Mapping Large Scale Structure in the Universe
A(n On-going) Case Study in Data Mining

Jim Heasley
Institute for Astronomy
University of Hawaii

Data Mining

- The primary purpose of data mining is to find meaningful patterns in data and where appropriate use these patterns to make predictions about the data.
- In this talk I will show an example of data mining as applied to a specific (albeit important) aspect of mapping the large scale structure in the universe.

Sky Surveys – A Cottage Industry

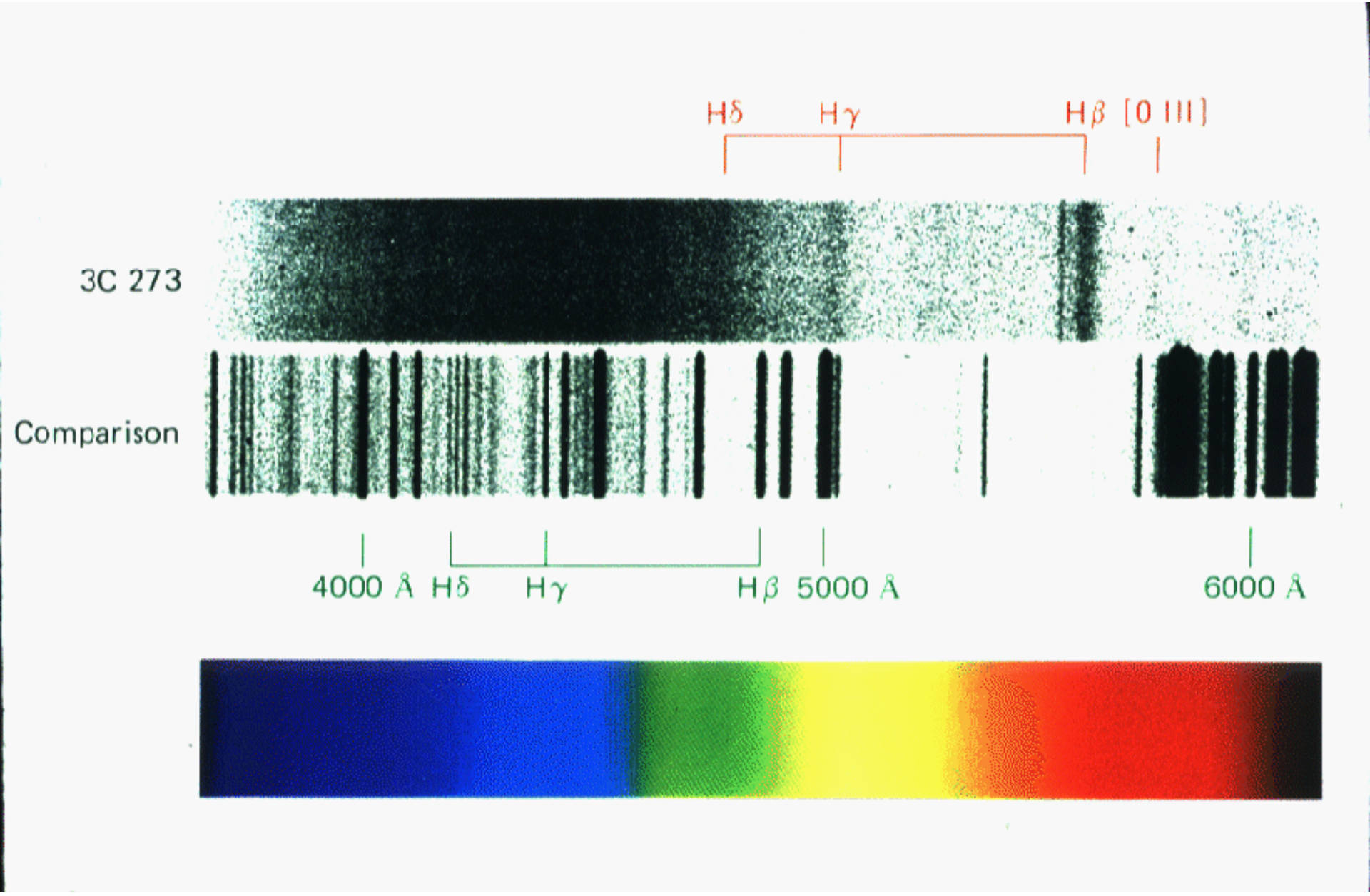
- The Pan-STARRS survey on Haleakala is the modern extension of the pioneering work of the Sloan Digital Sky Survey, covering the entire sky visible from Hawaii. Other surveys underway or planned include Skymapper (Australia), the Dark Energy Sky Survey (Chile), and starting around 2020, the Large Synoptic Survey Telescope (Chile).
- A common science goal of these programs to examine the spatial structure of matter in the universe in order to look for subtle signs imprinted upon it by the conditions in the early universe.

Fact – Images are projections!

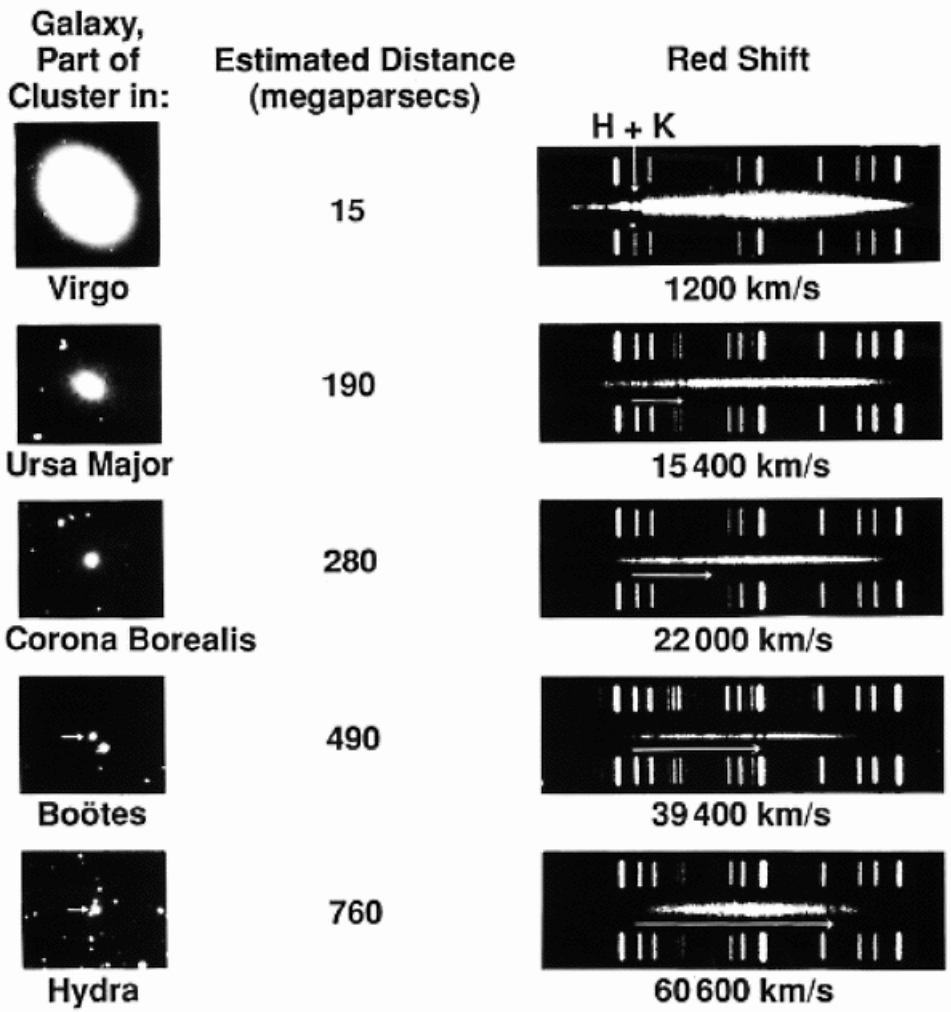
- All of the aforementioned sky surveys are imaging programs. As such, they produce pictures of the sky (usually in different wavelength bands). However, the images themselves are projections of the 3-dimensional distribution of sources in the universe onto the plane of the sky.
- Fortunately, we can use the information in the different filter bands to derive an estimate of a sources distance.

Edwin Hubble establishes the distance-redshift relationship

- In the 1920s Hubble showed that the distance to a galaxy is proportional to its velocity moving away from us. This is determined by measuring its redshift (recession velocity) from in its spectrum due to the doppler effect

