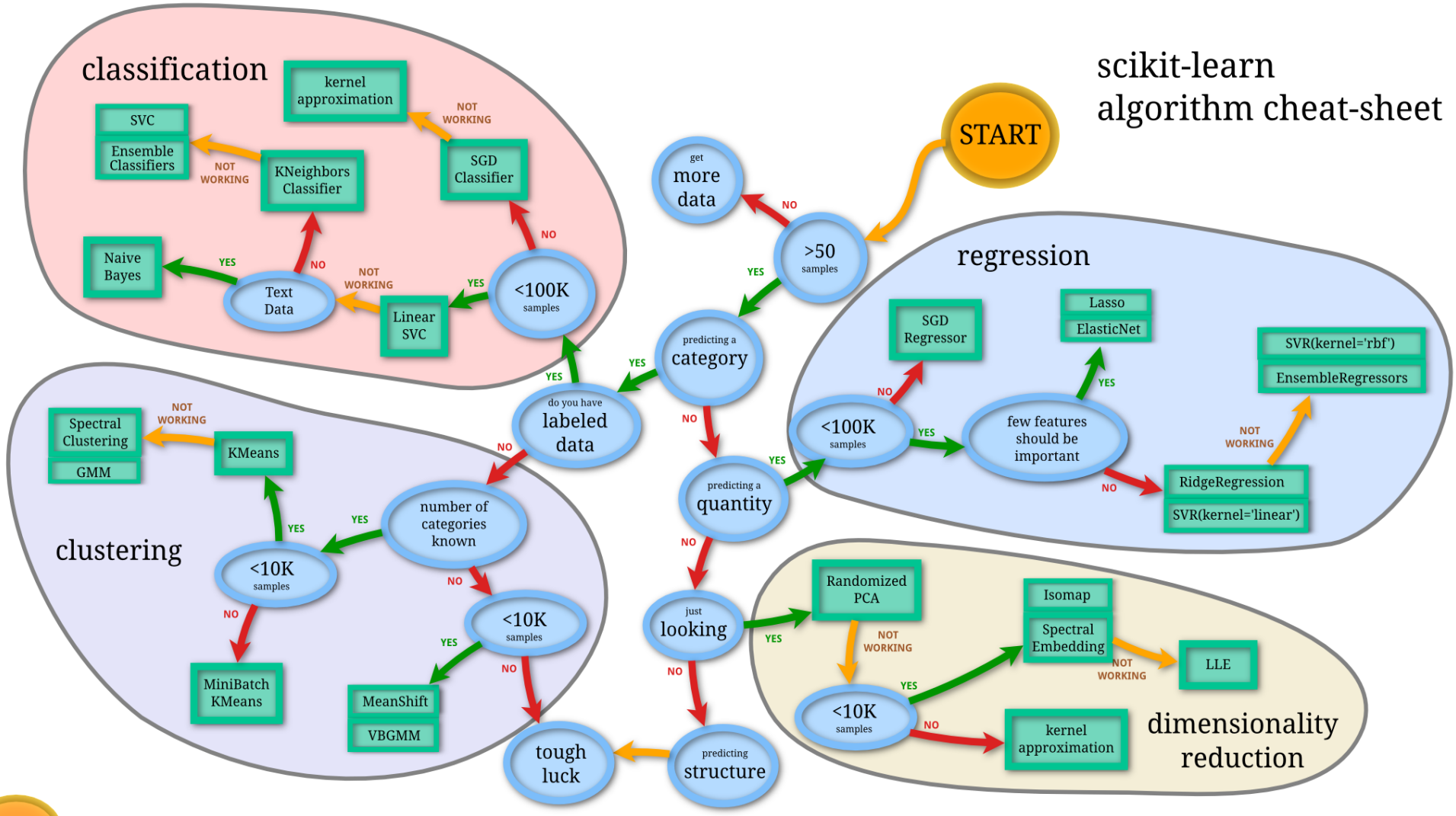


Outlier Detection

"I have 50 million objects and each of them is described via 20 values (features). Can you find the outliers for me, i.e., objects that are somehow different from the other ones?"

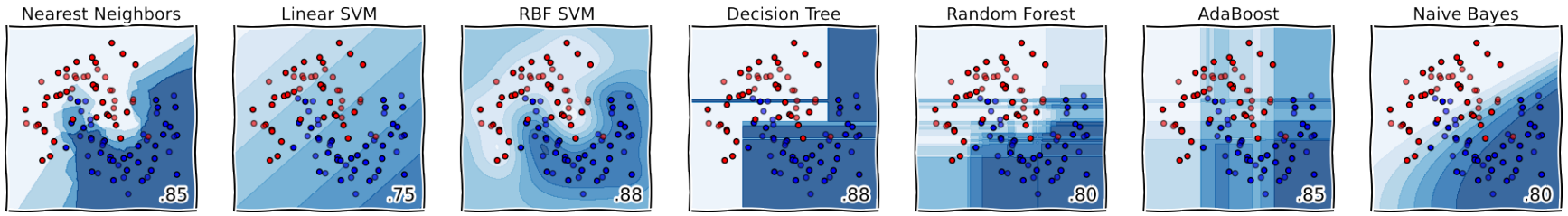
Joint work with Ashish Mahabal

Machine Learning: Many Problems+Techniques!

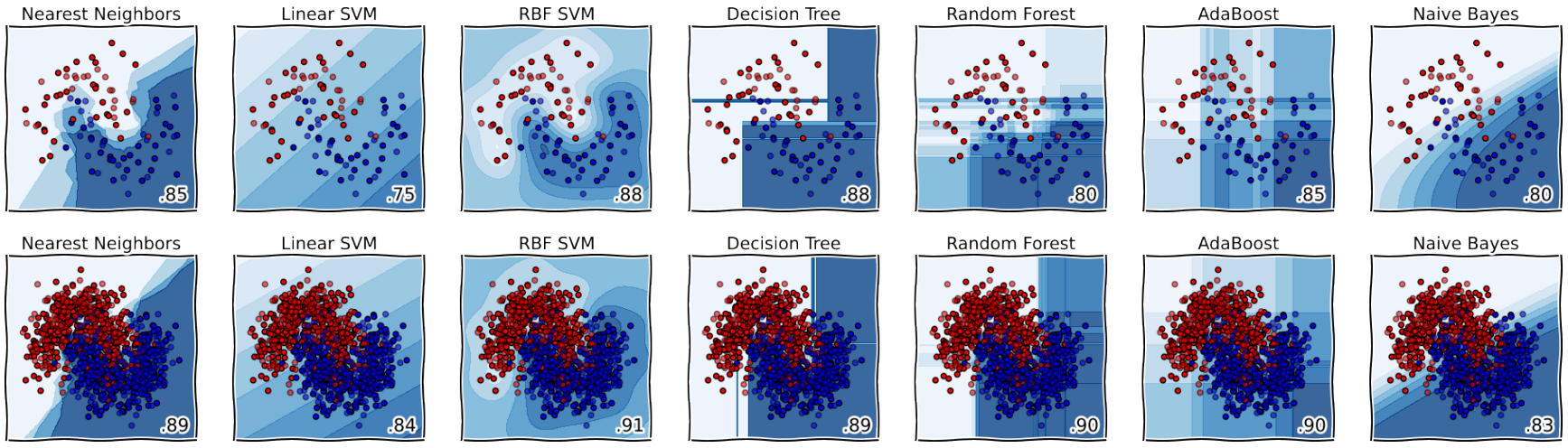


http://scikit-learn.org/stable/tutorial/machine_learning_map/

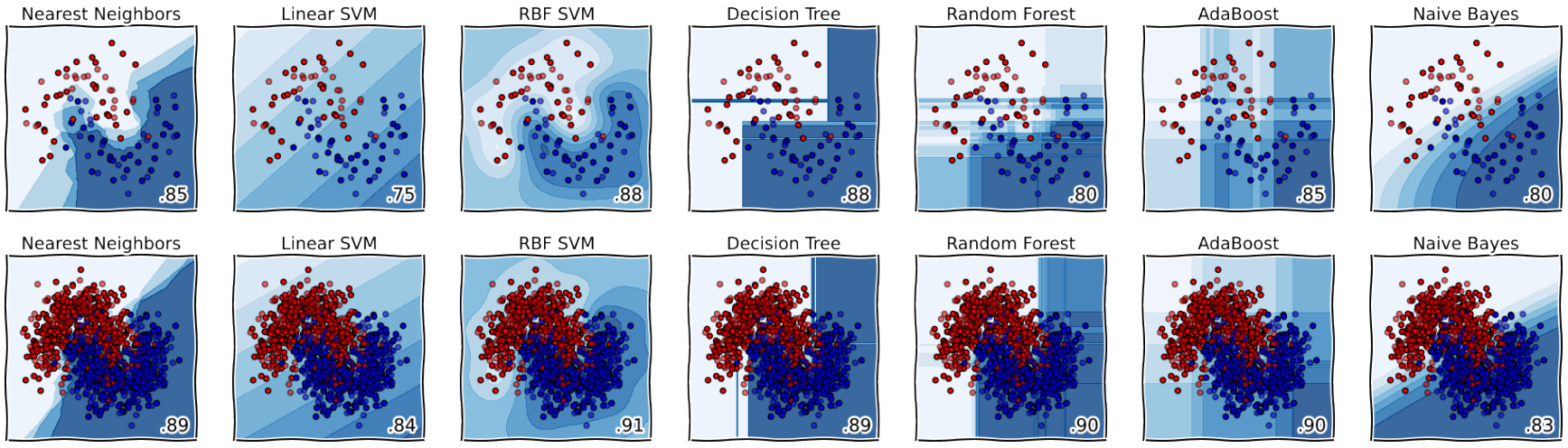
Why Big Data?


Nearest Neighbors
.85
Linear SVM
.75
RBF SVM
.88
Decision Tree
.88
Random Forest
.80
AdaBoost
.85
Naive Bayes
.80

Why Big Data?


Nearest Neighbors
.85
Linear SVM
.75
RBF SVM
.88
Decision Tree
.88
Random Forest
.80
AdaBoost
.85
Naive Bayes
.80
Nearest Neighbors
.89
Linear SVM
.84
RBF SVM
.91
Decision Tree
.89
Random Forest
.90
AdaBoost
.90
Naive Bayes
.83

Why Big Data?



Large-Scale Machine Learning

1. Interface: Machine Learning + Data Structures + Optimization + HPC + . . .
   (often depends on the particular application domain!)
2. Key Question: How can we analyze all the data efficiently and at low cost?

"Often, it is not the best algorithm that wins,
but the one that has the most data!"

[Andrew Ng]

Big Computers

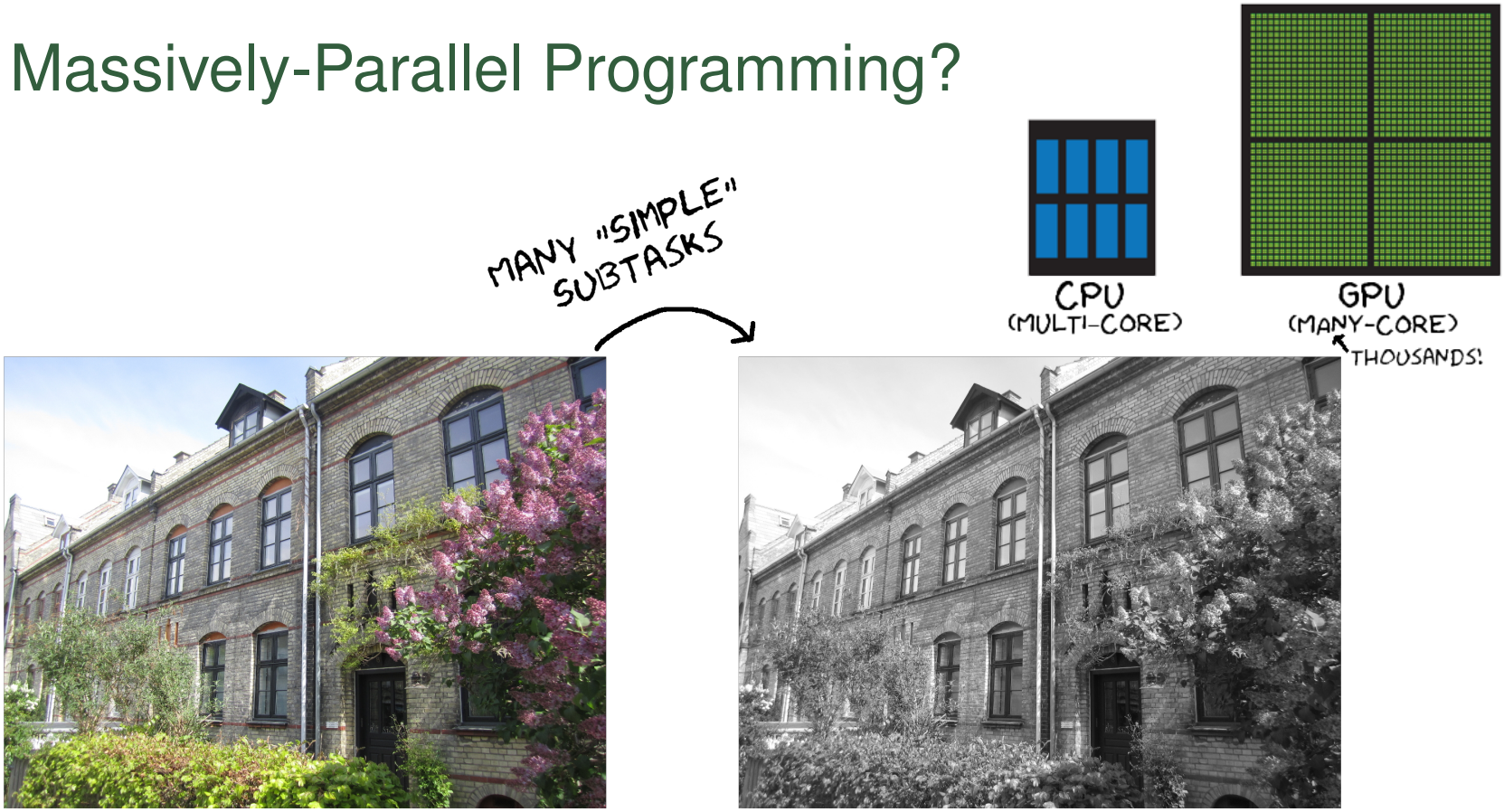
https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/

Big Computers

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
7	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233

http://www.top500.org

Massively-Parallel Programming?

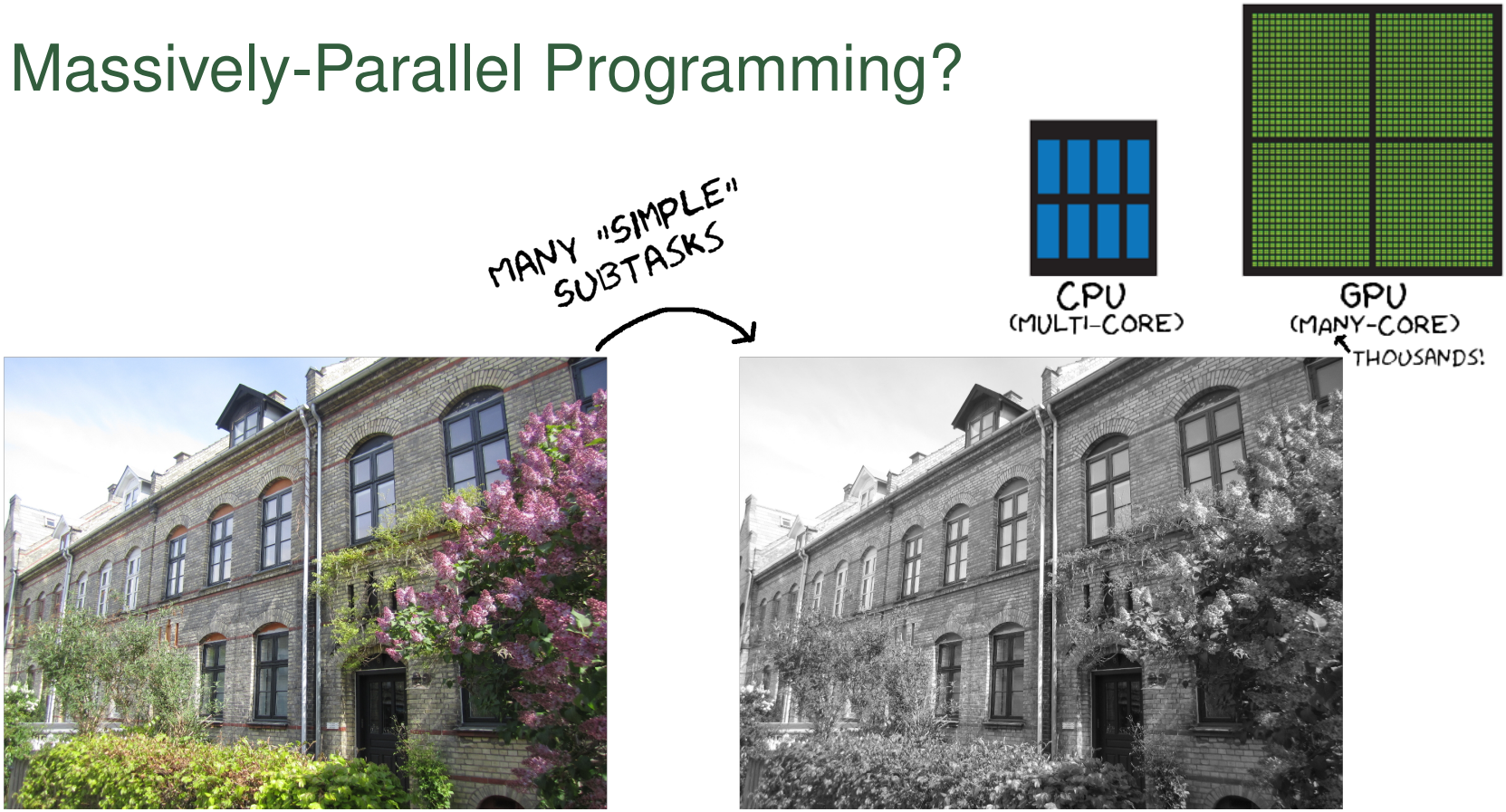


Graphics Processing Units (GPUs)

Can nowadays also be used for general computations and are well-suited for massively-parallel programming. Example: Adding two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{10000}$

1. CPU: Computes $x_1 + y_1$, $x_2 + y_2$, ... (sequentially)
2. GPU: Core i computes $x_i + y_i$ (in parallel)

Massively-Parallel Programming?



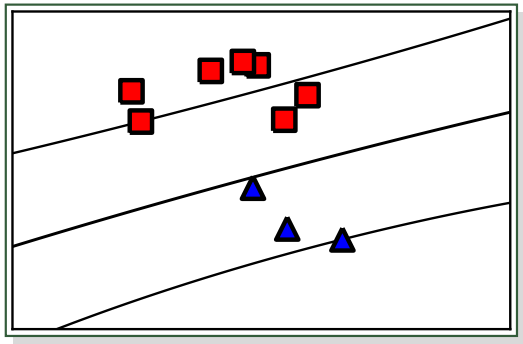
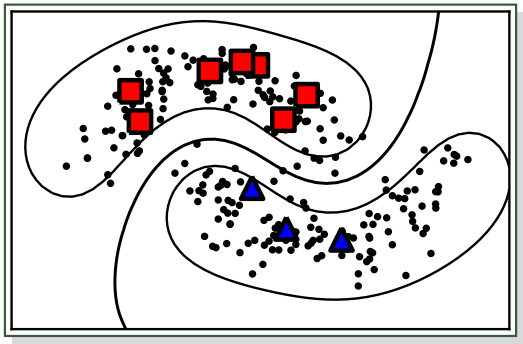
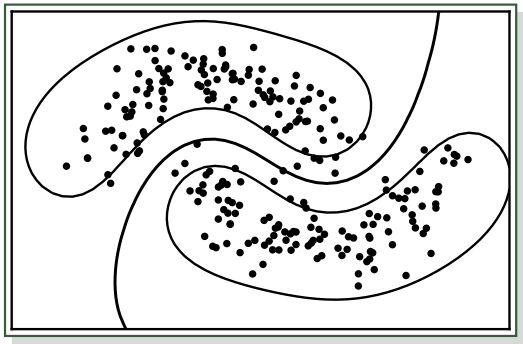
Graphics Processing Units (GPUs)

Can nowadays also be used for general computations and are well-suited for massively-parallel programming. Example: Adding two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{10000}$

1. CPU: Computes $x_1 + y_1$, $x_2 + y_2$, ... (sequentially)
2. GPU: Core i computes $x_i + y_i$ (in parallel)

Challenge for ML
Make computations "cheap"!