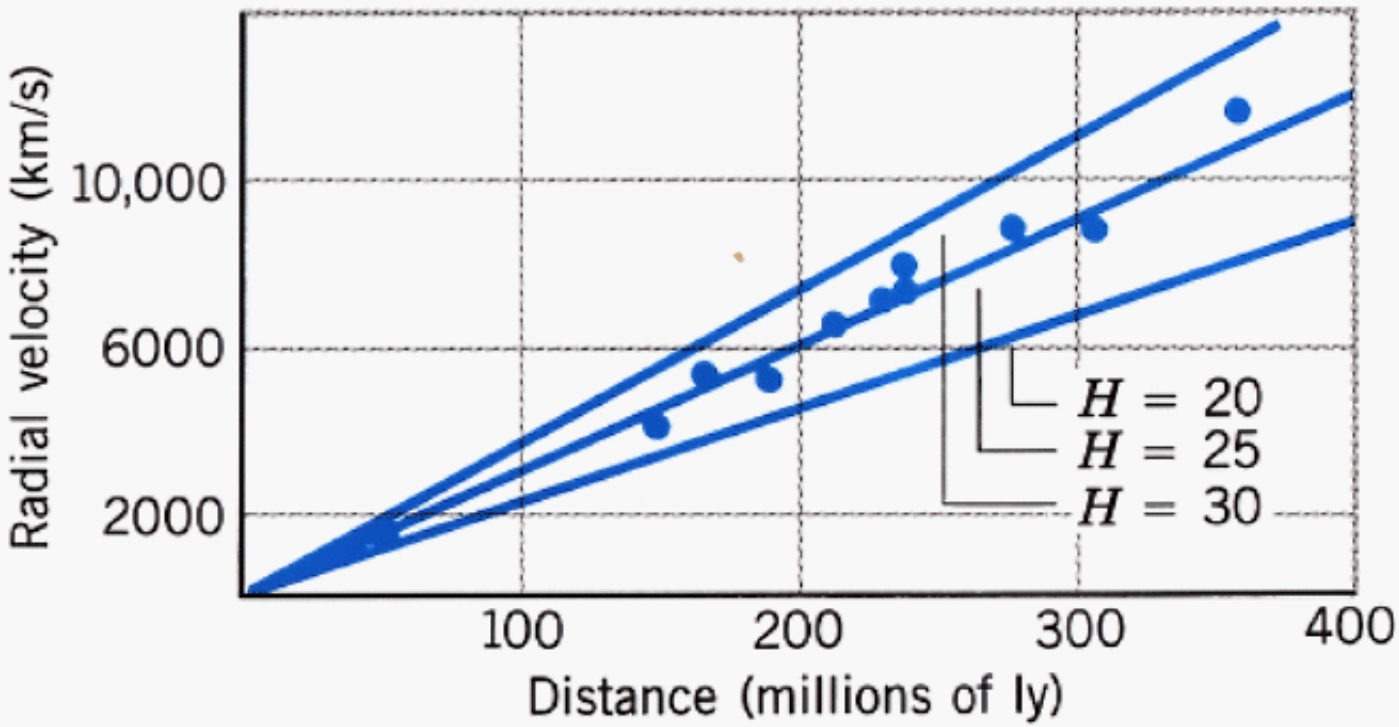

Hδ
Hγ
Hβ [O III]
3C 273
Comparison
4000 Å
Hδ
Hγ
Hβ
5000 Å
6000 Å

Relation Between Red Shift and Distance for Remote Galaxies



Red shifts of remote galaxies

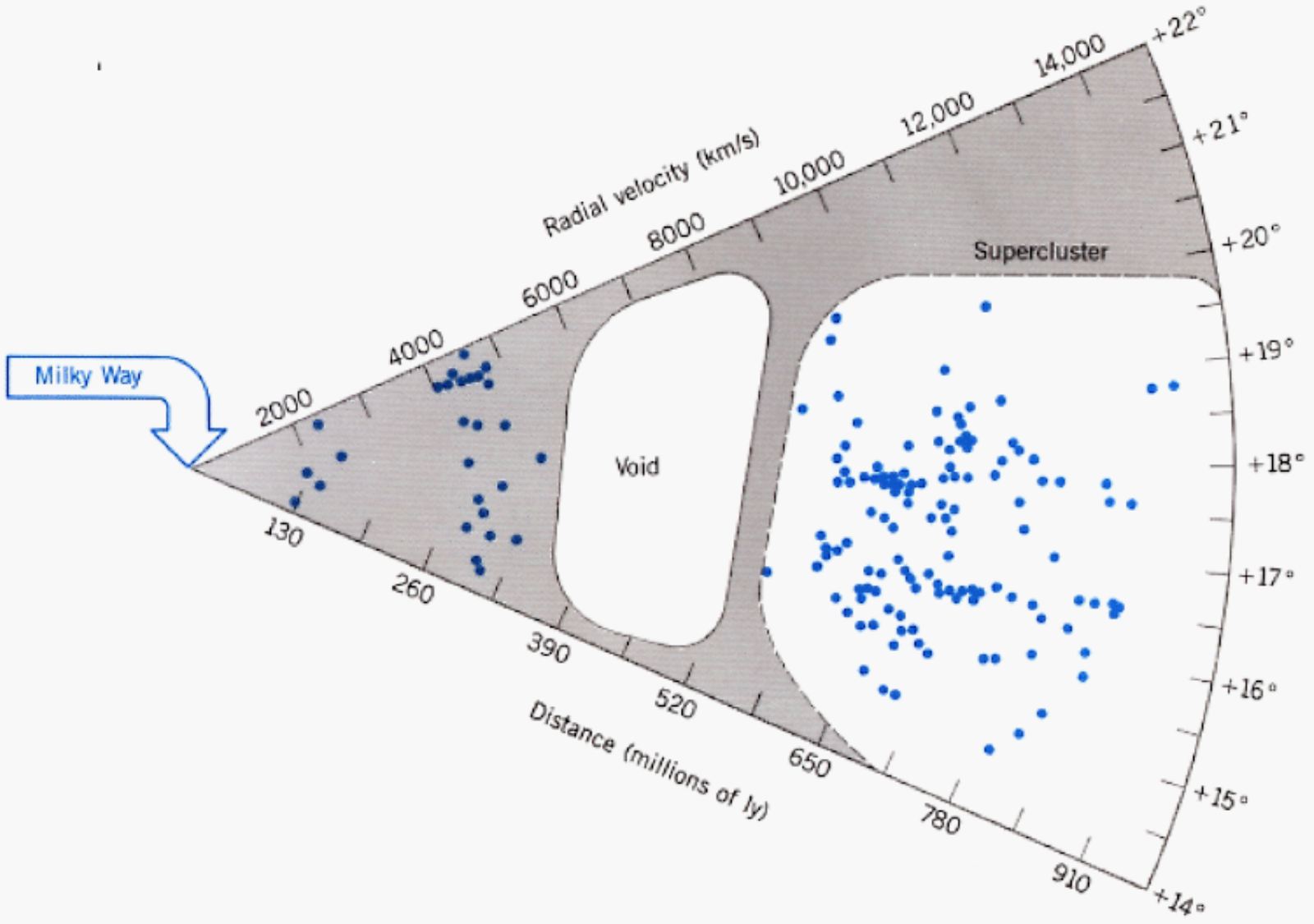
Hartmann/Impey: The Cosmic Journey, 5th ed., Fig. 25–2; Hartmann: The Cosmic Voyage, 1992 ed., Fig. 25–2



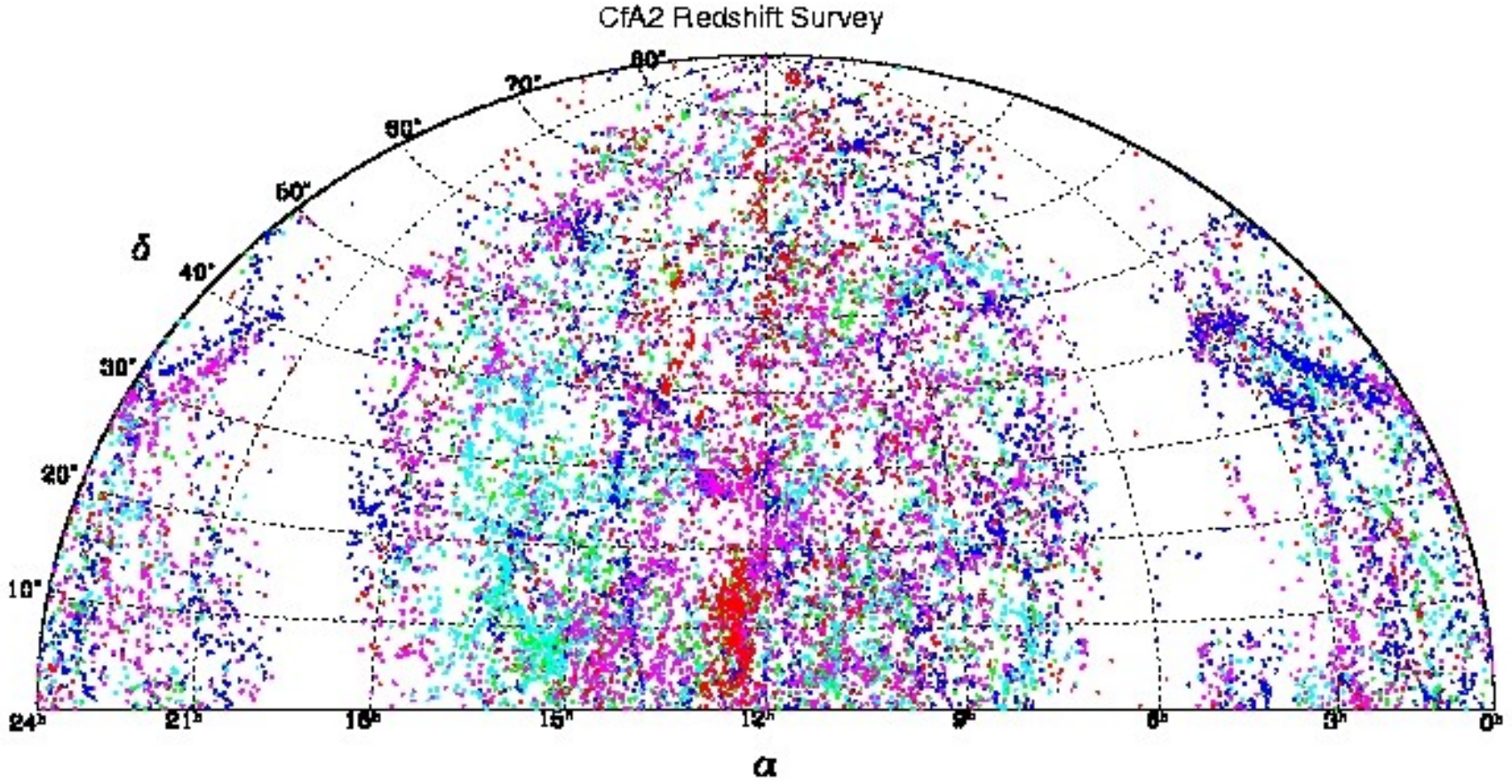
Hubble plot using Tully – Fisher relation and infrared fluxes for nearby calibrating galaxies to estimate the distances to eleven clusters of galaxies.

Putting it together!

- In the late 1970s astronomers started to map the 3-d structure of matter in the universe by making use of spectroscopy to get redshifts to galaxies and hence their distances.
- The next slide shows an early result from this sort of work. It is apparent that matter is not distributed uniformly through space but appears to be “clumpy” on large scales.
- The slide following that is a much more recent map showing large scale structure from a large sample of galaxies.



Slice of the Hercules supercluster along with the clusters Abell 2197 and 2199.