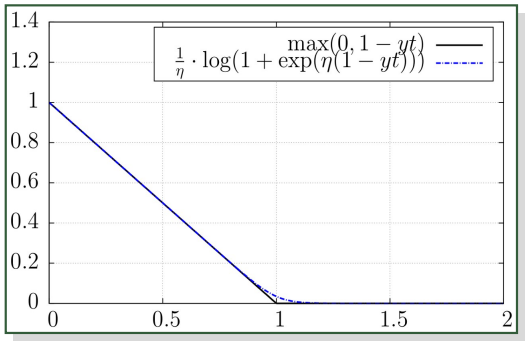
Example I: Semi-Supervised SVMs

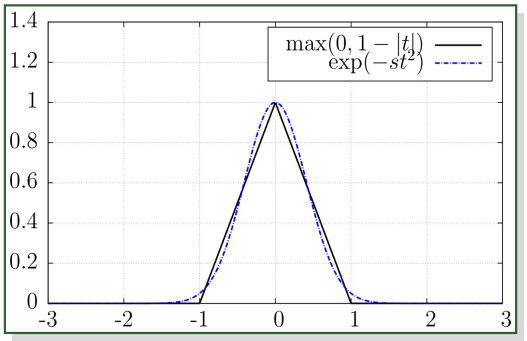




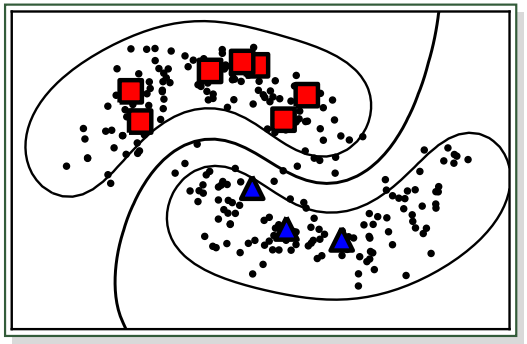
HARD OPTIMIZATION PROBLEM

$$\begin{aligned} &\underset{\substack{\mathbf{y}\in\{-1,+1\}^u,\\ \mathbf{w}\in\mathcal{H},\, b\in\mathbb{R},\, \xi'\in\mathbb{R}^l,\, \xi\in\mathbb{R}^u}}{\text{minimize}} && \frac{1}{2}\|\mathbf{w}\|^2 + C'\sum_{i=1}^{l}\xi'_i + C\sum_{i=1}^{u}\xi_i \\ &\text{s.t.} && y'_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b\right) \geq 1-\xi'_i,\ \xi'_i \geq 0, \\ &\text{and} && y_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_{l+i})\rangle\right) + b) \geq 1-\xi_i,\ \xi_i \geq 0 \end{aligned}$$

Example I: Semi-Supervised SVMs



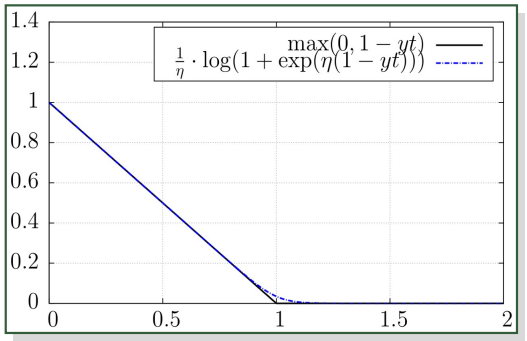


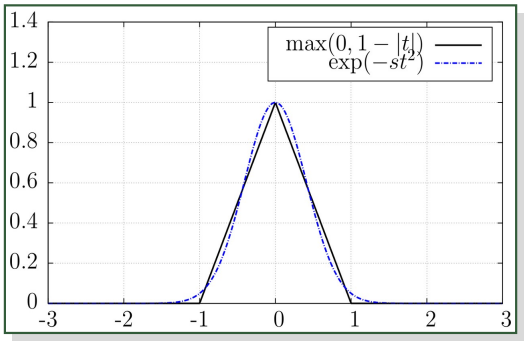


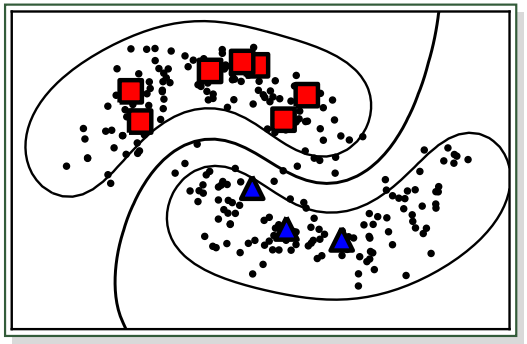
Quasi-Newton Framework (Simplified)

1: Initialize matrices
2: ...
3: **for** $i = 1$ **to** τ **do**
4: ...
5: **while** termination criteria not fulfilled **do**
6: Compute $F_{\alpha_i}(\mathbf{c}_j)$ and $\nabla F_{\alpha_i}(\mathbf{c}_j)$
7: ...
8: **end while**
9: ...
10: **end for**

Example I: Semi-Supervised SVMs







Quasi-Newton Framework (Simplified)

1: Initialize matrices
2: ...
3: **for** $i = 1$ **to** τ **do**
4: ...
5: **while** termination criteria not fulfilled **do**
6: Compute $F_{\alpha_i}(\mathbf{c}_j)$ and $\nabla F_{\alpha_i}(\mathbf{c}_j)$
7: ...
8: **end while**
9: ...
10: **end for**

Example I: Speed-Up

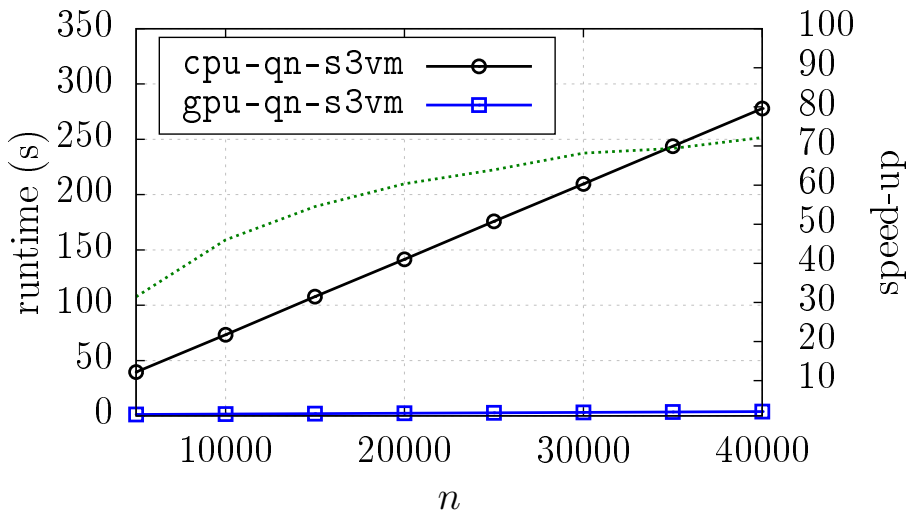
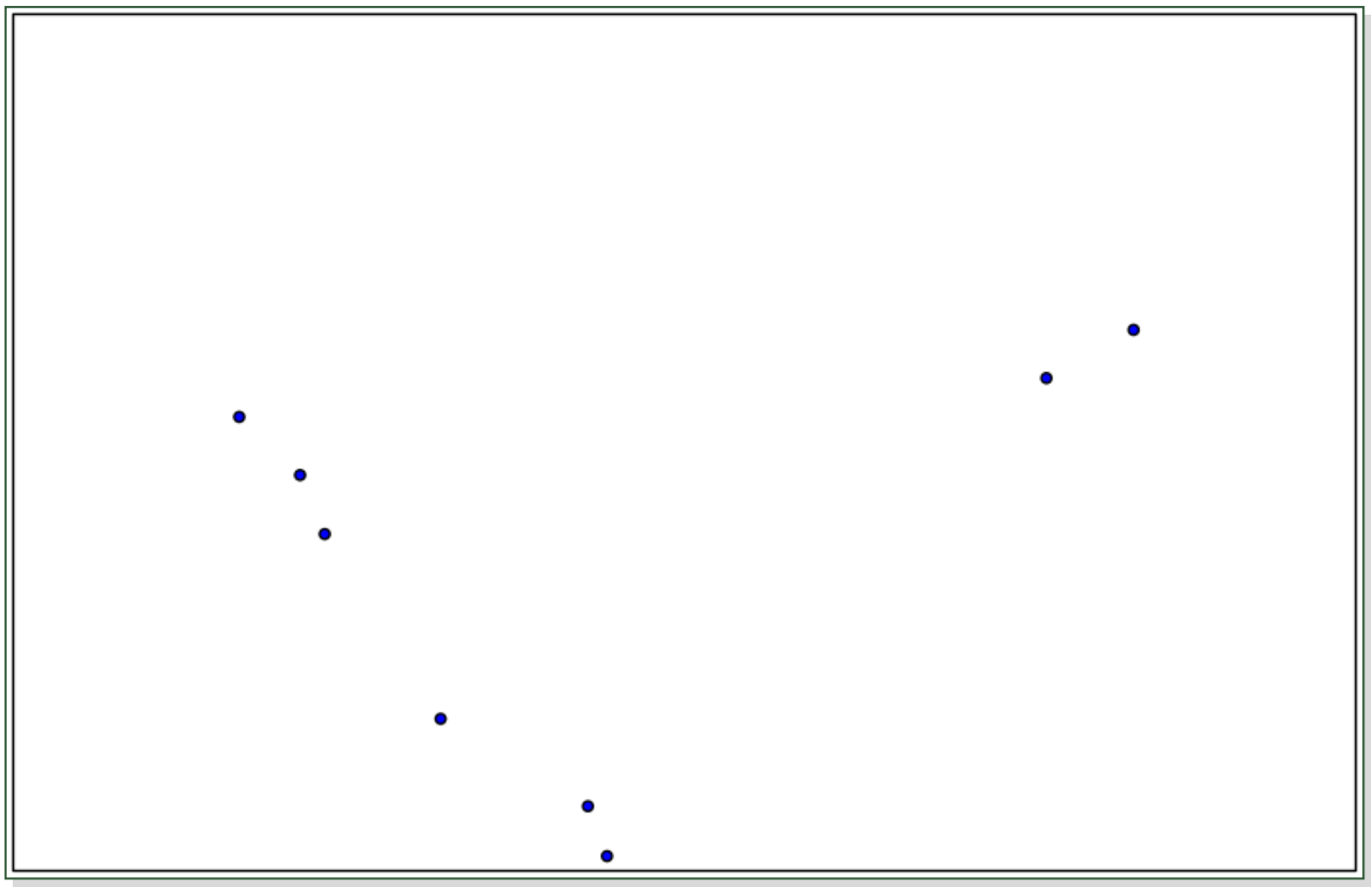


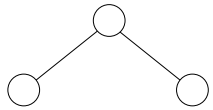
Figure 3: Runtime comparison between the CPU implementation and its GPU variant given the `epsilon` data set ($\lambda = 1$).

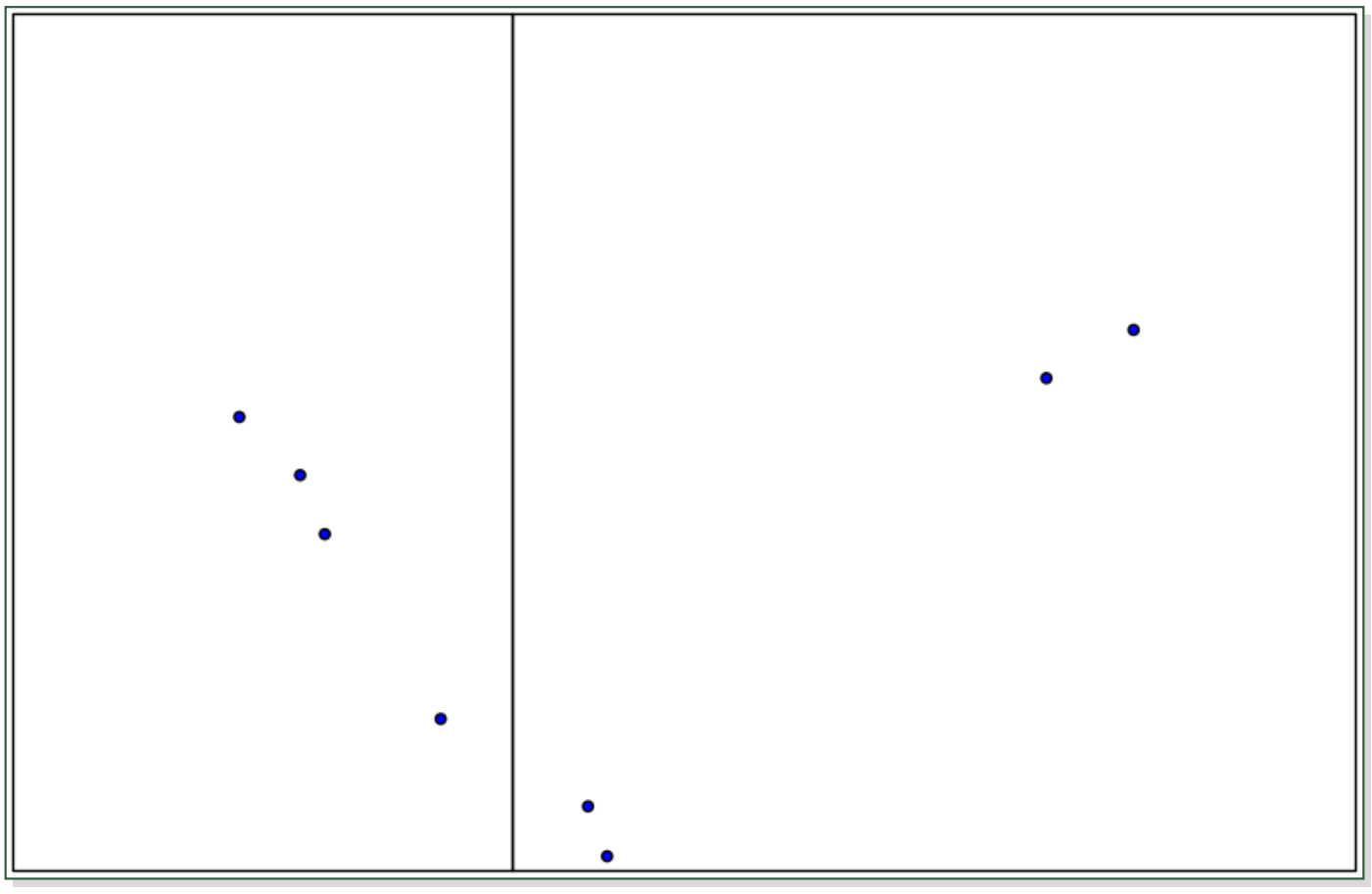
Intel i7@4.00GHz (1 core used)
GeForce Titan Z (1 device, 2880 cores, 6GB RAM)