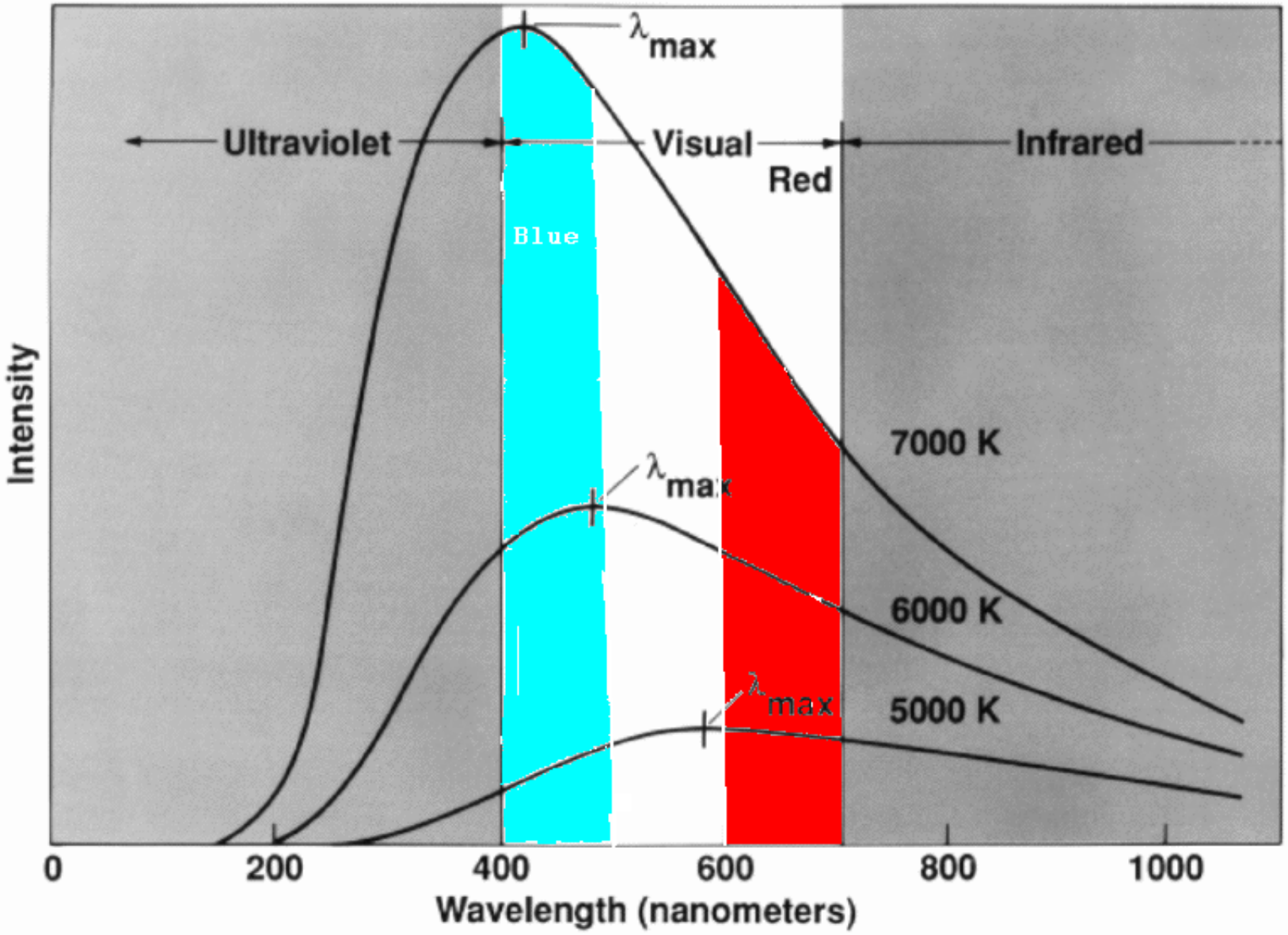


So many galaxies, so little time…

- While a great deal of progress since the first trials of this approach, getting the spectroscopy to measure redshifts and hence derive the distances is very time consuming.
- Even with modern spectrographs where one can get spectra for multiple galaxies at a time, it is clear that we will never be able to get such data except for the brightest galaxies.
- Another approach is needed to determine distances, even approximate ones, to the majority of galaxies being recorded in surveys such as Pan-STARRS.

Photometric Redshifts

- When one takes a survey image through a given filter you are sampling that portion of the source's energy distribution that falls inside the pass-band of the filter. Using multiple filters, one can sample – albeit crudely – the source's spectral energy distribution SED.
- Note that redshift doesn't just apply to spectral ones – the entire source's SED shifts to the red as a galaxy moves away from us.
- This led to the realization that one migh use multi-color images (photometry) of galaxies as a rapid means of obtaining redshifts to many sources at once.



Intensity of radiation emitted at different wavelengths

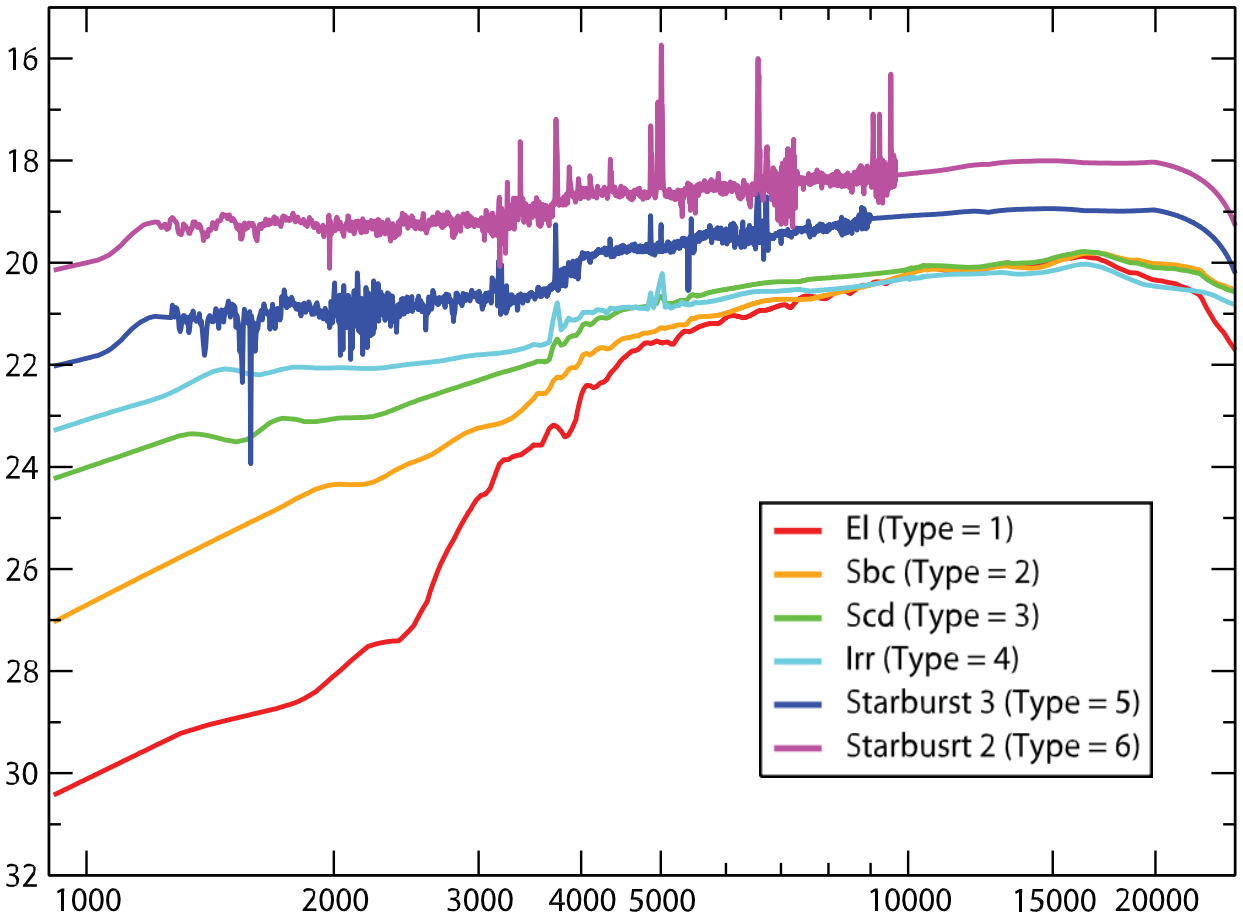
Seeds: Horizons, 1995 ed., Fig. 6–6; Foundations of Astronomy, 1994 ed., Fig. 7–2

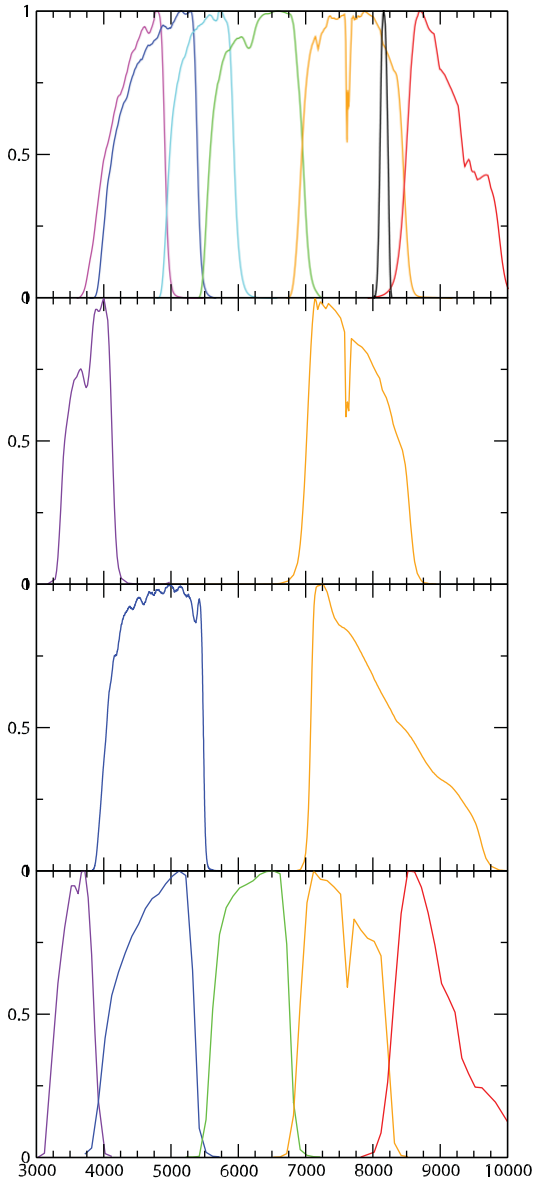
Photometric Redshifts

- Astronomers have recognized the importance of photometric redshifts (photoZ) for investigating the large scale structure of the universe for (at least) several decades.
- With the advent of large scale photometric sky surveys, it is now practical to employ these methods for cosmological studies.
- Approaches to finding photoZ fall into two camps:
  - Template Fitting
  - Empirical Approaches

Template Fitting Approach

- Select a sample of known SEDs that you believe represent the sample of galaxies for which you want to derive redshifts.
- For an assumed redshift, compute the predicted fluxes that would be measured in each of the observed filters.
- Compare the predicted fluxes with the observed fluxes. The template and redshift z which give the best match to the observation is assumed to be the photometric redshift for the target galaxy.