Example II: Buffer k-d Trees

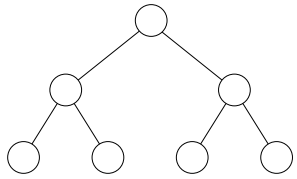


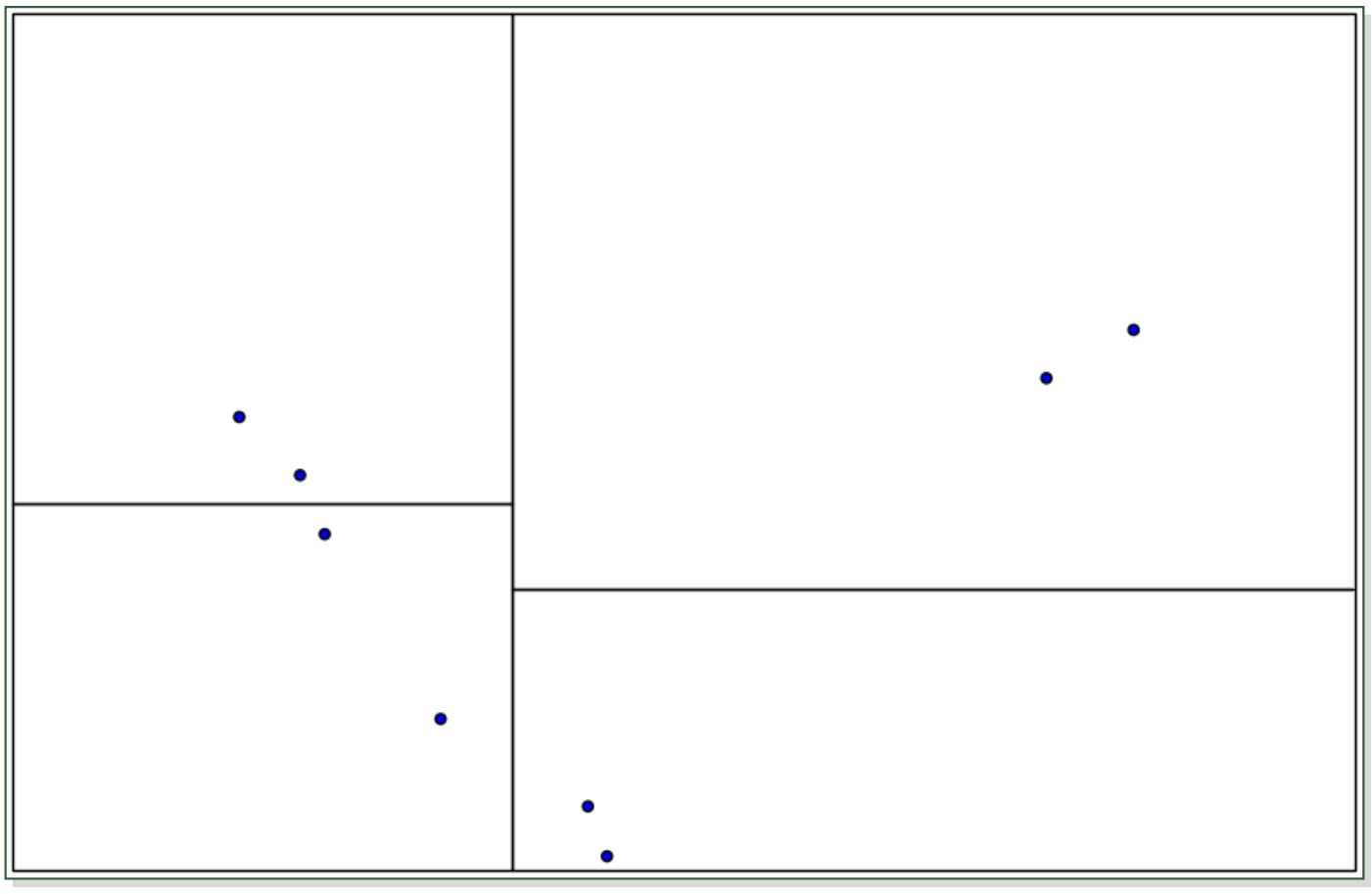
Example II: Buffer *k*-d Trees



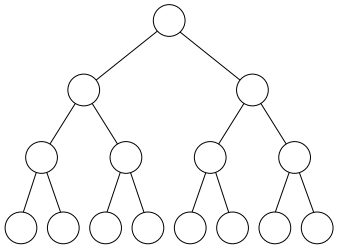


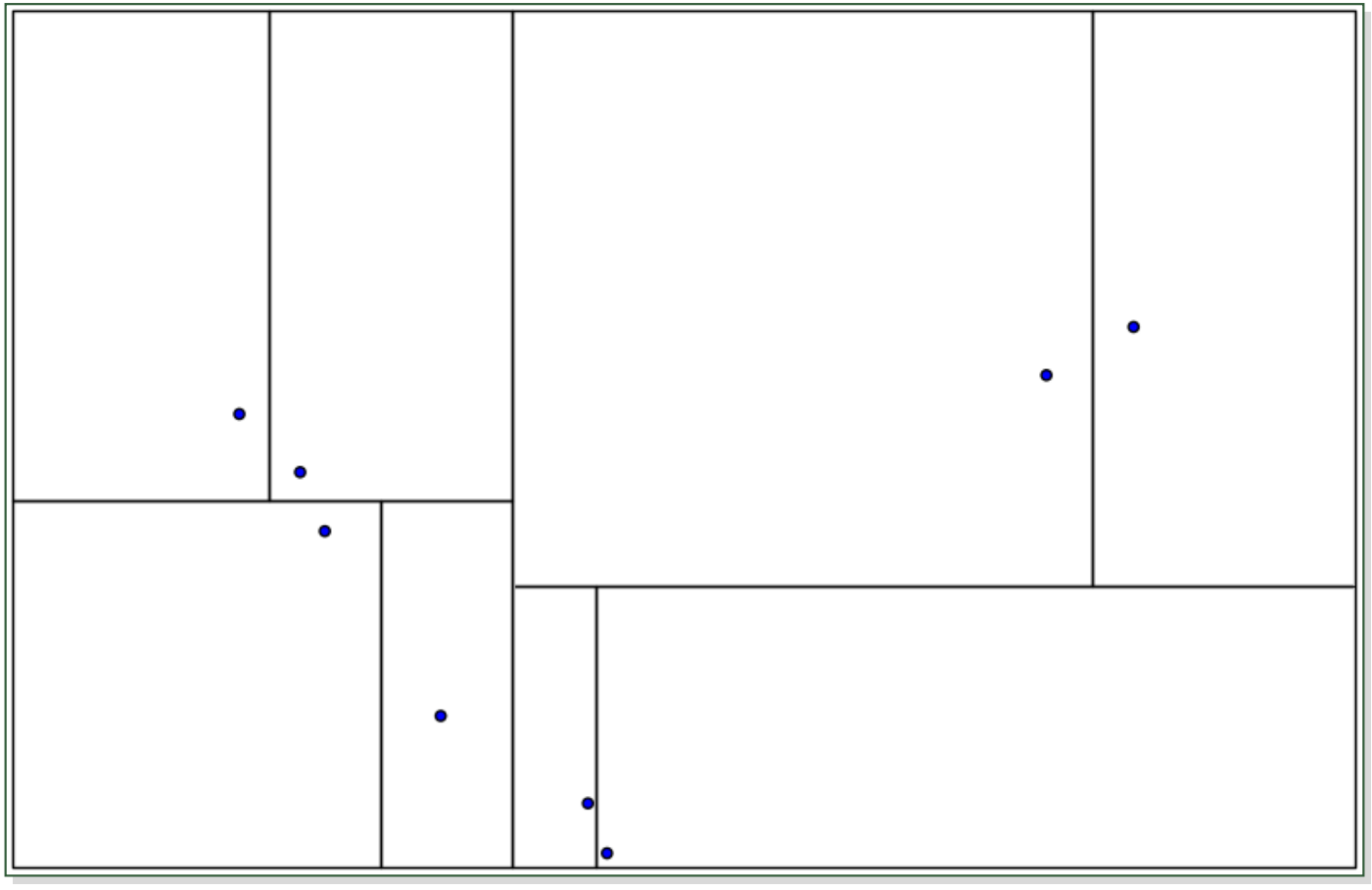
Example II: Buffer *k*-d Trees



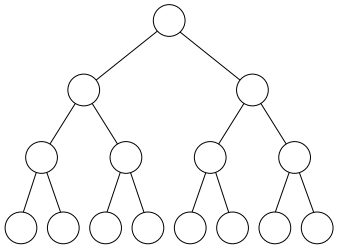


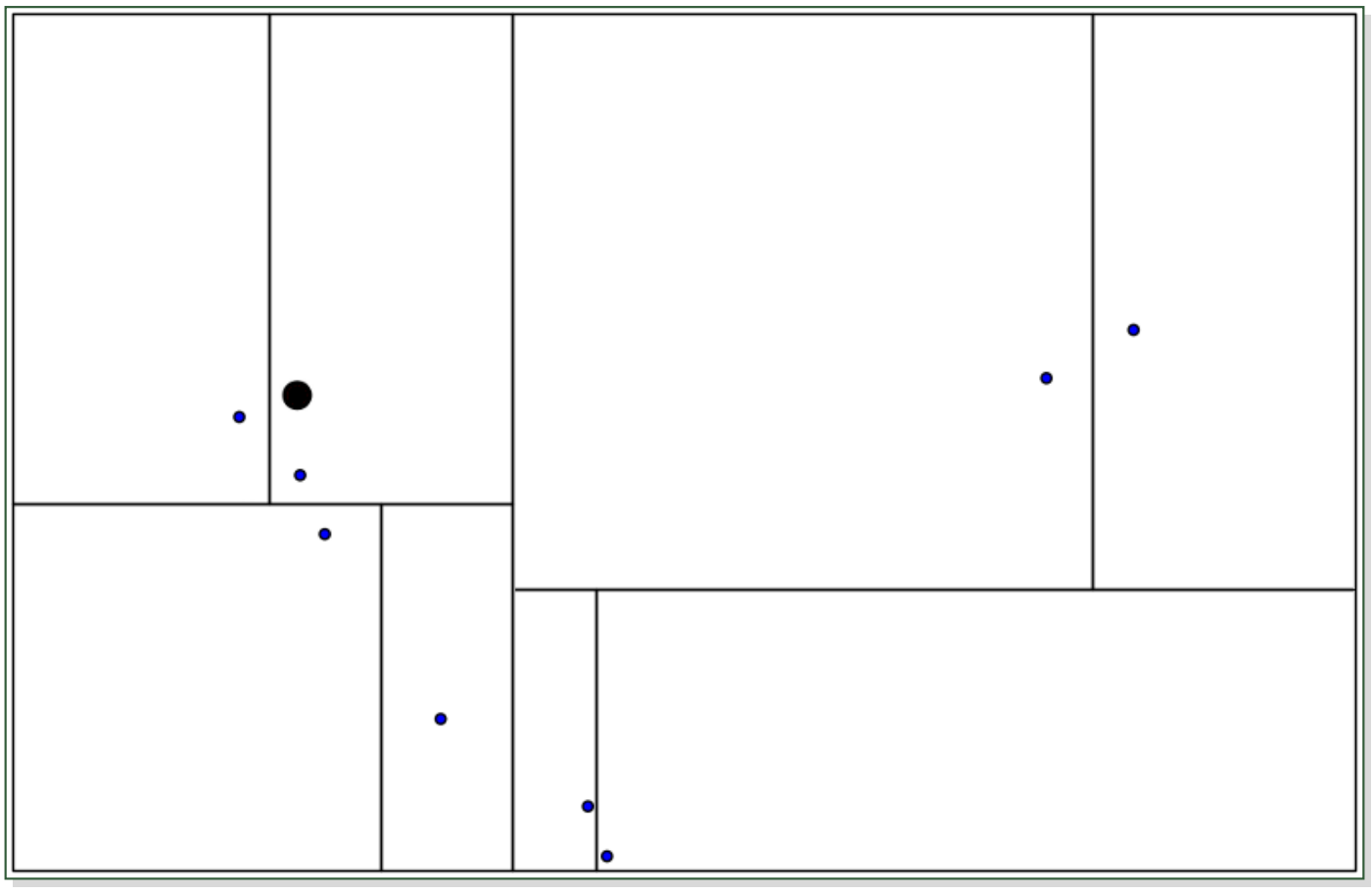
Example II: Buffer *k*-d Trees



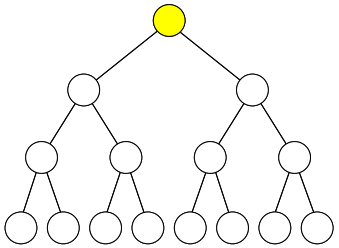


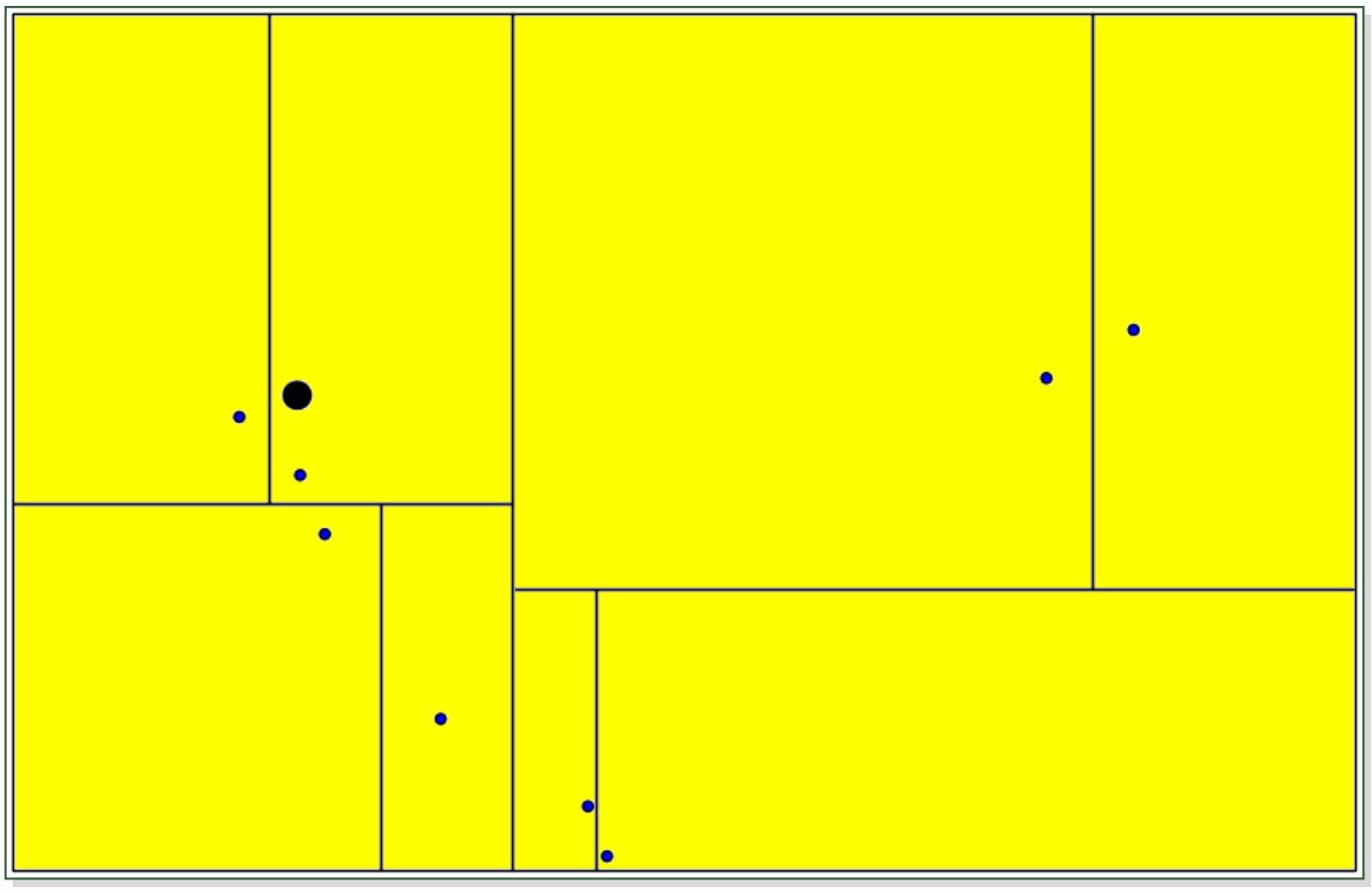
Example II: Buffer *k*-d Trees



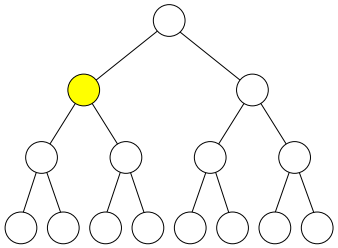


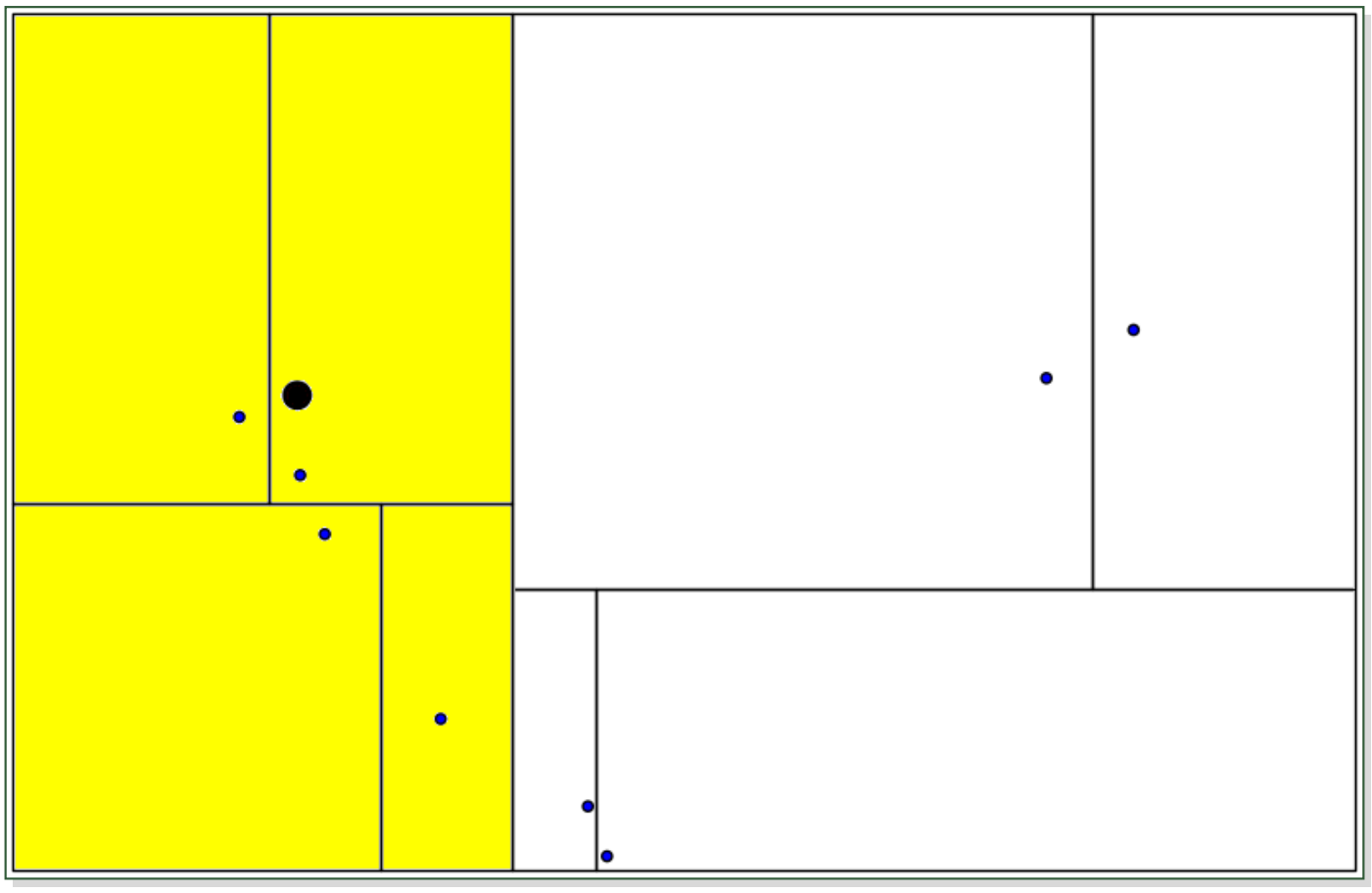
Example II: Buffer *k*-d Trees



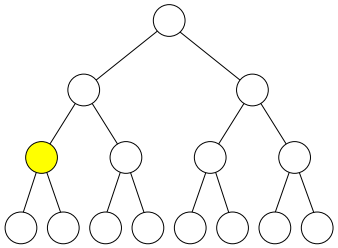


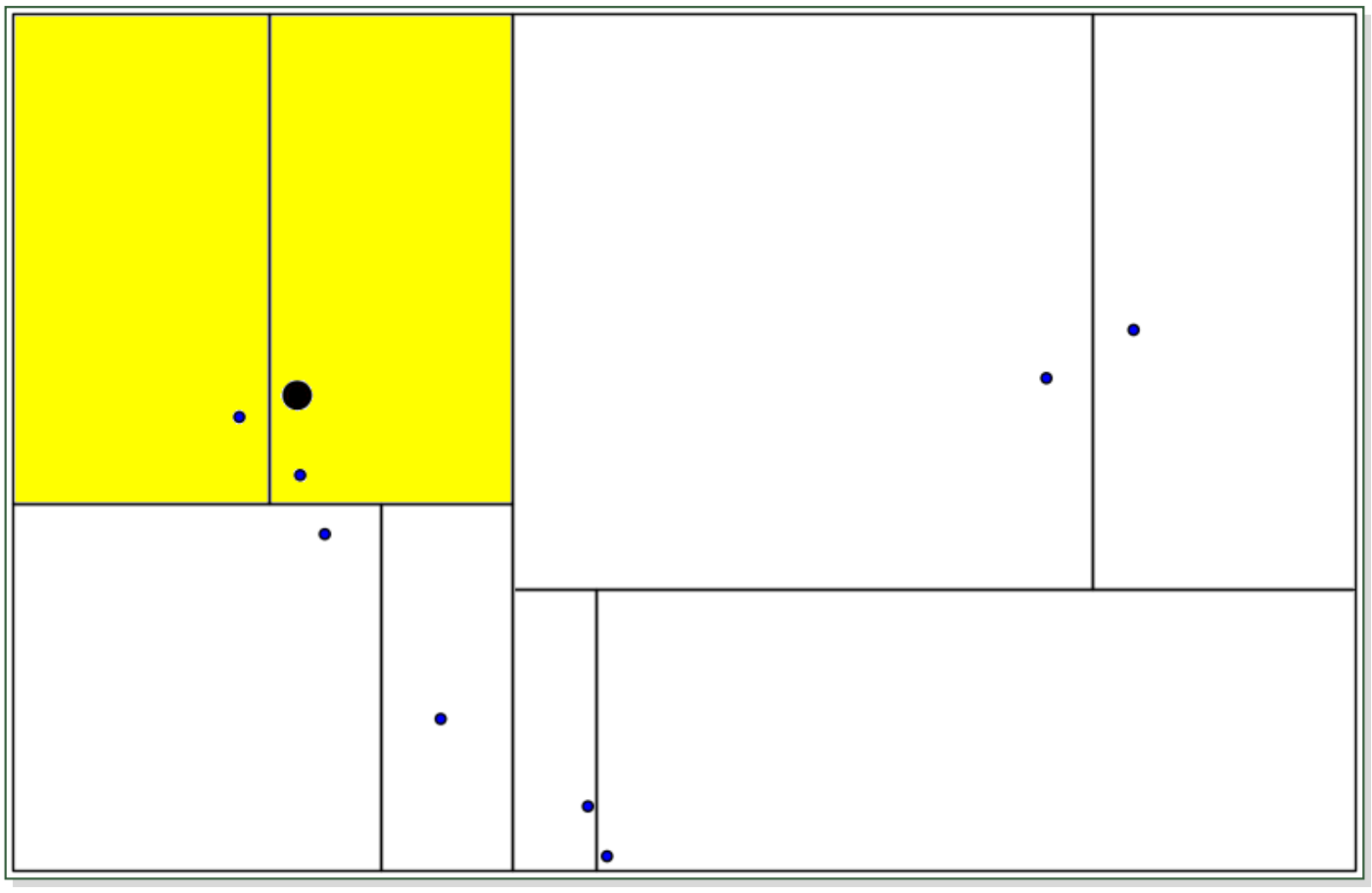
Example II: Buffer *k*-d Trees



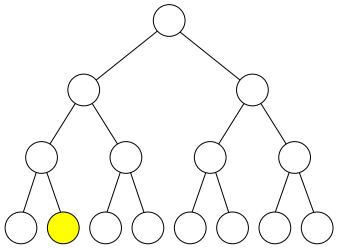


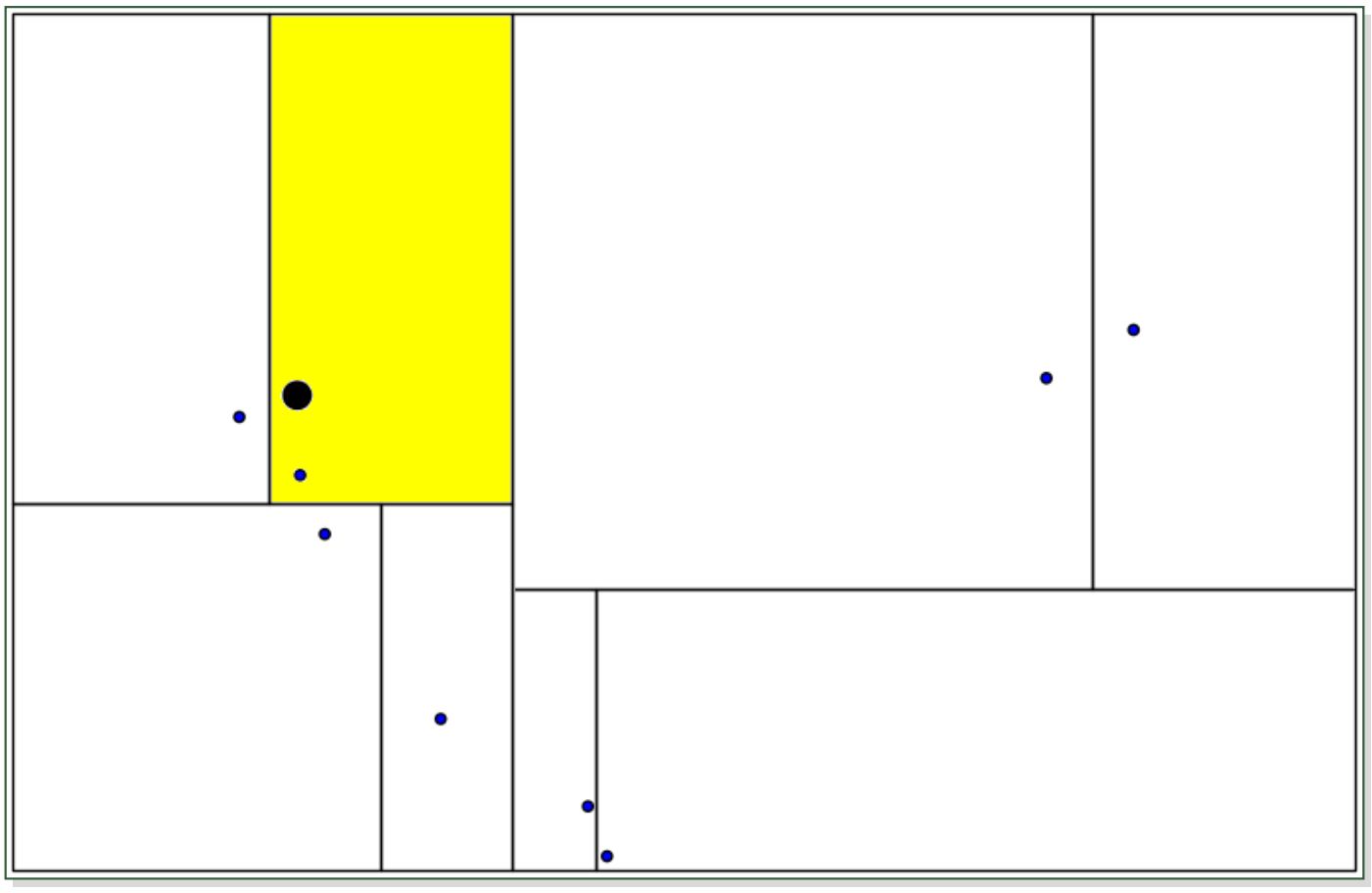
Example II: Buffer *k*-d Trees



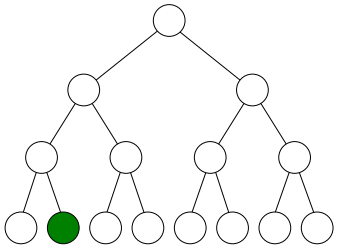


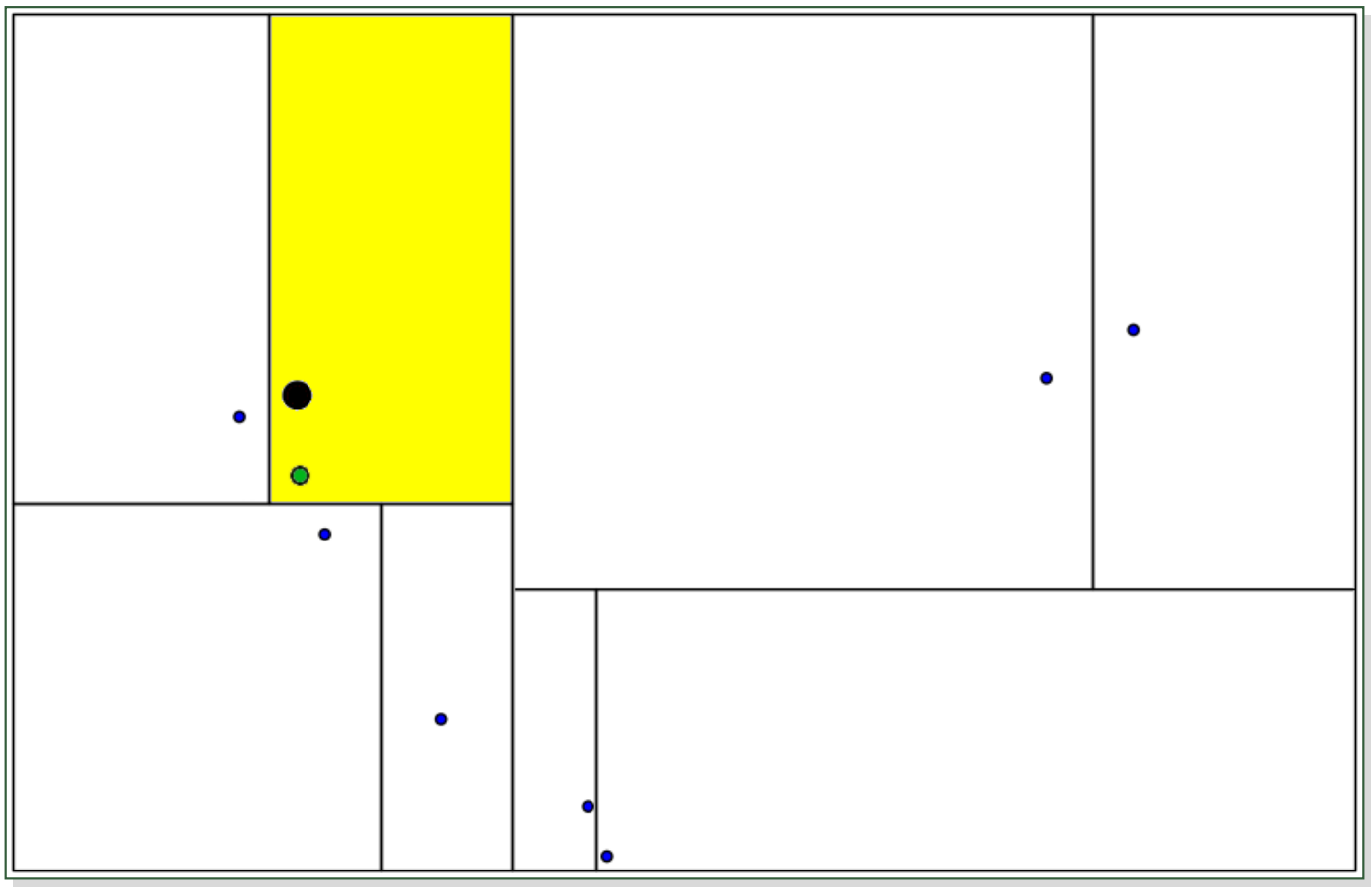
Example II: Buffer *k*-d Trees



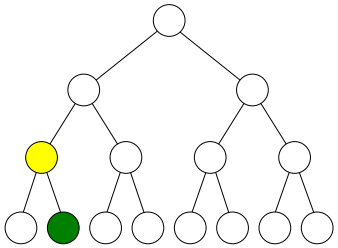


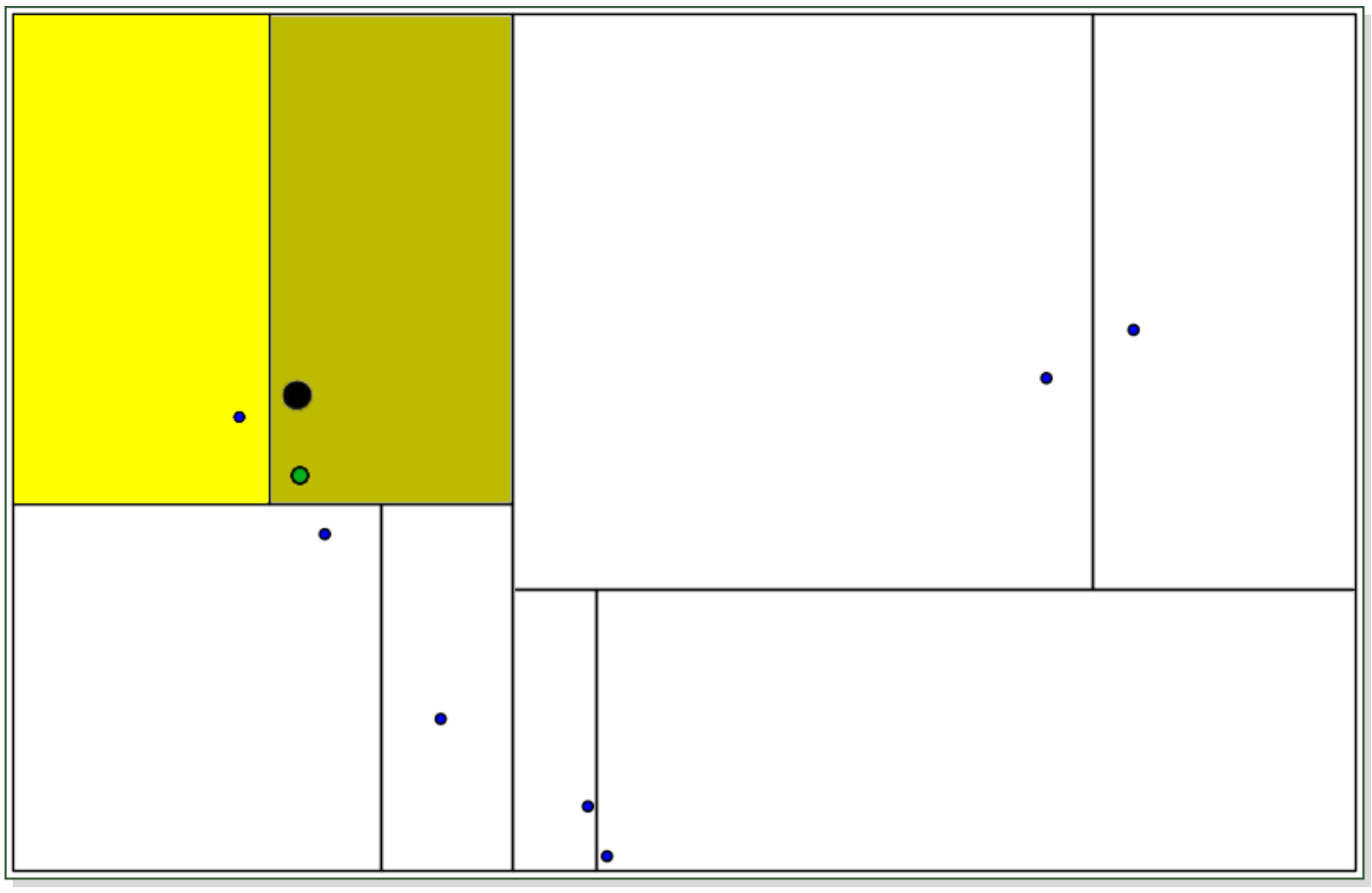
Example II: Buffer *k*-d Trees



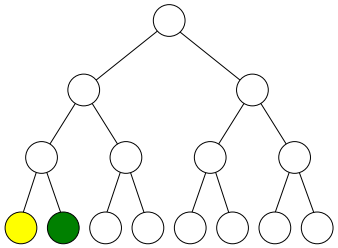


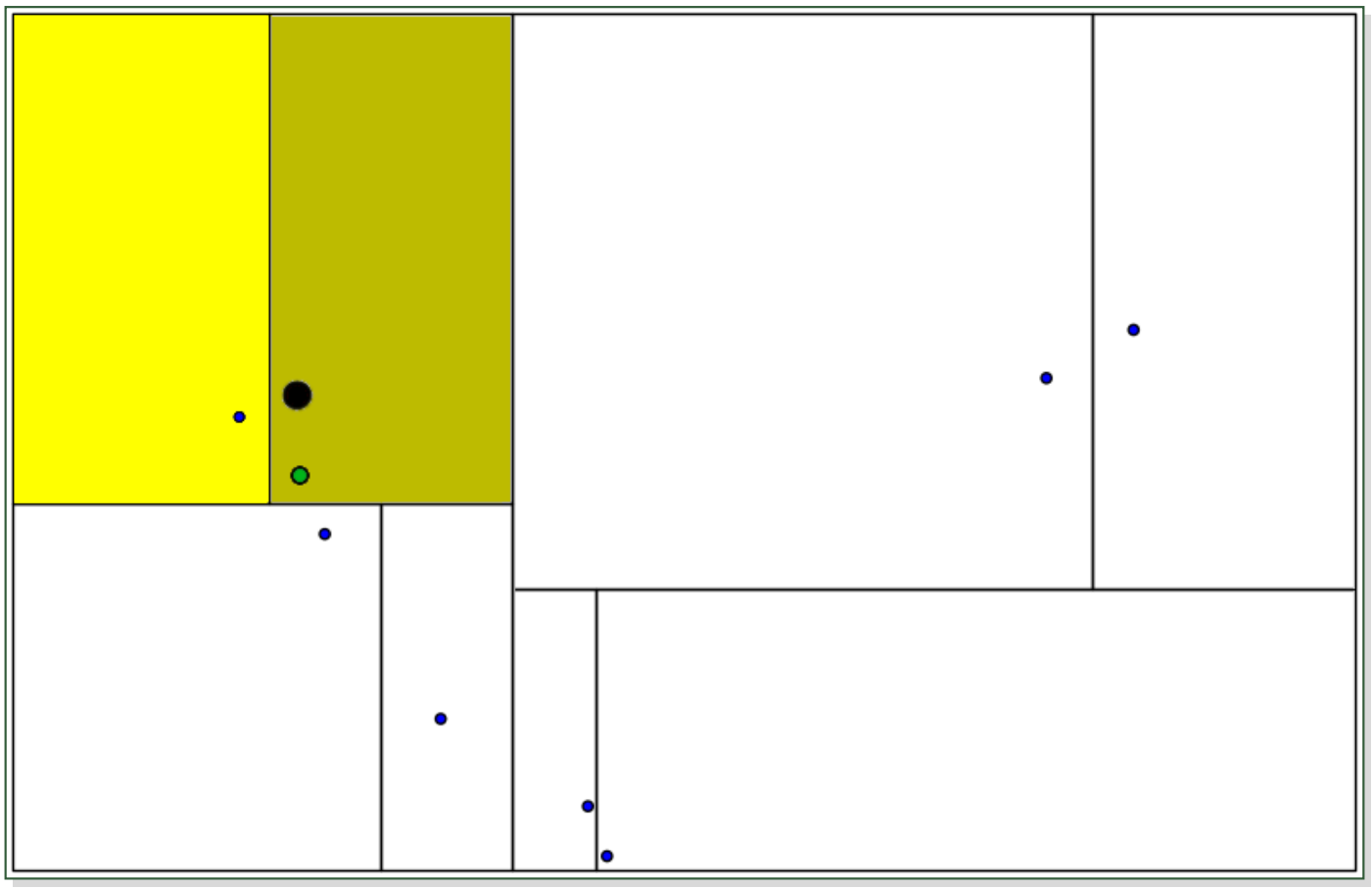
Example II: Buffer *k*-d Trees



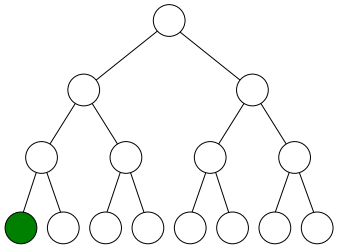


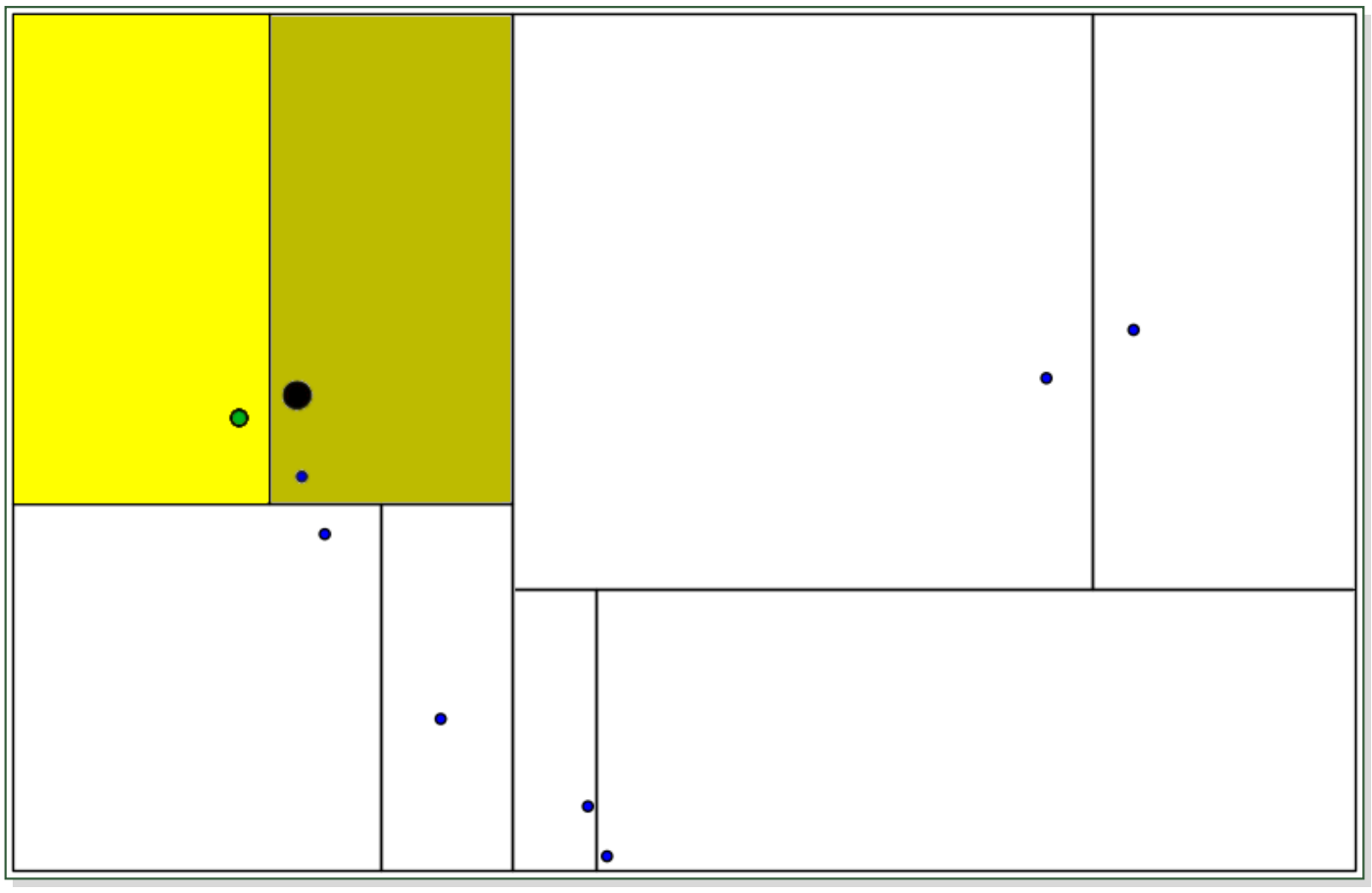
Example II: Buffer *k*-d Trees



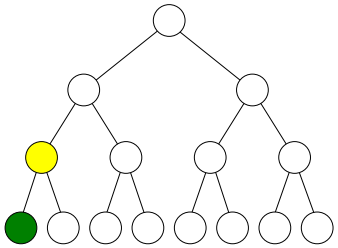


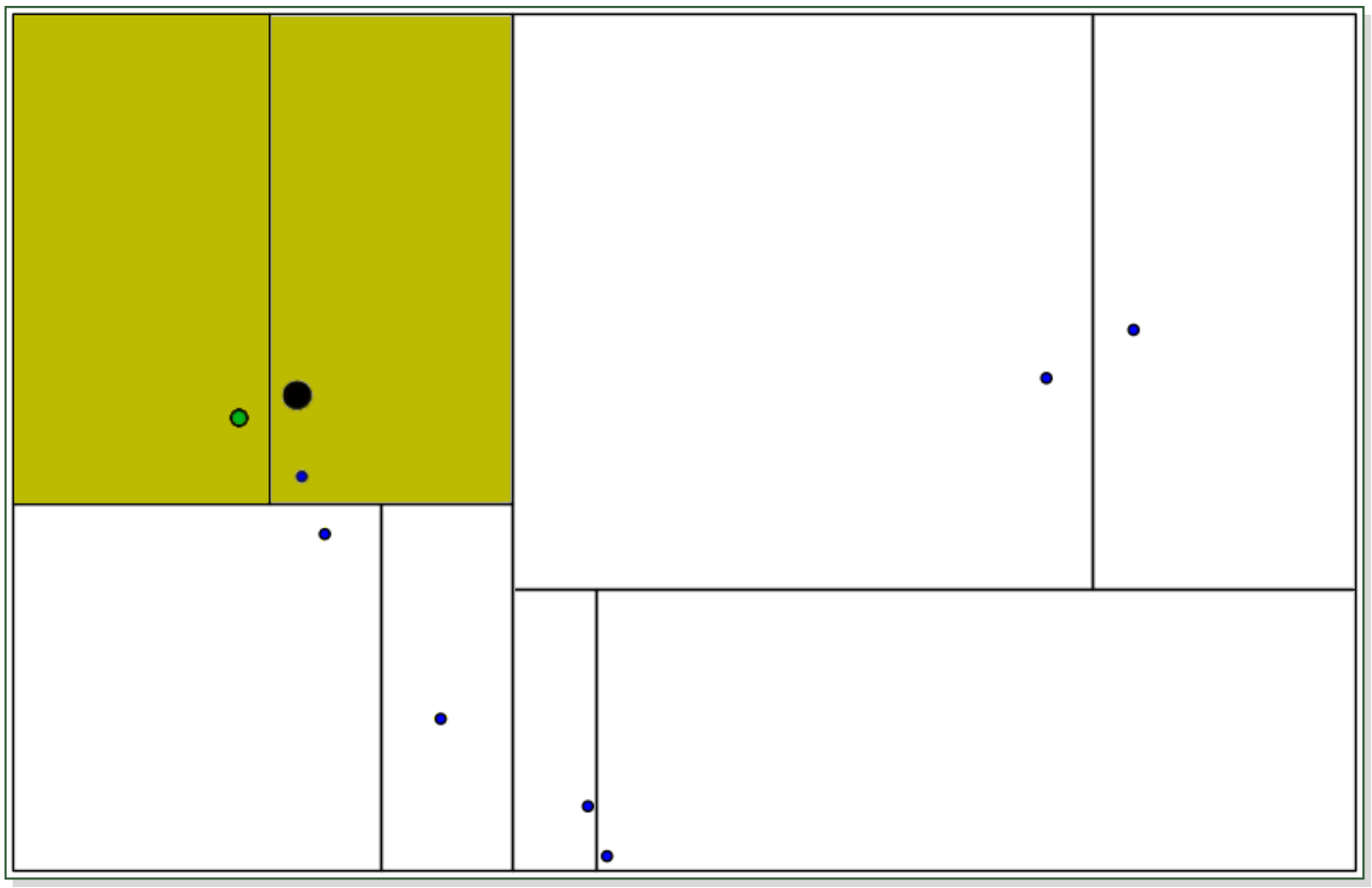
Example II: Buffer *k*-d Trees



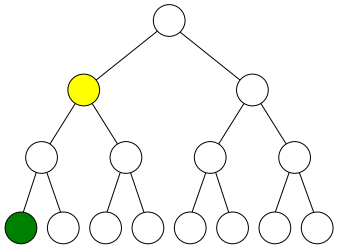


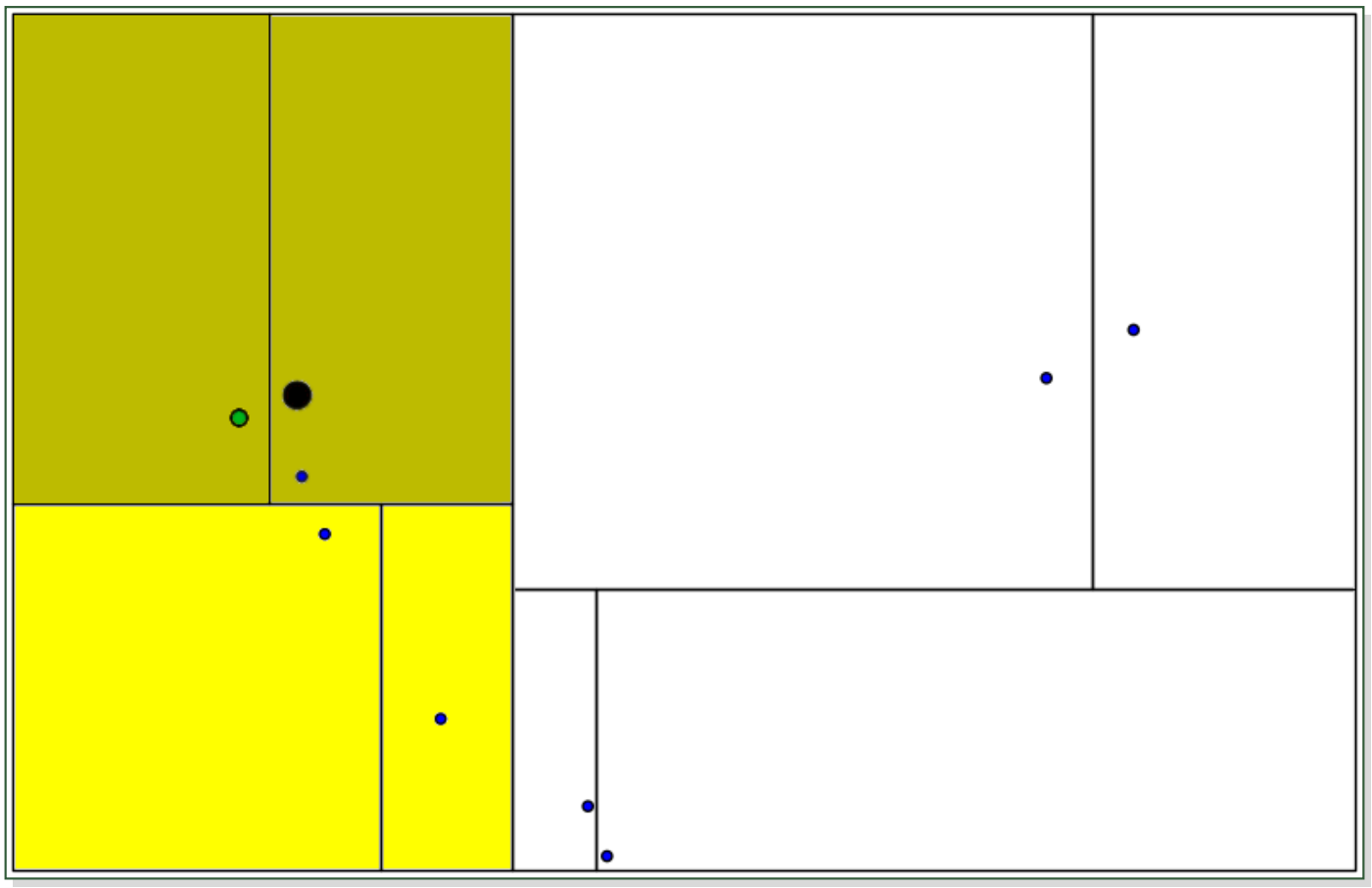
Example II: Buffer *k*-d Trees



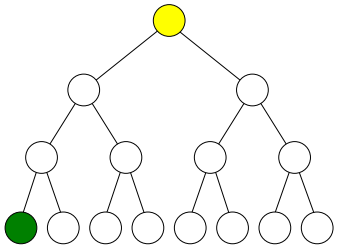


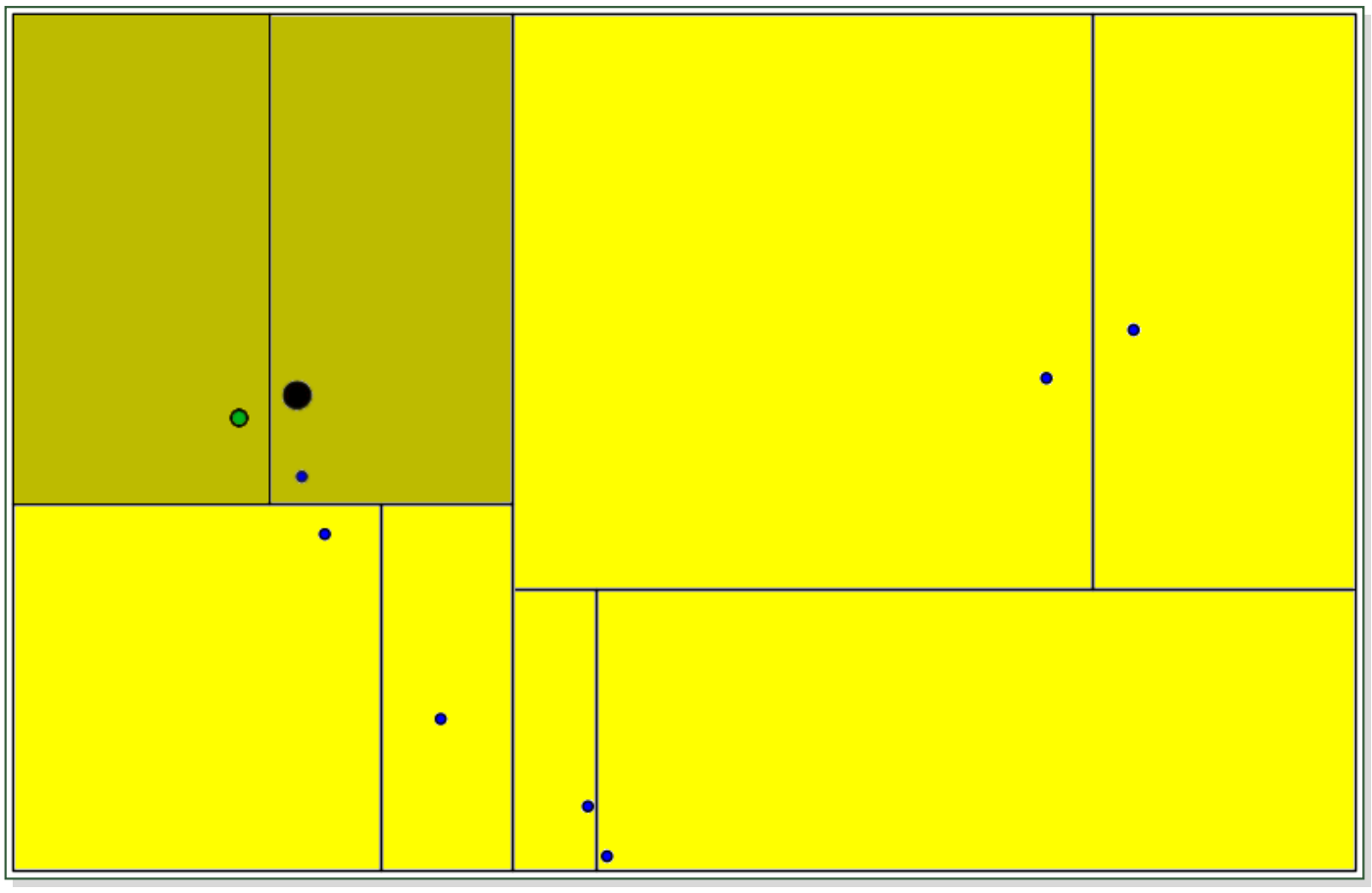
Example II: Buffer *k*-d Trees



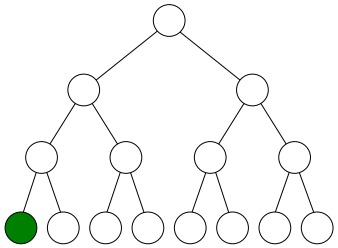


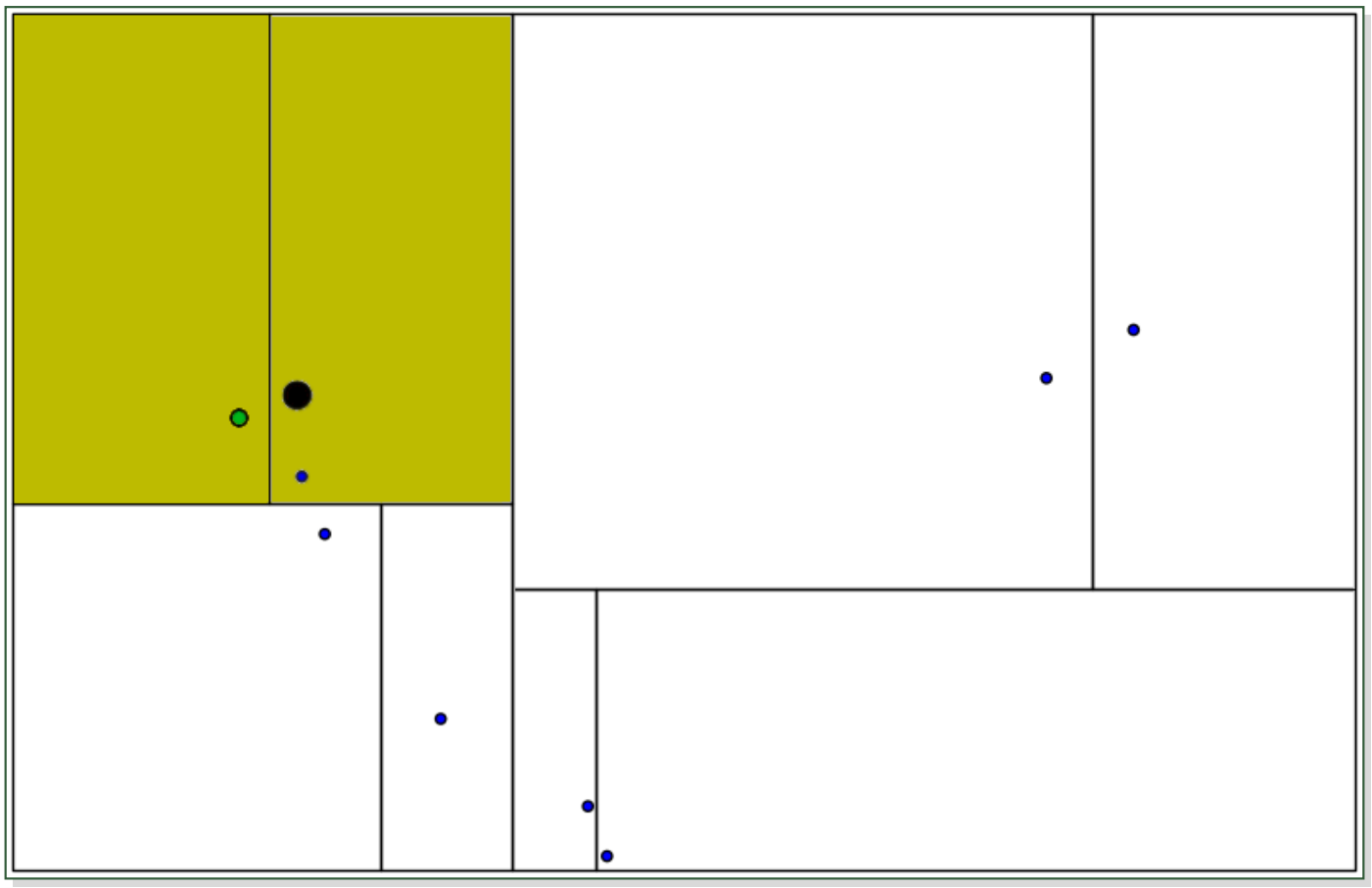
Example II: Buffer *k*-d Trees



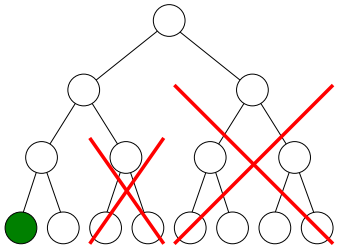


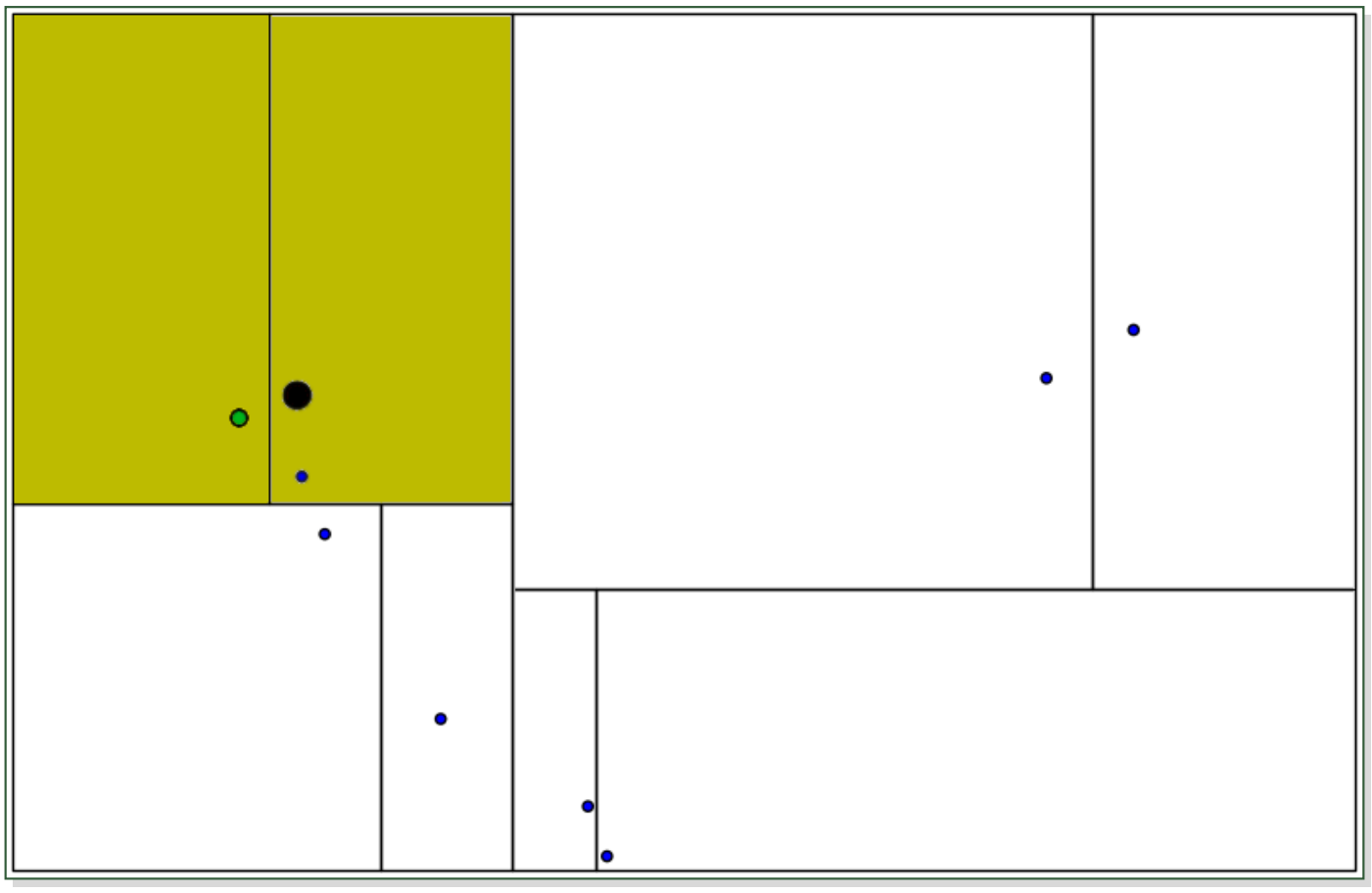
Example II: Buffer *k*-d Trees



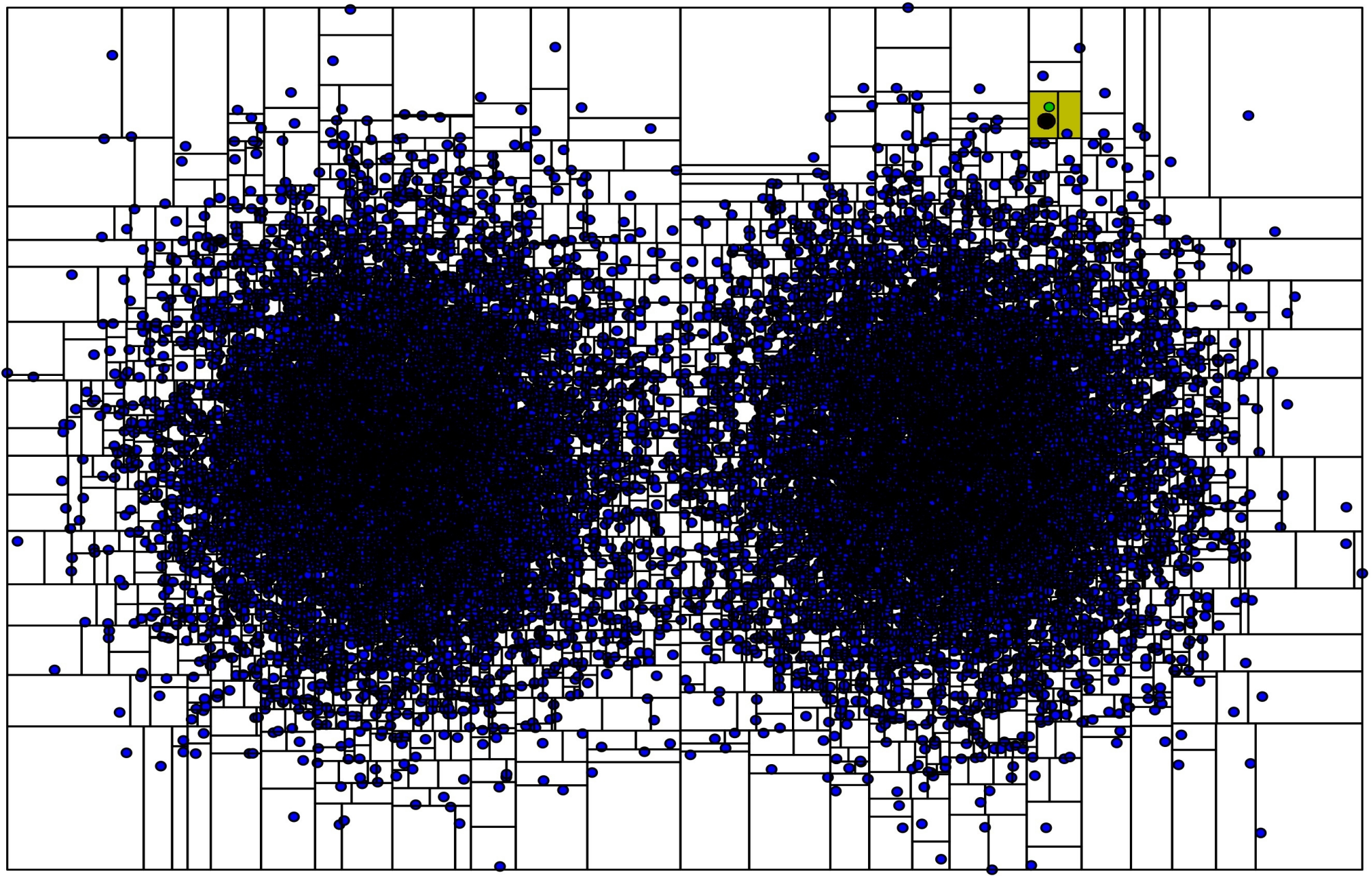


Example II: Buffer *k*-d Trees





Example II: Buffer *k*-d Trees



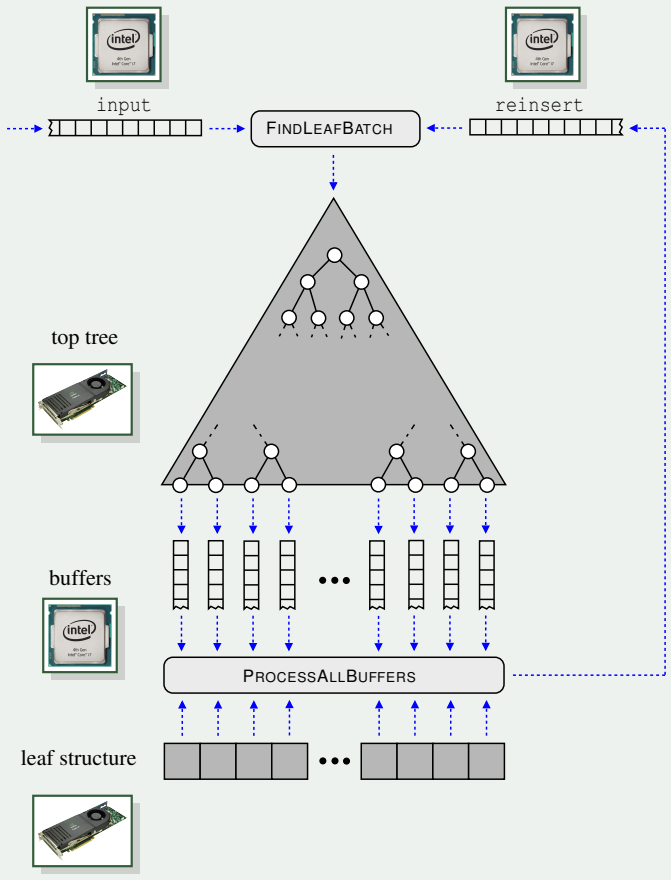
Example II: Buffer *k*-d Trees

Even with k-d trees, it can easily take hours or even days. This is not gonna be fast enough for future datasets/tasks!

Example II: Buffer *k*-d Trees