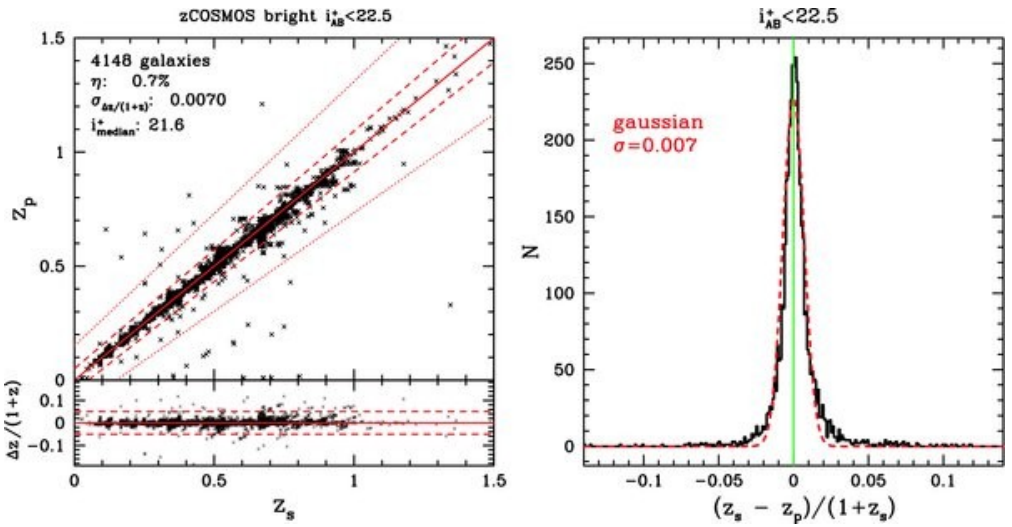

16
18
20
22
24
26
28
30
32
1000
2000
3000
4000
5000
10000
15000
20000
El (Type = 1)
Sbc (Type = 2)
Scd (Type = 3)
Irr (Type = 4)
Starburst 3 (Type = 5)
Starbusrt 2 (Type = 6)


1
0.5
0
0.5
0
0.5
0
0.5
0
3000
4000
5000
6000
7000
8000
9000
10000

State-of-the-Art in Template Fitting: The Cosmos 30-band PhotoZs

Ilbert et al 2009, A.J. 690, 1236

- PhotoZ determinations from 30 broad, intermediate, & narrow band photometry from UV (Galex) to mid-IR (Spitzer).
- Calibrated with spectra from VLT & Keck
- Used Le PHARE template fitting
- Include adjustments for emission lines – improves fit by ~2.5x to yield $\sigma_{\Delta z/(1+z)}$ = 0.007



From Ilbert et al 2009

Empirical Approach -- The Basic Assumption

- Let **x** be a vector of attributes that characterize a member of the general population we are studying. Let **y** be the vector of attributes we want to predict.
- We assume that there is an underlying relationship **y** = f(**x,** β) that describes the relationship we want to derive, and there are no hidden variables. Here β represents parameters specific to the modeling approach.
- We often assume that the dependent variable varies smoothly with the independent variables.
- Given a ***training set*** that is a subset of the general population, we use these data to derive an approximation to our functional relationship f(**x**).

Empirical PhotoZ Methods

- Artificial Neural Networks
- Support Vector Machines
- Self-Organizing Maps
- Gaussian Process Regression
- Kernel Regression
- Linear/Nonlinear polynomial fitting
- Instance Based Learning & Nearest Neighbors
- Boosted Decision Trees
- Regression Trees

And these are just the ones I've found so far!

Empirical PhotoZ Methods

- Artificial Neural Networks
- Support Vector Machines
- Self-Organizing Maps
- Gaussian Process Regression
- Kernel Regression
- Linear/Nonlinear polynomial fitting
- Instance Based Learning & Nearest Neighbors
- Boosted Decision Trees
- Regression Trees

And these are just the ones I've found so far!

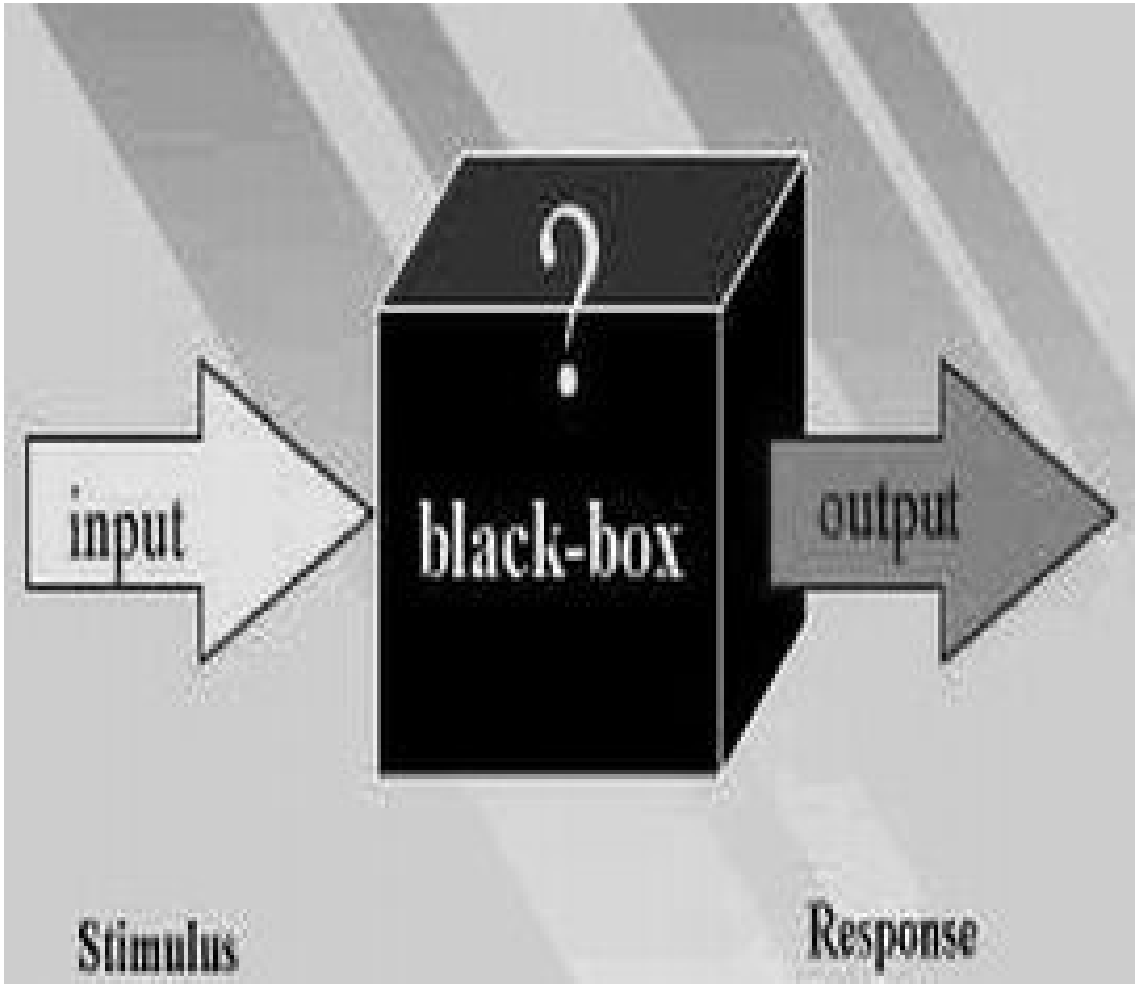
Some considerations

- We need a metric to determine how close $\mathbf{x}_i$ is to $\mathbf{x}_j$ We generally use the standard Euclidian metric.
- It may be necessary to scale the variables so they have a similar range.
- Too many independent variables may be a bad thing.
- It might be best to apply a PCA to perform dimension reduction before doing the analysis.

K-Nearest Neighbors

- For the training set, build a search structure such as a kd-tree or a box-decomposition tree (or your favorite multidimensional search tree).
- For any unknown source, find its k nearest neighbors ($k \geq 1$).
- Produce a weighted sum of the y_i values, i = 1 to k. Assign this weighted sum as the value of y for the unknown point.

SVMs, SOMs, ANNs



Support Vector Machines

- **Support vector machines**) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis . The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, a SVM training algorithm builds a model that assigns new examples into one category or the other. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

- In addition to performing linear classification, SVMs can efficiently perform non-linear classification using what is called the kernel trick. implicitly mapping their inputs into high-dimensional feature spaces.