Buffer *k*-d Trees (Sketch)

1. Top tree: First levels of a standard *k*-d tree (i.e., its median values), laid out in memory in a pointer-less manner.
2. Leaf structure: Training patterns, sorted *in-place* during the construction of the top tree (w.r.t. the median values). Each block of the leaf structure corresponds to a leaf of the top tree.
3. Buffers: One buffer for each leaf of the top tree; each buffer can store a predefined number *B* of query indices.
4. Queues `input` and `reinsert`: Two (first-in-first-out) queues of size *m*.



Key Idea: Reorganize tree traversal and use GPU for compute-intensive parts!

Example II: Buffer K-D Trees

Intel i7@3.40GHz (4 cores, 8 hard. threads), GeForce GTX 770 (1536 cores, 4GB RAM)

	psf_colors ($d = 4$)	psf_mag ($d = 5$)	psf_model_mag ($d = 10$)	all_mag ($d = 15$)	all_colors ($d = 12$)	all ($d = 27$)
kdtree(cpu)	71 ($\times$ 5)	57 ($\times$ 5)	527 ($\times$15)	4616 ($\times$22)	16394 ($\times$34)	-
bufferkdtree(gpu)	14	12	36	210	478	1717

Table 1: Runtime comparison in seconds (speed-up in brackets)

Fabian Gieseke, Cosmin E. Oancea, Ashish Mahabal, Christian Igel, and Tom Heskes. *Bigger Buffer k-d Trees on Multi-Many-Core Systems*, BDL, 2016.

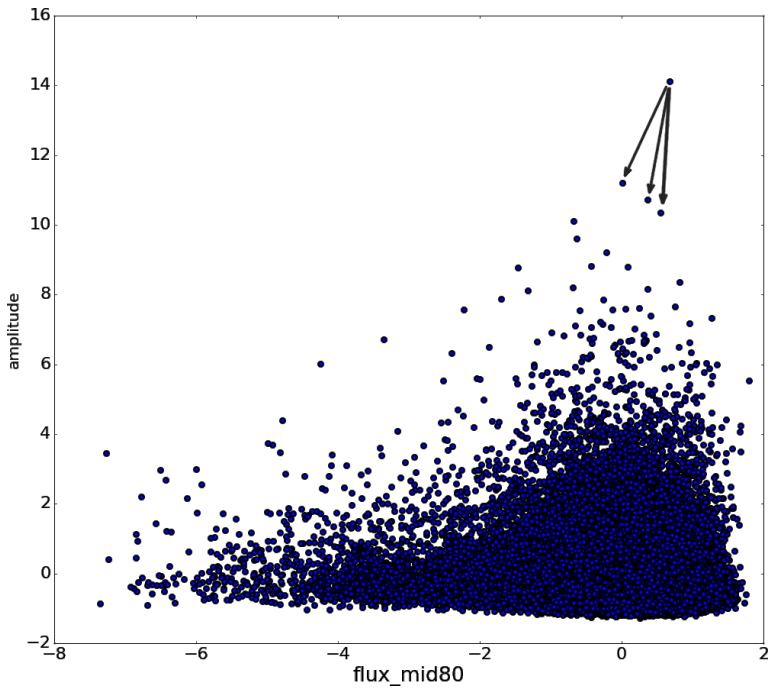
Fabian Gieseke, Justin Heinermann, Cosmin Oancea, and Christian Igel. Buffer k-d Trees: Processing Massive Nearest Neighbor Queries on GPUs, ICML, 2014.

Example II: Buffer K-D Trees

Intel i7@3.40GHz (4 cores, 8 hard. threads), GeForce GTX 770 (1536 cores, 4GB RAM)

	psf_colors ($d = 4$)	psf_mag ($d = 5$)	psf_model_mag ($d = 10$)	all_mag ($d = 15$)	all_colors ($d = 12$)	all ($d = 27$)
kdtree(cpu)	71 (× 5)	57 (× 5)	527 (×15)	4616 (×22)	16394 (×34)	-
bufferkdtree(gpu)	14	12	36	210	478	1717

Table 1: Runtime comparison in seconds (speed-up in brackets)



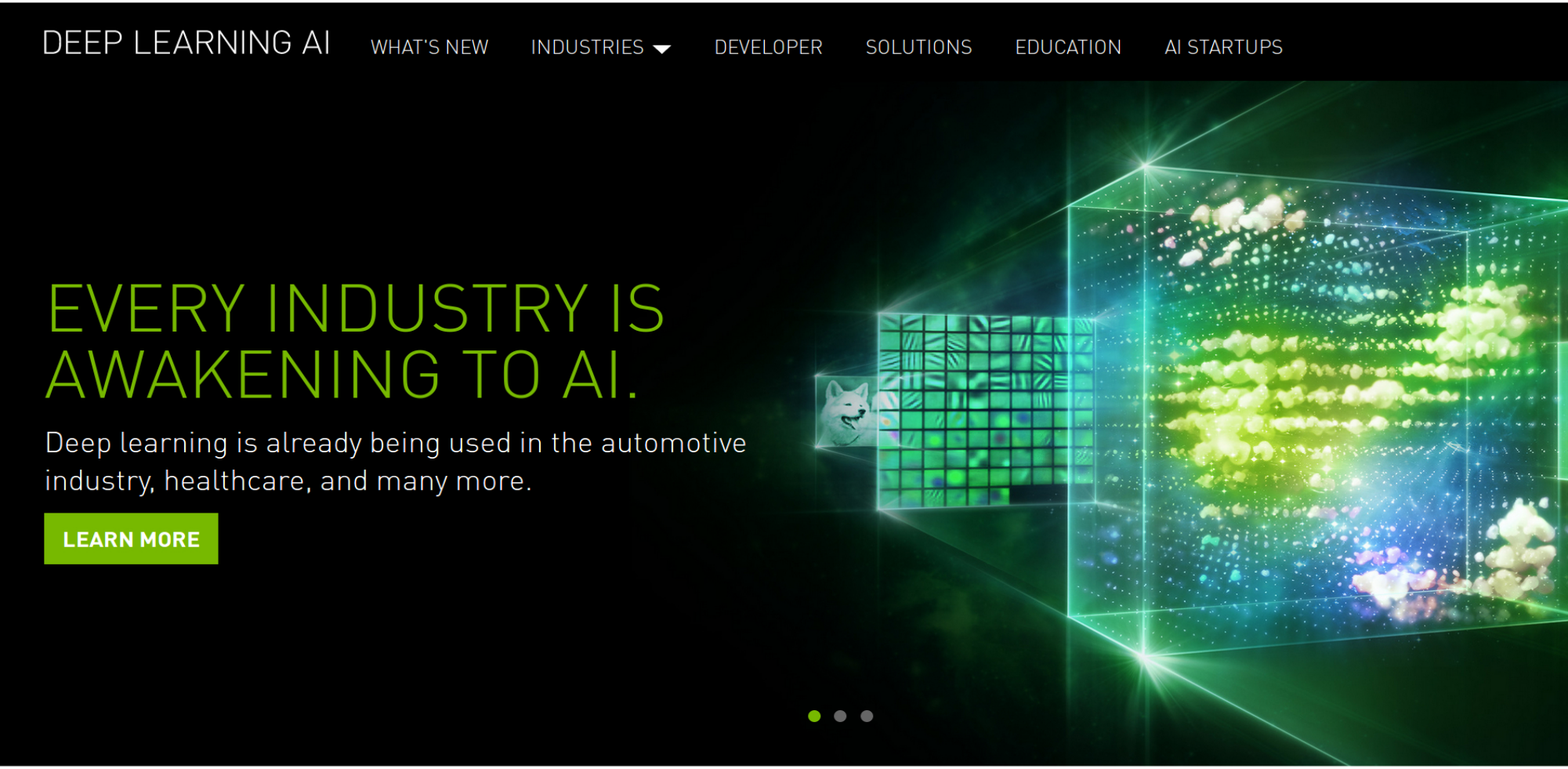
Fabian Gieseke, Cosmin E. Oancea, Ashish Mahabal, Christian Igel, and Tom Heskes. *Bigger Buffer k-d Trees on Multi-Many-Core Systems*, BDL, 2016.

Fabian Gieseke, Justin Heinermann, Cosmin Oancea, and Christian Igel. Buffer k-d Trees: Processing Massive Nearest Neighbor Queries on GPUs, ICML, 2014.

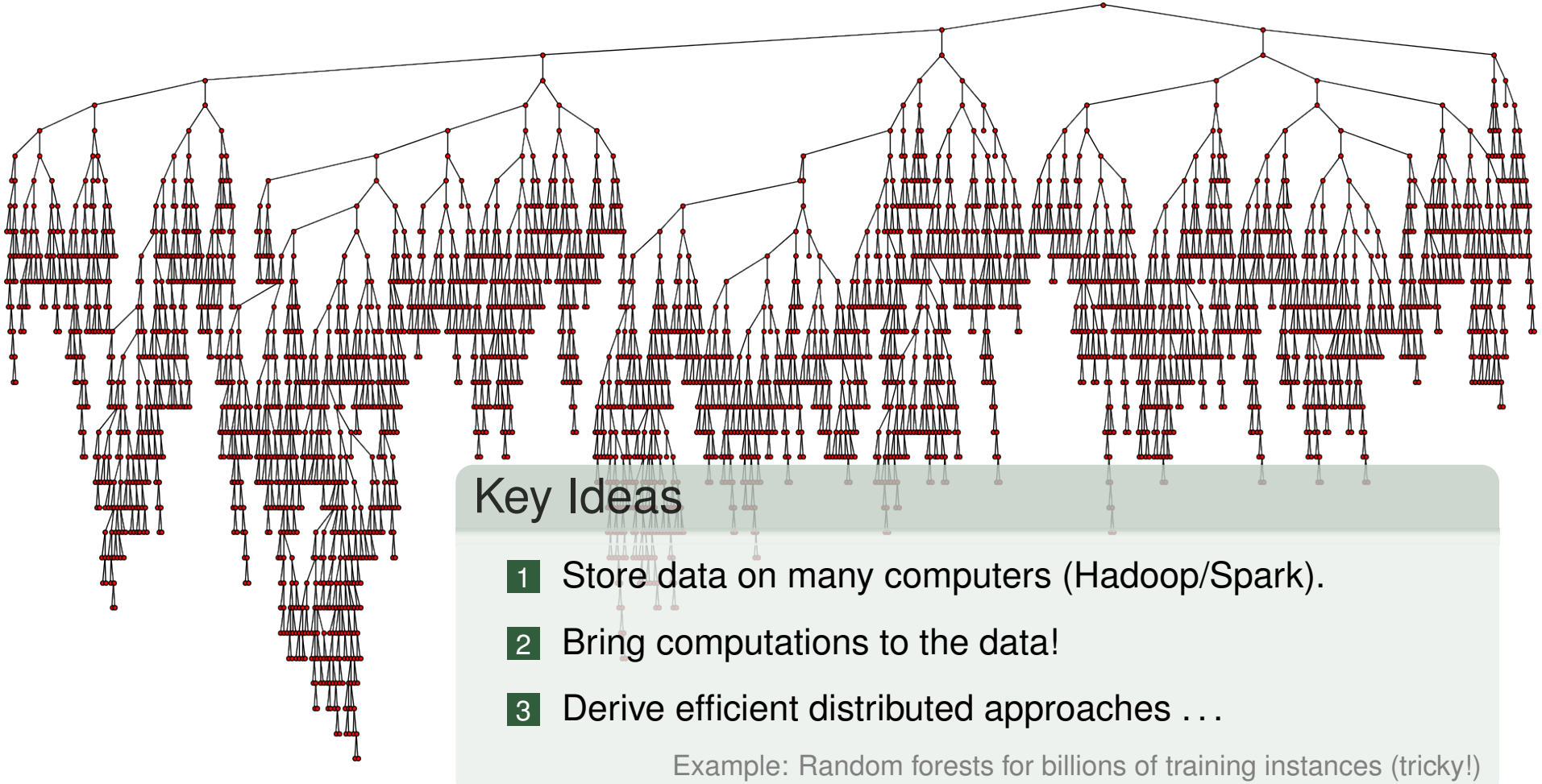
Now Possible!
In hours instead of days . . .