SVM Torch

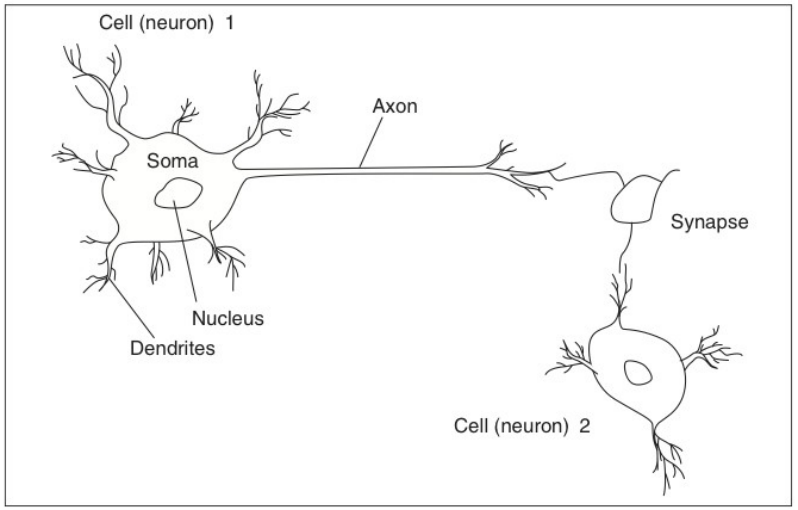
- The SVM methodology has been applied to photometric redshifts by Wadadekar (2005, Pub. Astro. Soc. Pacific 117, p. 79). He used the SVM Torch package, now part of the Torch machine learning libray (http://www.torch.ch/).
- He found a Gaussian kernel with $\sigma = 1$ and an error pipe size $\varepsilon = 0.01$ gave the "best" results.
- For a test set of data from the SDSS, he found an rms deviation of 0.027. This is comparable to the results found by other approaches for a similar data set.

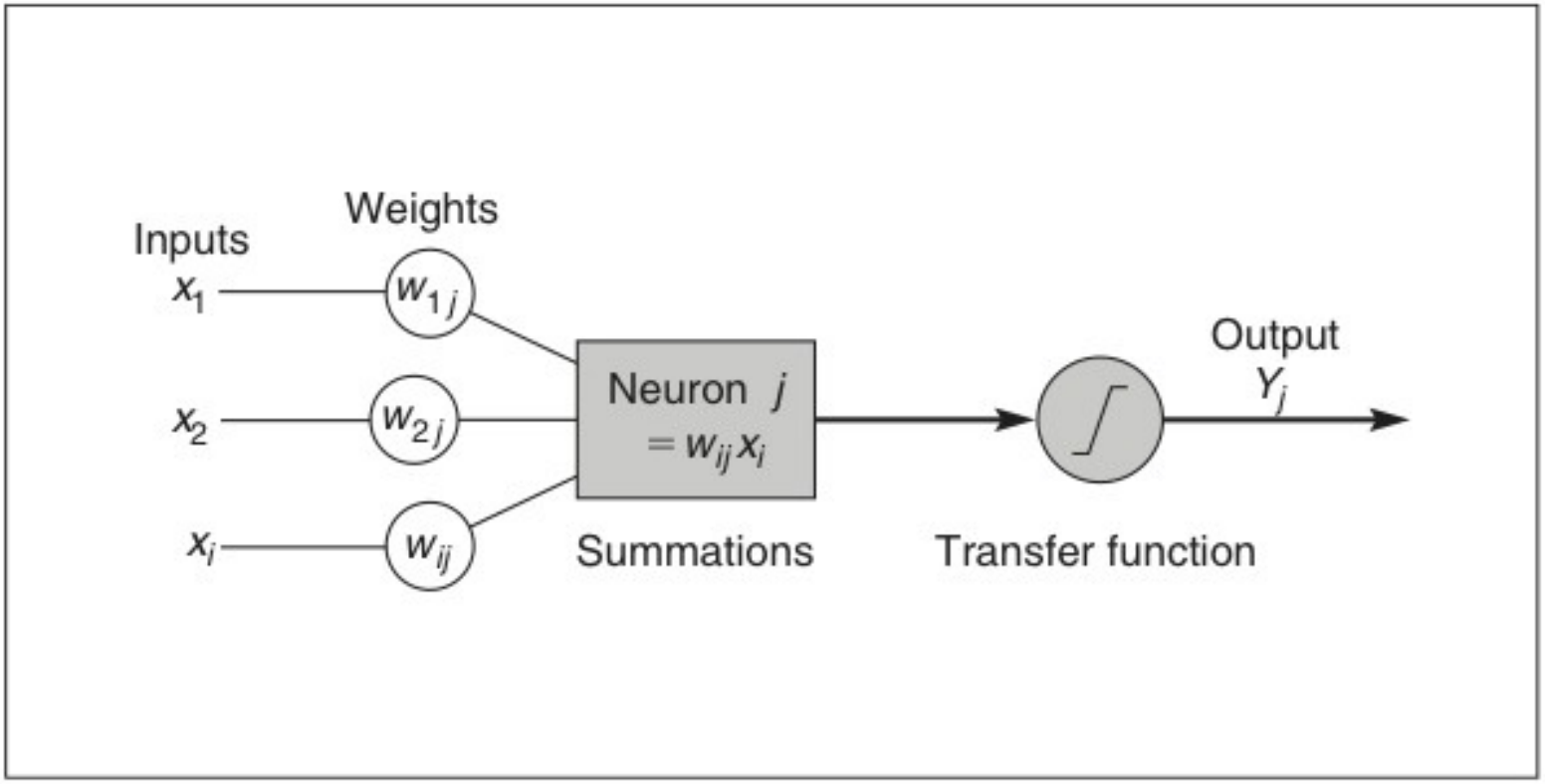
Self Organizing Maps – Side Bar

- A **self-organizing map** (**SOM**) (also known as Kohonen map) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a **map**. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.
- In the SOM generated, points that are close together in the original (high order) space also fall close together in the output map. This makes SOMs useful visualizing low-dimensional views of high-dimensional data.
- One can think of a SOM as a structure imposed on near neighbors.

Artificial Neural Networks

Neural networks represent a brain metaphor for information processing. The model is biologically inspired rather than an exact replica of how the brain actually functions. They are used in many forecasting applications and business classification applications due to their ability to "learn" from the data, their nonparametric nature (i.e., no rigid assumptions), and their ability to generalize.

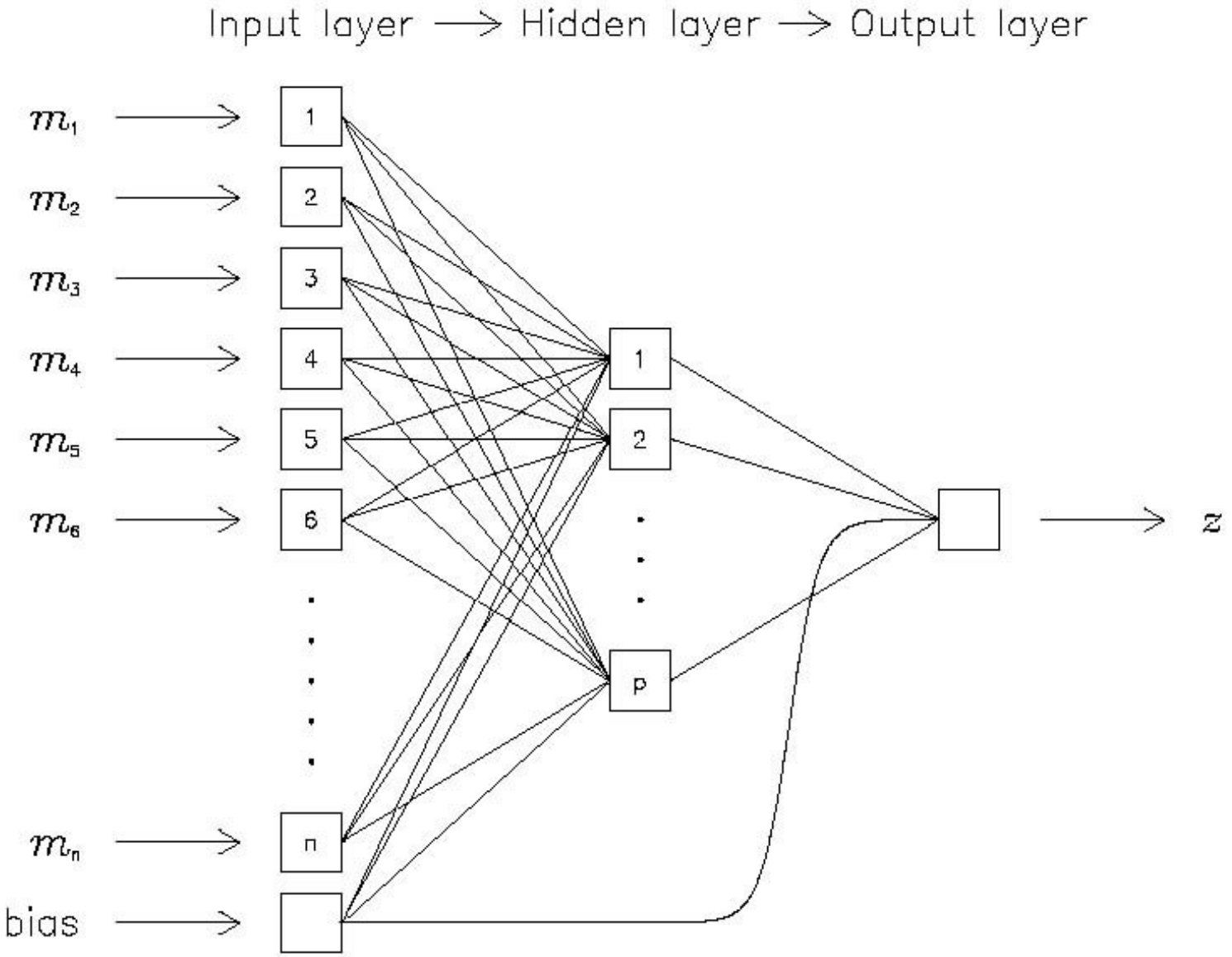




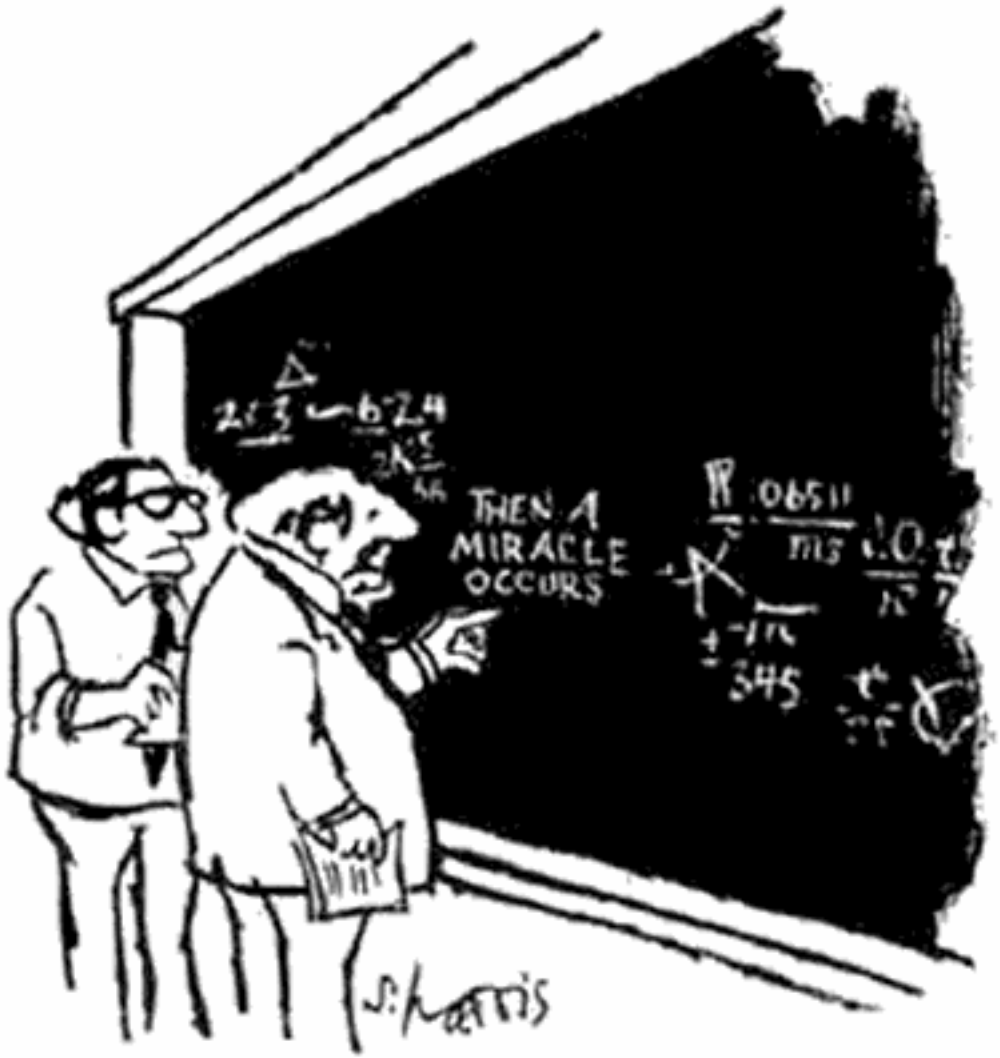
The computer neuron.

The basic building block of the ANN is a computer neuron (or node). It takes some number of inputs (stimulai) and generates a weighted sum of these. The weights represented the learned “knowledge” of the network. The weighted input is then fed to a transfer function (usually a sigmoid function) that produces the neuron’s output. Many neurons can be combined into a network, often with “hidden” layers between the initial input and the final output. The most common form of this is known as a multilayer perceptron neural network.

The Multilayer Perceptron Network used by the ANNz Software



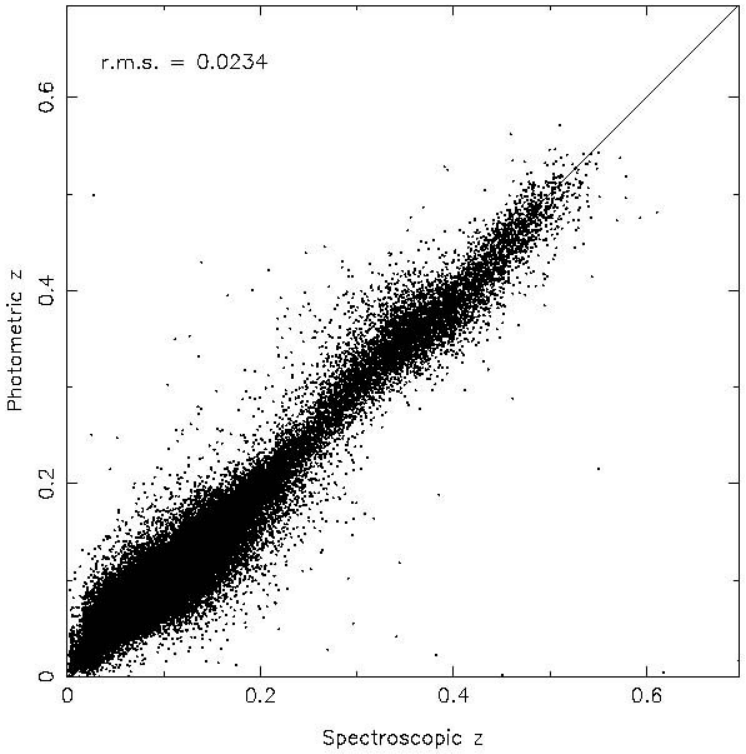
From Collister and LaHav 2004.

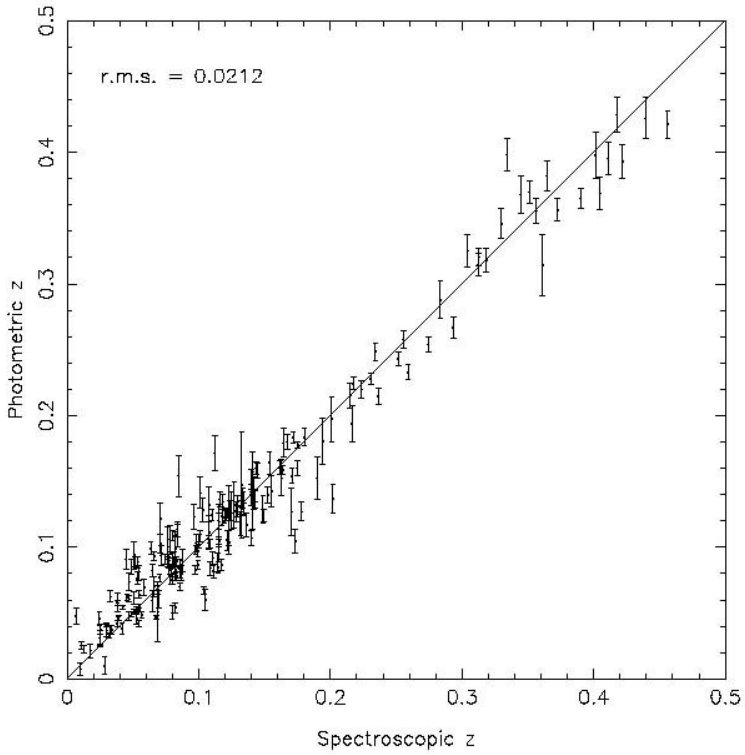


"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

ANNz

- Collister & LaHav, 2004 Pub.A.S.P. 116.345.
- Publicly available software (no longer maintained?)
- User provides input photometry & error estimates, spectroscopic redshift for training and validation data sets.
- Test set of unknowns (with or without redshift).
- User must specify the network topology (number of hidden layers and numbers of nodes in each).
- The key to the entire process is determining the weights that connect the compute neurons. This is done iteratively by minimizing a cost function (like a χ^2 term) + a regularization term (to keep the weights from becoming too large. Initial weights selected at random.
- Weights are found using iterative quasi-Newton method. One may get stuck in a local minimum. Use a “committee” approach of multiple networks to vote on the answer.





ANNz Output for the SDSS EDR

Left panel: Comparison between spectroscopic redshifts for 10,000 randomly selected galaxies from the SDSS EDR with photometric redshifts derived by ANNz. The RMS scatter in Δz is 0.0229. The right panel shows a random selection of 200 galaxies from the left panel, shown with the error estimates provided by ANNz. The test example provided with the ANNz code is almost identical to this case and produces a similar (small) scatter "out-of-the-box."

From Collister and LaHav 2004.

Cubist – A Decision Tree Based Approach

- The last method I want to describe is implemented by the CUBIST algorithm developed by Ross Quinlan (rulequest.com).
- Cubist partitions the input training set using a decision tree approach, but rather than having a discrete value at each tree leaf, it uses a multi-linear function to describe the data for that partition.
- Rulequest claims that cubist performs almost as well as more complicated algorithms like ANNs, but is computationally much faster. This is verified by the tests I have performed.

An Empirical Cookoff

- Data set: 12,000 photometry+z tuples from the SDSS early data release (the standard AnnZ test example).
- Performed 10-fold cross validation over the entire sample for AnnZ, SVMTorch, a home grown k-NN procedure, and Cubist.
- As expected, AnnZ and SVMTorch were much slower than the nearest neighbor and Cubist methods.