Example III: Deep Learning

https://www.nvidia.com/en-us/deep-learning-ai/



Example IV: Distributed Computing



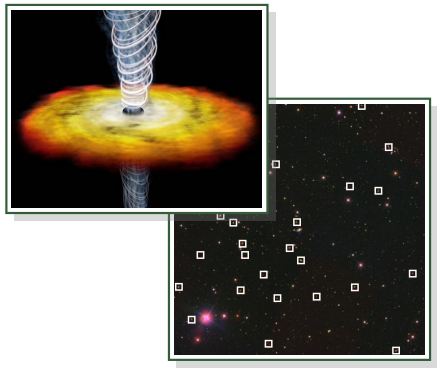
Key Ideas

1. Store data on many computers (Hadoop/Spark).
2. Bring computations to the data!
3. Derive efficient distributed approaches . . .

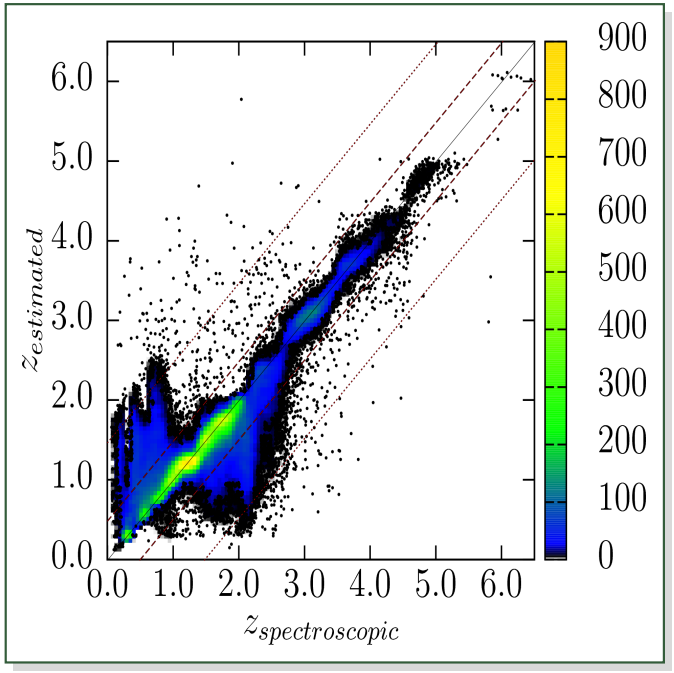
Example: Random forests for billions of training instances (tricky!)

Outline

Photometric k-NN Regression → Quasars



$$f(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$$

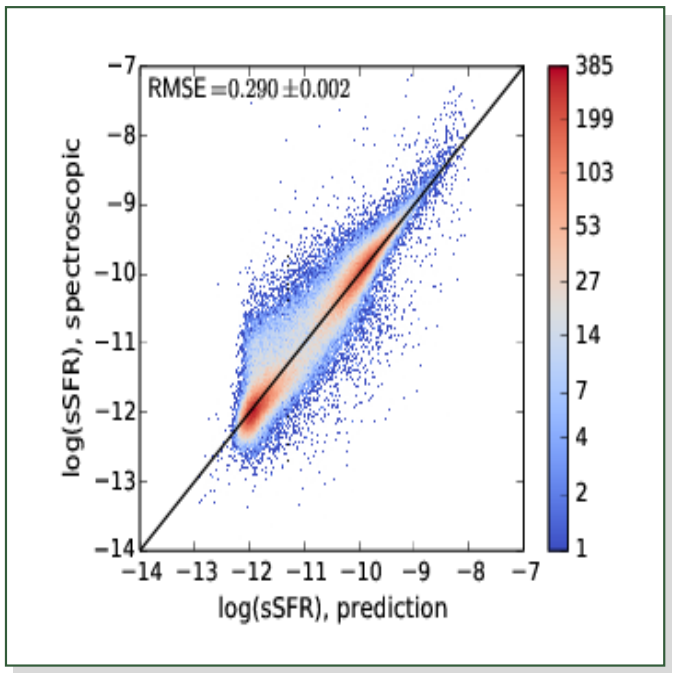


Nearest Neighbor Regression

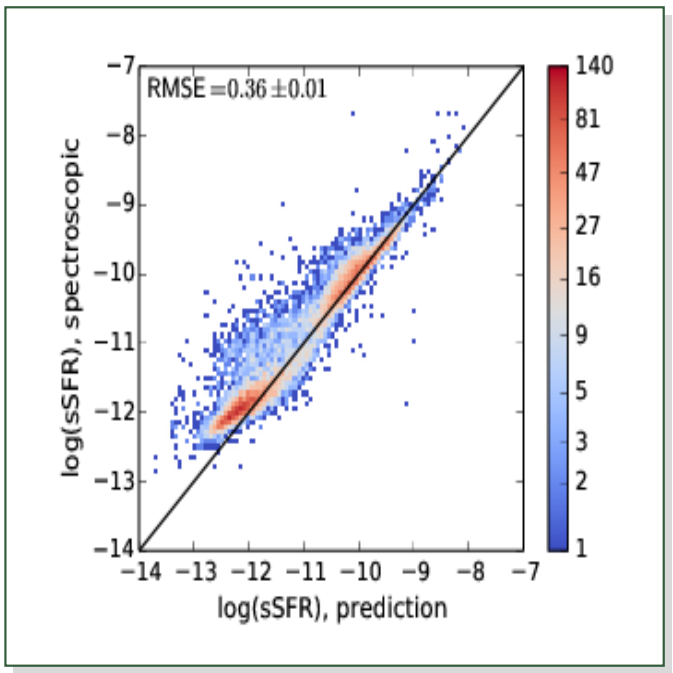
Polsterer, Zinn, Gieseke. *Finding New High-Redshift Quasars by Asking the Neighbours*, MNRAS, 2013.

Photometric k-NN Regression → SSFR

$$f(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$$



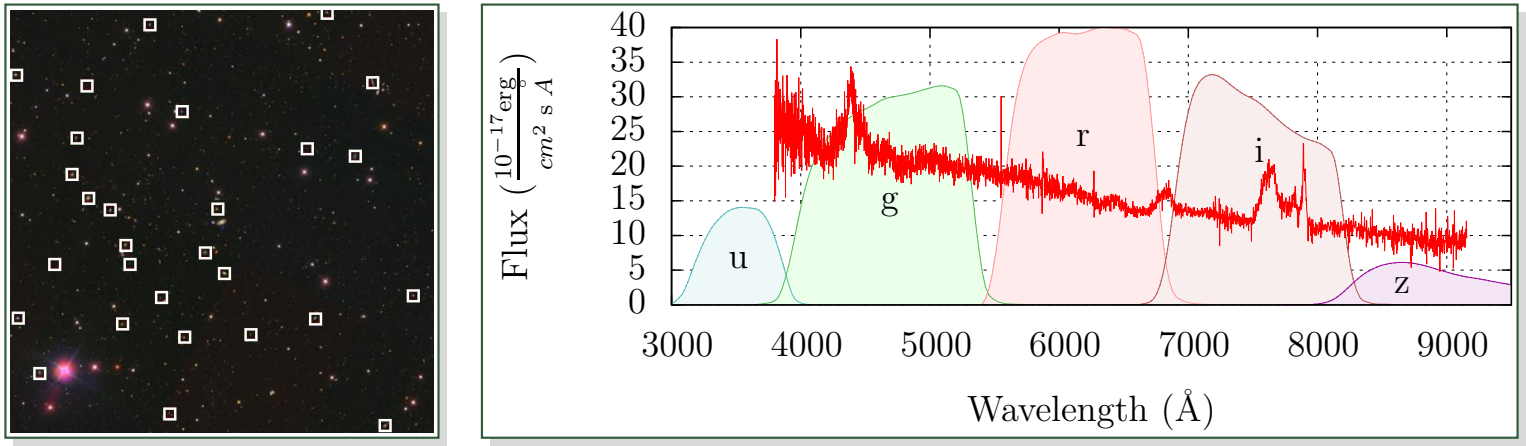
Nearest Neighbor Regression



Physical Model

Stensbo-Smidt, Gieseke, Zirm, Pedersen, Igel. *Sacrificing information for the greater good: how to select photometric bands for optimal accuracy*, MNRAS, 2017.

Special Problem: Sample Selection Bias

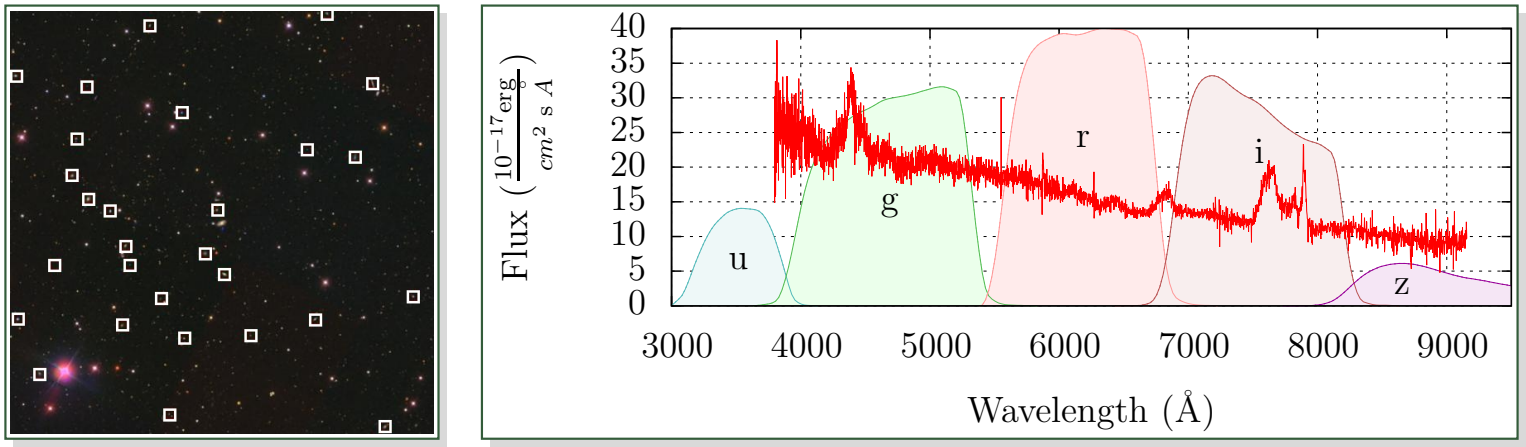


Photometric Target Selection

- Training data: All photometric objects with spectra
- Test data: All photometric objects!

Spectroscopic follow-up observations are made of potentially interesting objects.
This leads to a heavy sample selection bias!

Special Problem: Sample Selection Bias



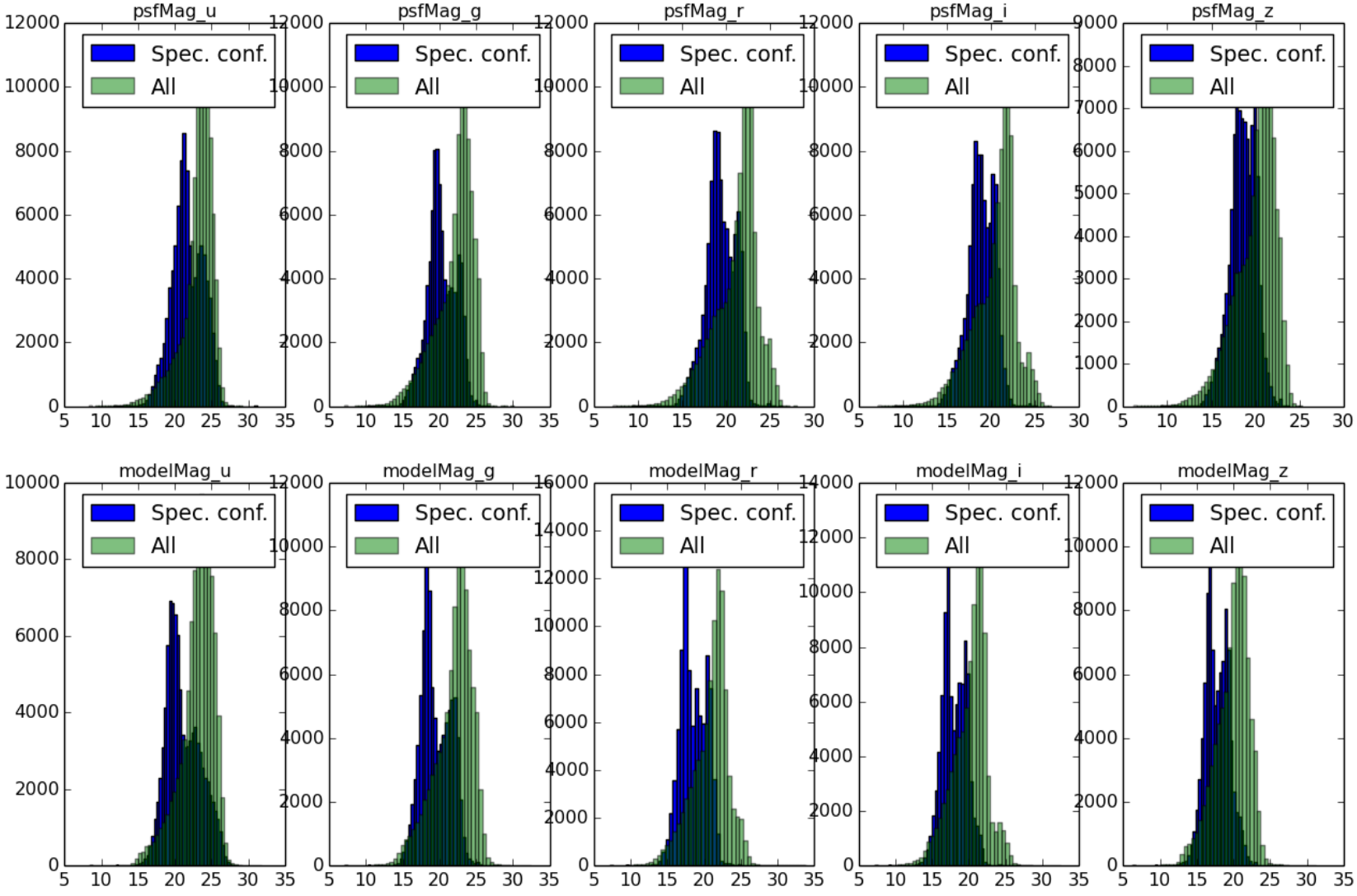
Photometric Target Selection

- Training data: All photometric objects with spectra
- Test data: All photometric objects!

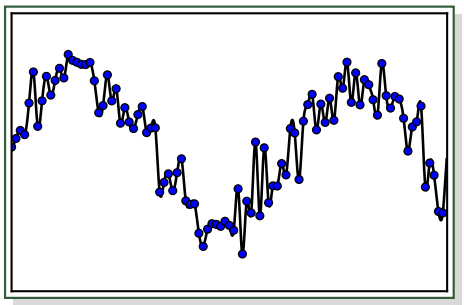
Spectroscopic follow-up observations are made of potentially interesting objects.
This leads to a heavy sample selection bias!

What can we say about the true performance of a model?

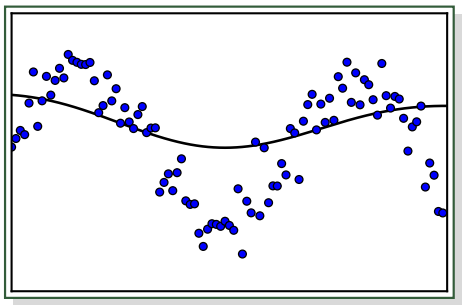
Sample Selection Bias?



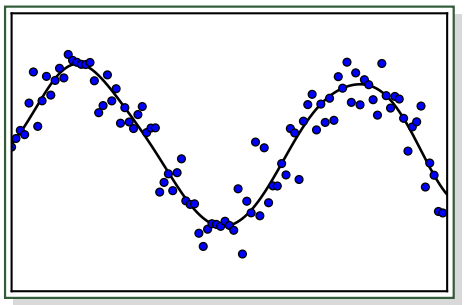
Adaptation of Regression Models



$\lambda =$ small



$\lambda =$ large

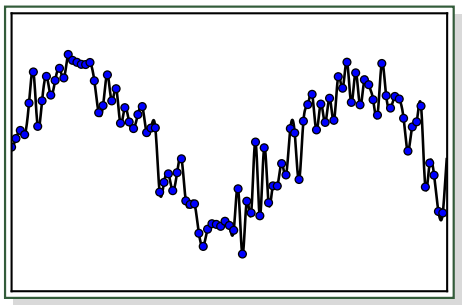


$\lambda =$ middle

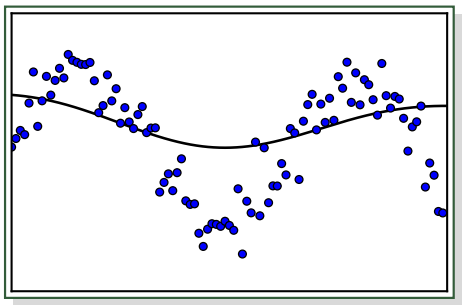
$$\underset{f \in \mathcal{H},\, b \in \mathbb{R}}{\text{minimize}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(\mathbf{x}_i) + b) +}_{\text{Small loss on training data}} \underbrace{\lambda \|f\|^2}_{\text{Not too complex}}$$

Kremer, Gieseke, Pedersen, Igel. *Nearest Neighbor Density Ratio Estimation for Large-Scale Applications in Astronomy*, Astronomy and Computing, 2015.
Beck, Lin, Ishida, Gieseke, Souza, Costa-Duarte, Hattab, Krone-Martins. *On the realistic validation of photometric redshifts*, MNRAS, 2017.

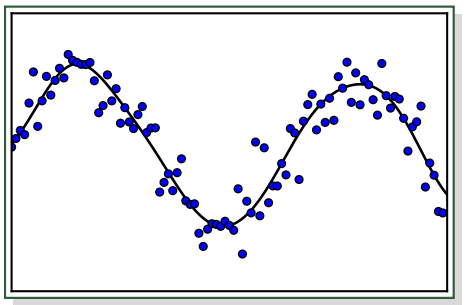
Adaptation of Regression Models



$\lambda =$ small



$\lambda =$ large



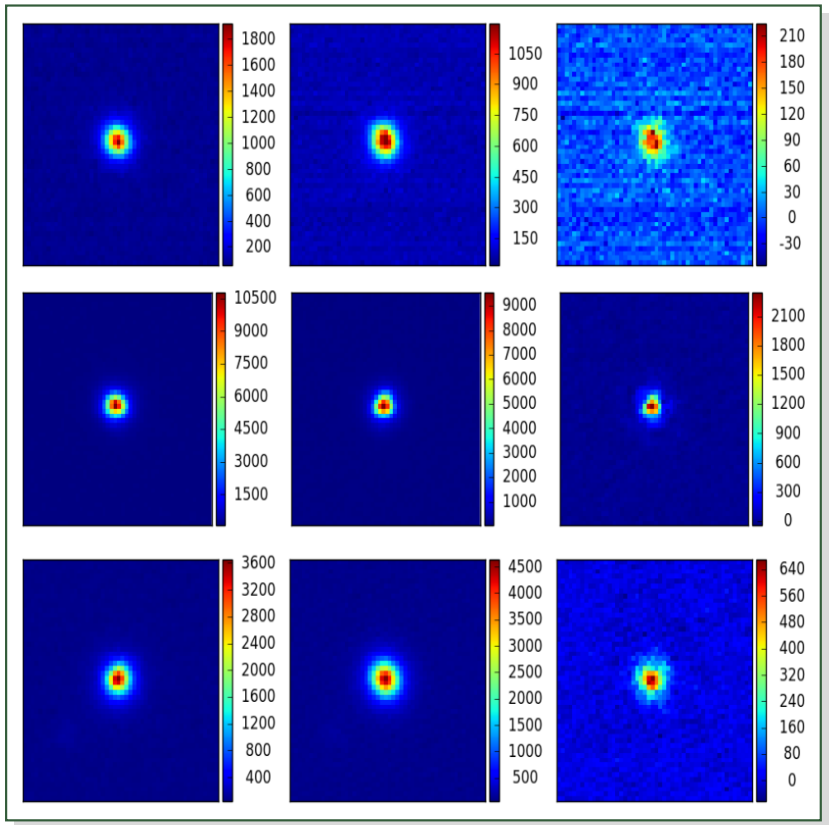
$\lambda =$ middle

$$\underset{f \in \mathcal{H},\, b \in \mathbb{R}}{\text{minimize}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \beta_i \mathcal{L}(y_i, f(\mathbf{x}_i) + b)}_{\text{Small loss on test data}} + \underbrace{\lambda \|f\|^2}_{\text{Not too complex}}$$

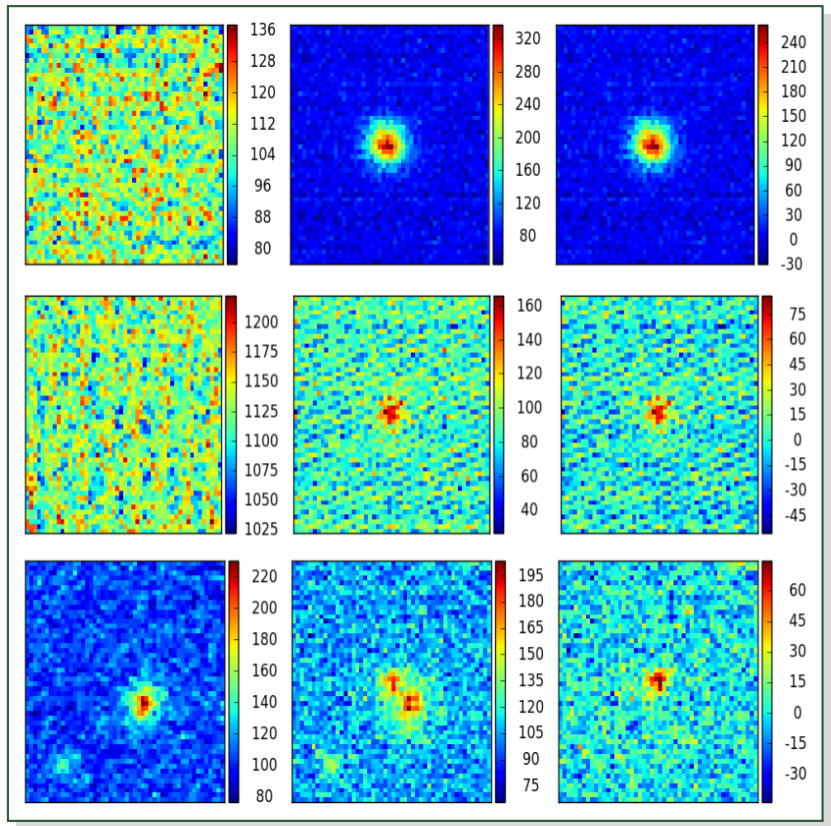
- Introduce reweighting coefficients $\beta_1, \ldots, \beta_n \in \mathbb{R}$
- Estimate: $\beta_i = \frac{P_{test}(\mathbf{x}_i)}{P_{train}(\mathbf{x}_i)}$

Kremer, Gieseke, Pedersen, Igel. *Nearest Neighbor Density Ratio Estimation for Large-Scale Applications in Astronomy*, Astronomy and Computing, 2015.
Beck, Lin, Ishida, Gieseke, Souza, Costa-Duarte, Hattab, Krone-Martins. *On the realistic validation of photometric redshifts*, MNRAS, 2017.

Transient Detection



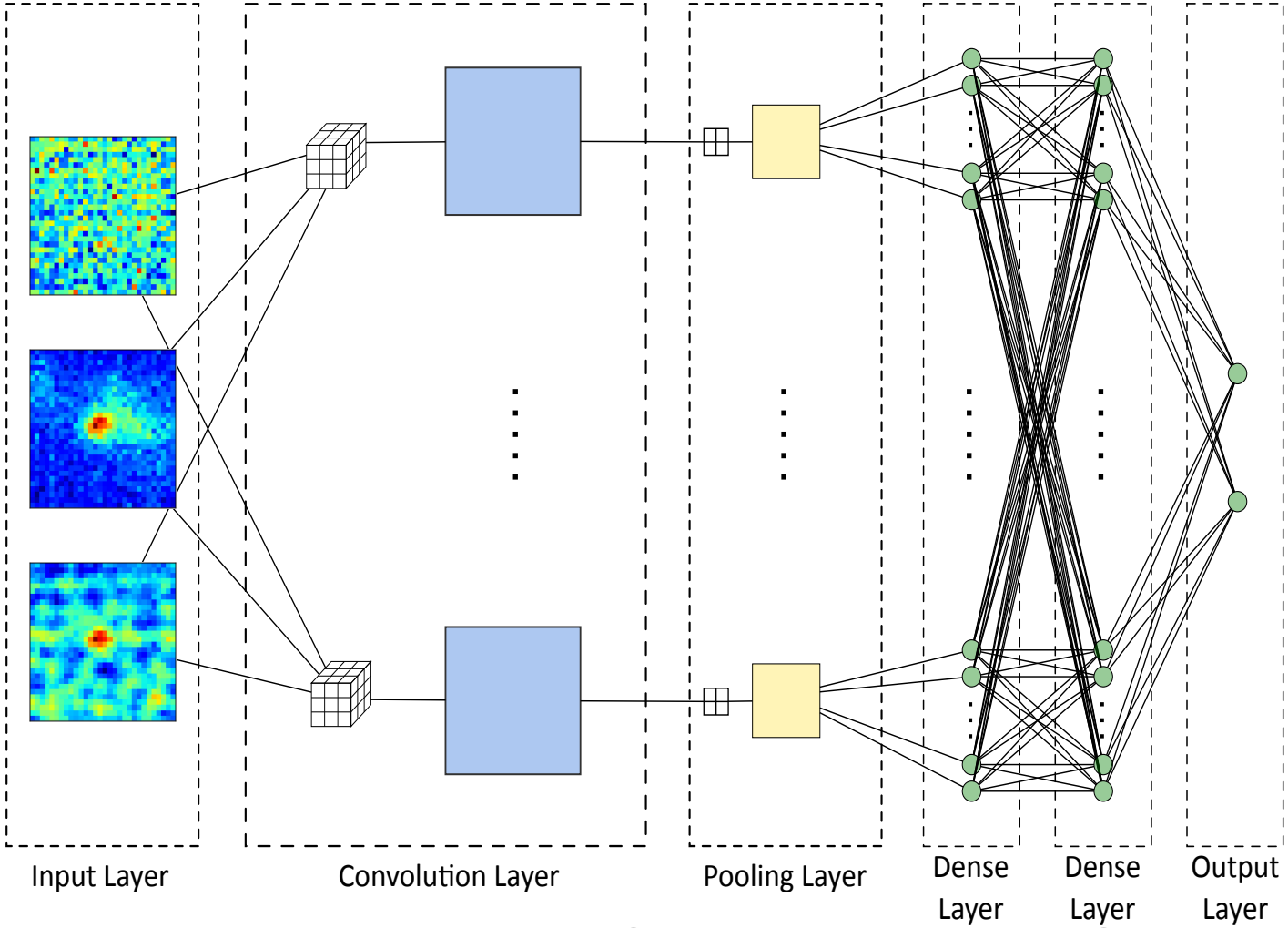
Bogus



Real

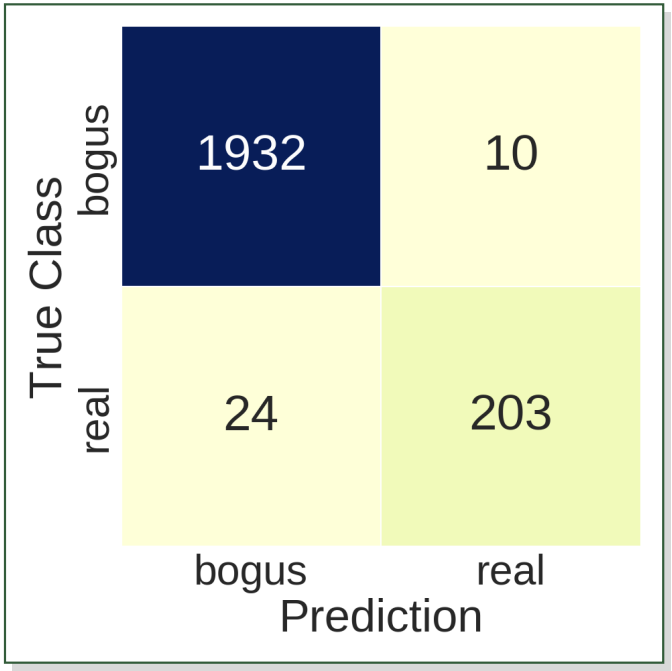
Gieseke, Bloemen, Bogaard, Heskes, Kindler, Scalzo, Ribeiro, van Roestel, Groot, Yuan, Möller, Tucker. *Convolutional Neural Networks for Transient Candidate Vetting in Large-Scale Surveys*, under review, 2017.

Transient Detection via CNNs

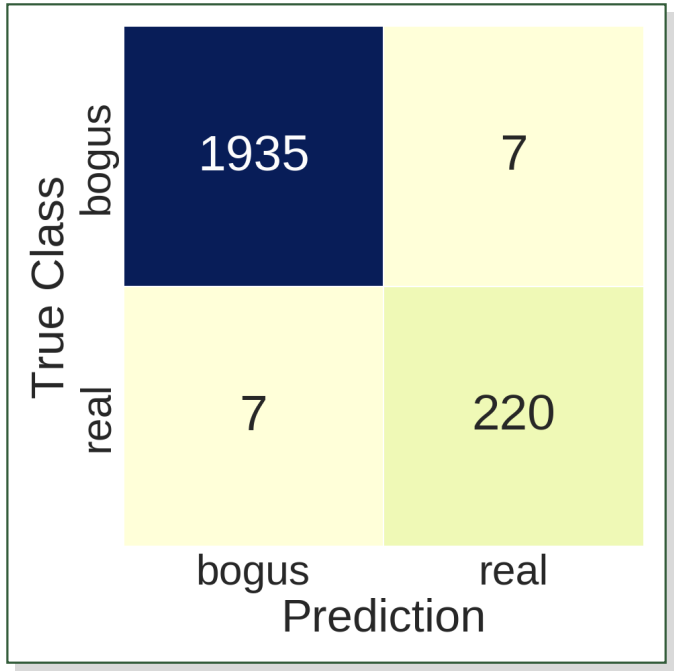


Gieseke, Bloemen, Bogaard, Heskes, Kindler, Scalzo, Ribeiro, van Roestel, Groot, Yuan, Möller, Tucker. *Convolutional Neural Networks for Transient Candidate Vetting in Large-Scale Surveys*, under review, 2017.

Transient Detection



Current Model

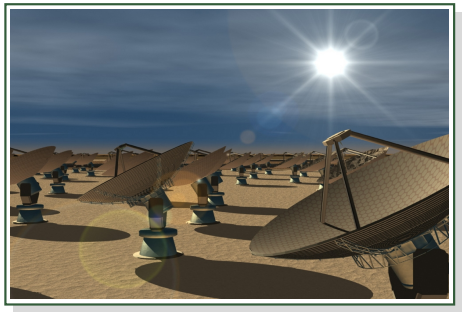


Convolutional Neural Network

Gieseke, Bloemen, Bogaard, Heskes, Kindler, Scalzo, Ribeiro, van Roestel, Groot, Yuan, Möller, Tucker. *Convolutional Neural Networks for Transient Candidate Vetting in Large-Scale Surveys*, under review, 2017.

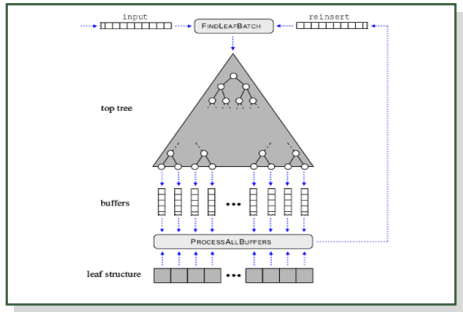
Outline

Large-Scale Machine Learning



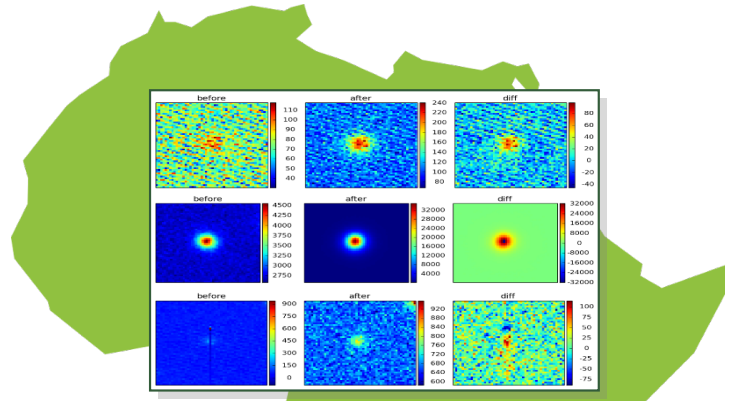
Big Data

- Huge increase in data volumes!
- Today: TBs
- Tomorrow: PBs



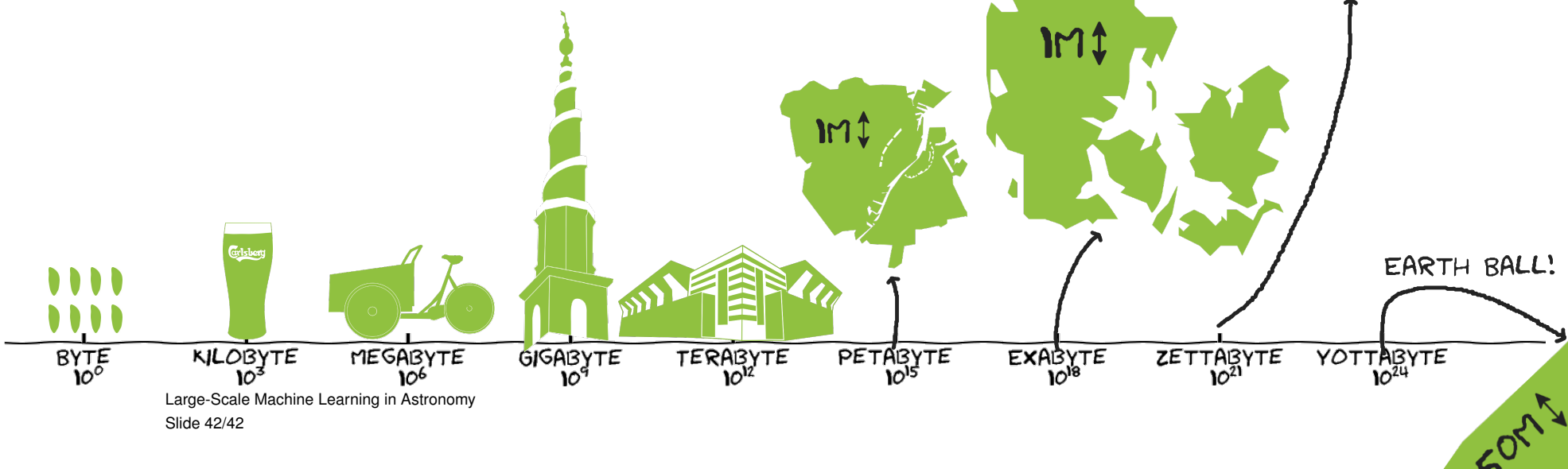
Large-Scale Machine Learning

- More data → better models!
- Time-consuming analysis
- Combination of many techniques

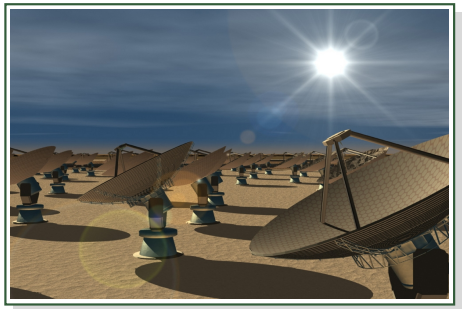


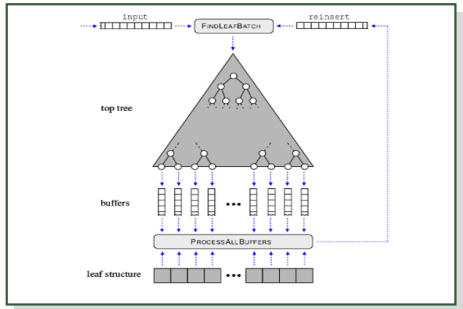
Astronomy

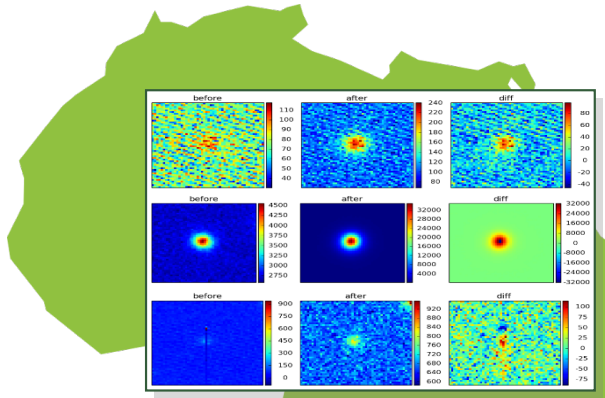
- Many challenging problems!
- Often: Important to use all data
- Interdisciplinary research



Large-Scale Machine Learning







Big Data

- Huge increase in data volumes!
- Today: TBs
- Tomorrow: PBs

Large-Scale Machine Learning

- More data → better models!
- Time-consuming analysis
- Combination of many techniques

Astronomy

- Many challenging problems!
- Often: Important to use all data
- Interdisciplinary research