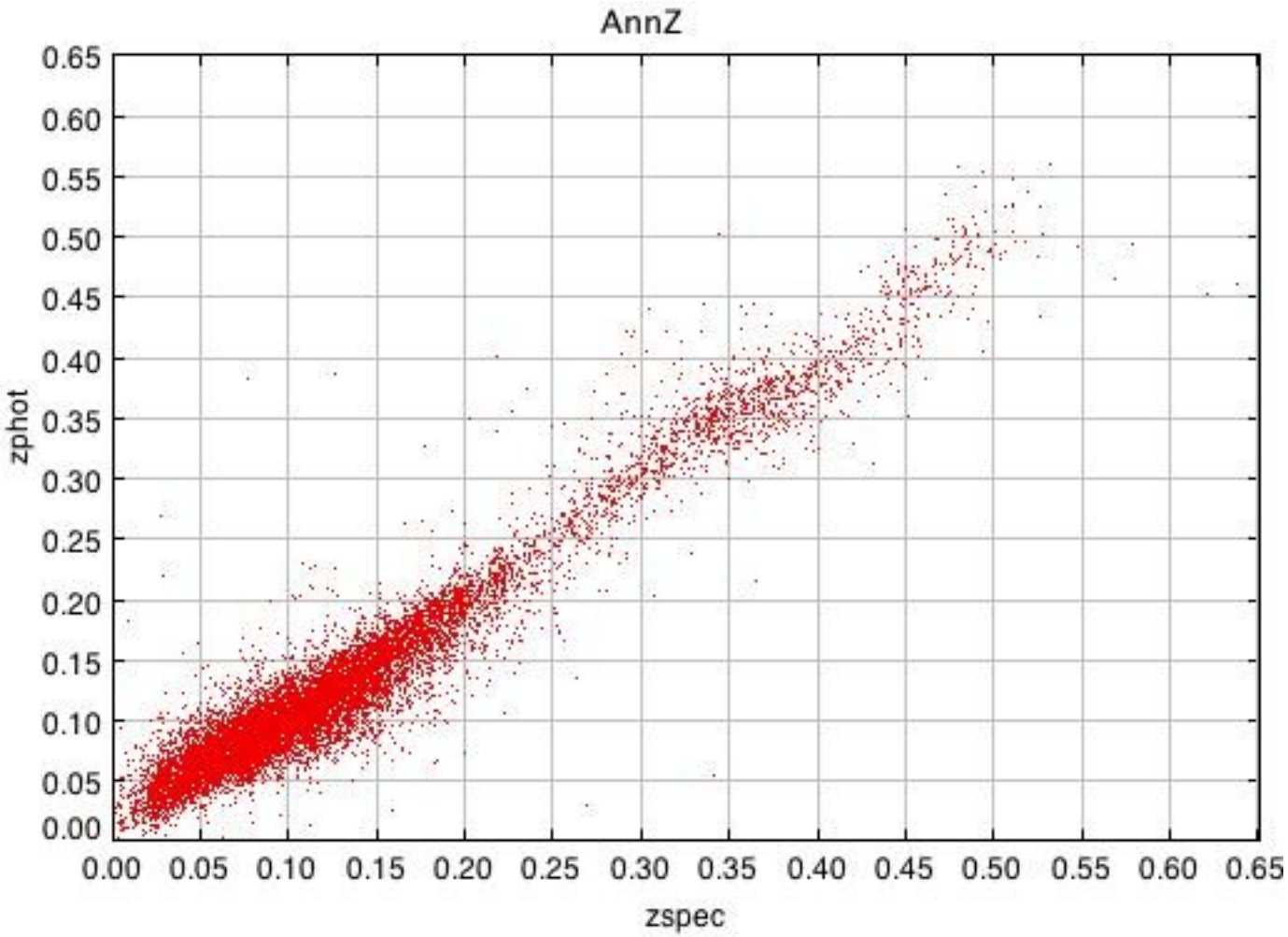
And the results are

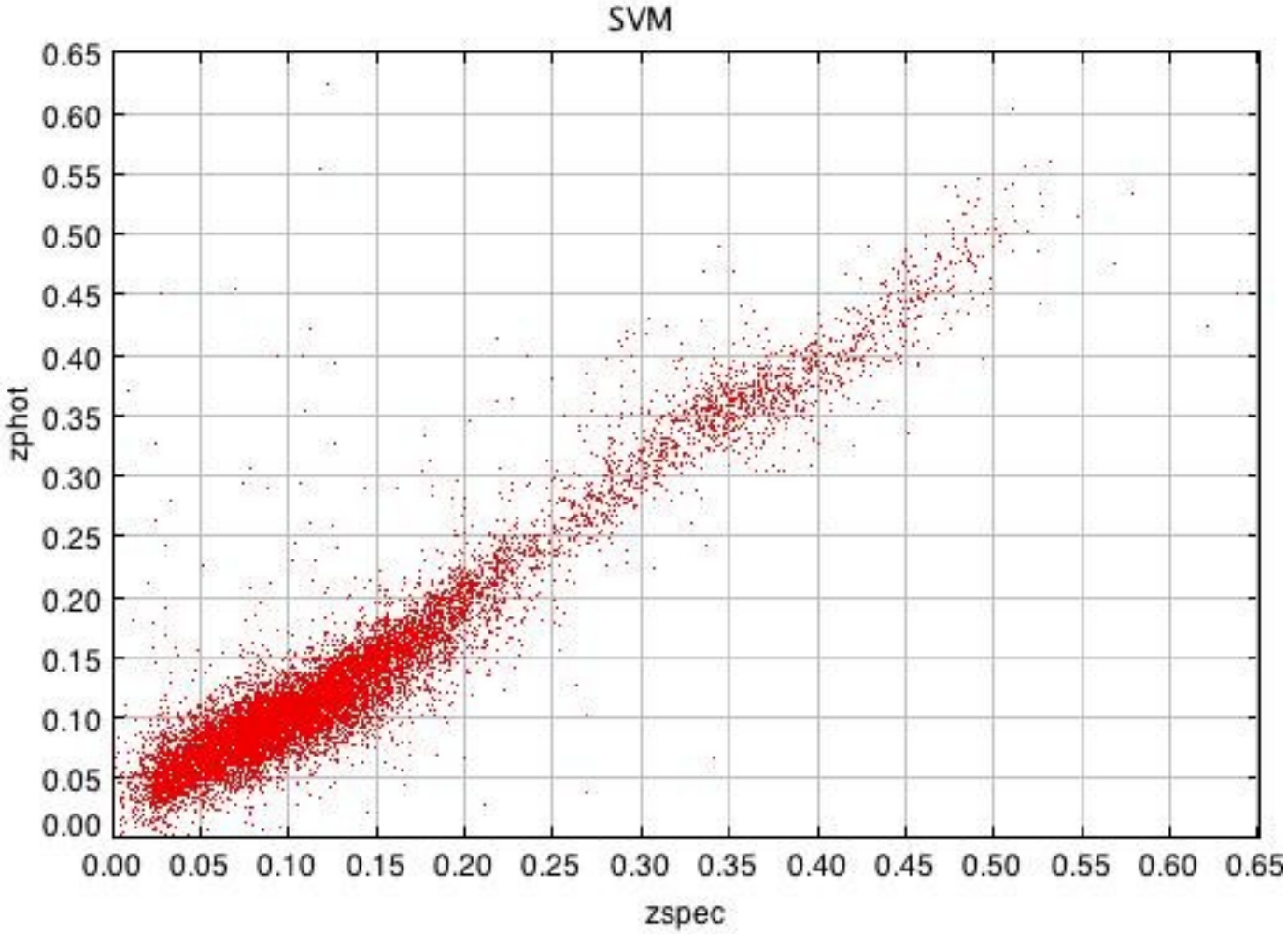
Using Magnitudes

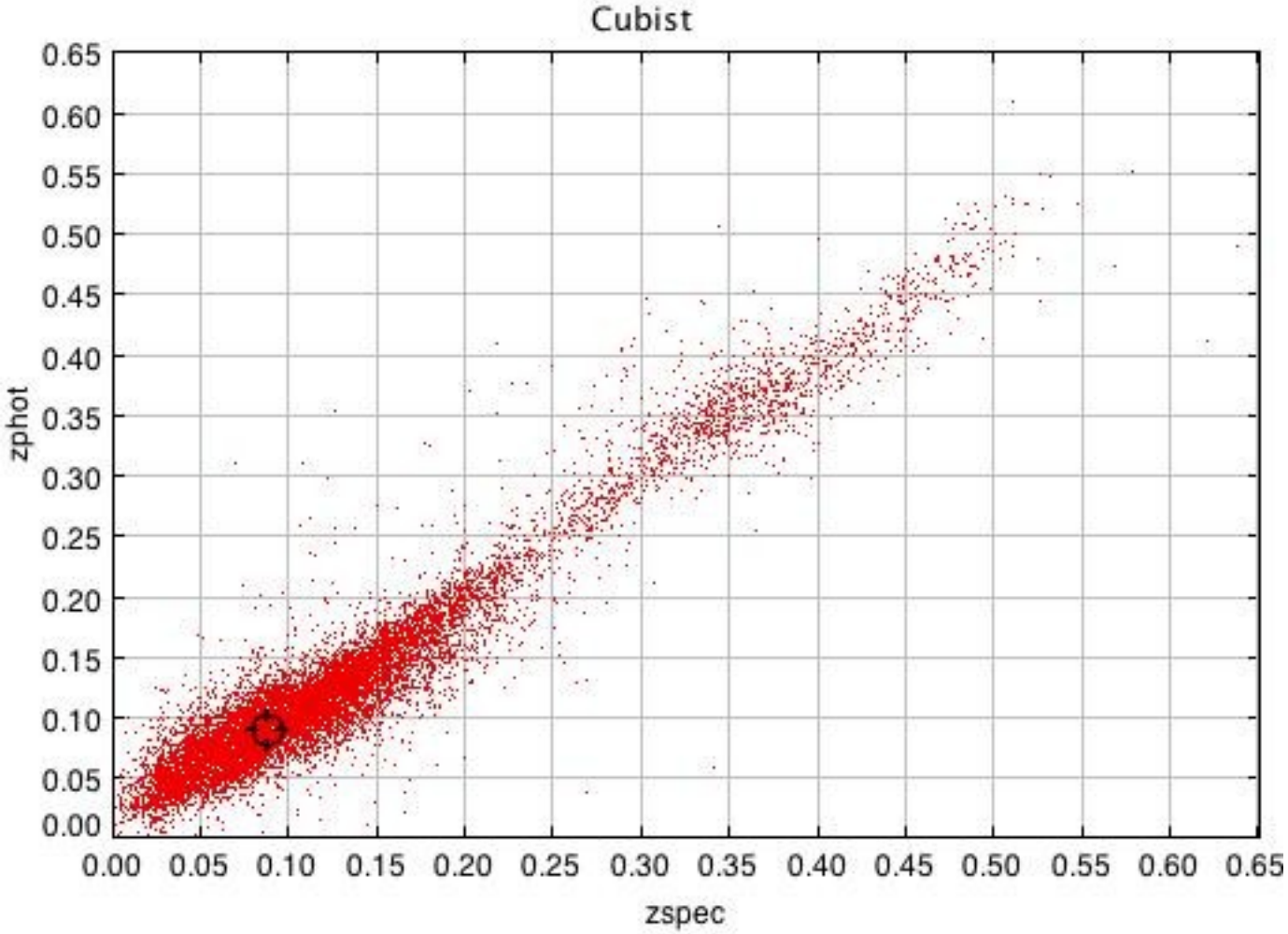
Method	Average Δz	adev	sdev	# outliers
AnnZ	2.44 e-5	0.0163	0.0219	46
SVM Torch	-1.54e-4	0.0173	0.0231	124
k-NN	8.52e-4	0.0220	0.0285	108
Cubist	4.88e-4	0.0172	0.0230	63

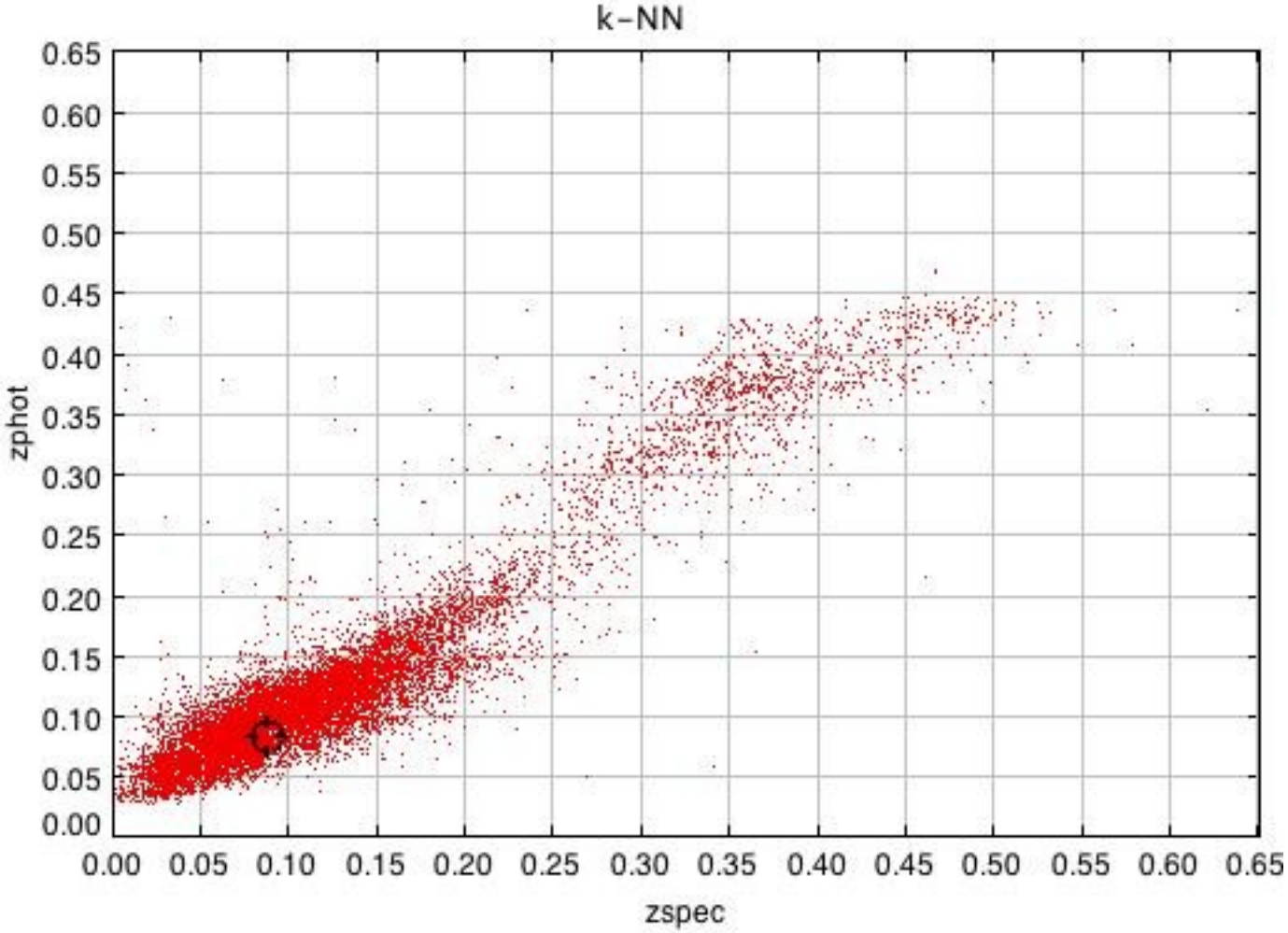
Using Colors

Method	Average Δz	adev	sdev	# outliers
AnnZ	7.46 e-5	0.0181	0.0241	68
SVM Torch	1.77e-3	0.0203	0.0264	122
k-NN	1.13e-3	0.0191	0.0257	111
Cubist	6.43e-4	0.0182	0.0243	71


AnnZ
zphot
zspec
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65


SVM
zphot
zspec
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65


Cubist
zphot
zspec
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65


k-NN
zphot
zspec
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65

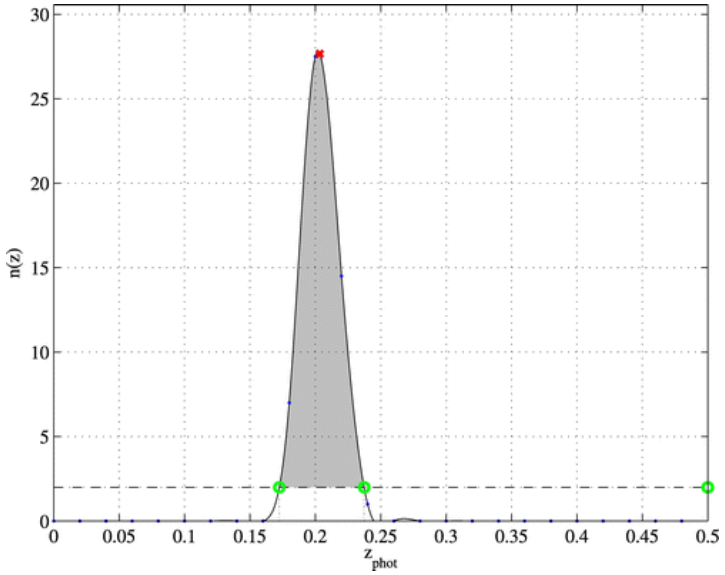
Photometric Redshift Distributions

- To estimate the photometric redshift distribution for each source we set $x_i = x_i + \sigma_i$ where σ_i is a vector where each component is the photometric error for the particular filter multiplied by a gaussian deviate.
- Repeat the full analysis to determine the photo-z measures (say using Cross validation).
- Do this many (~1000) times. Scramble the order of the data so we guarantee we create unique training sets on each pass.

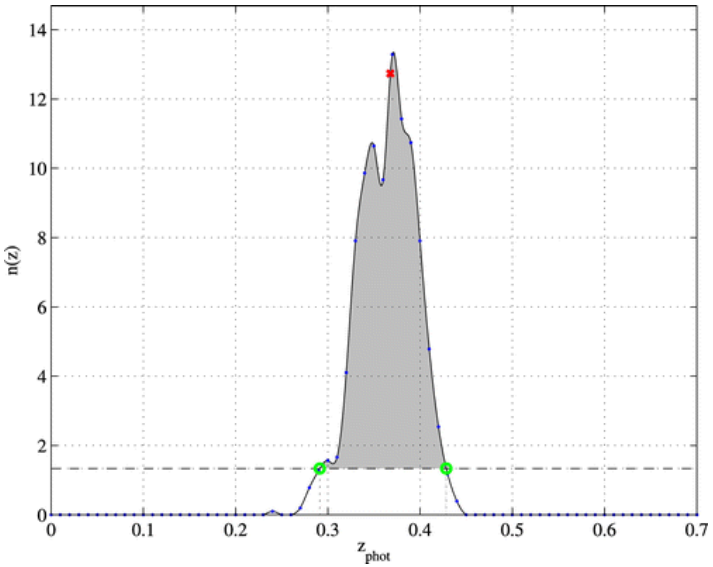
Photometric Redshift Distributions

- It is possible for a given photometric data set x_i to give a very different value for the photoZ after it is perturbed by the error vector.
- This can happen in cases where the solutions are degenerate, i.e., there is insufficient information to distinguish between cases. Usually the addition of observations in other filters can beak the degeneracy.
- Having the photometric redshift distribution can point out which cases are “questionable.”

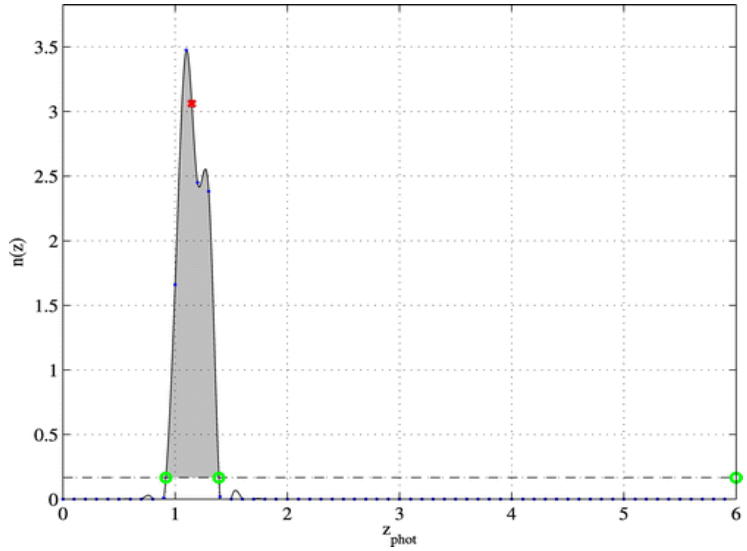
Main sample galaxy



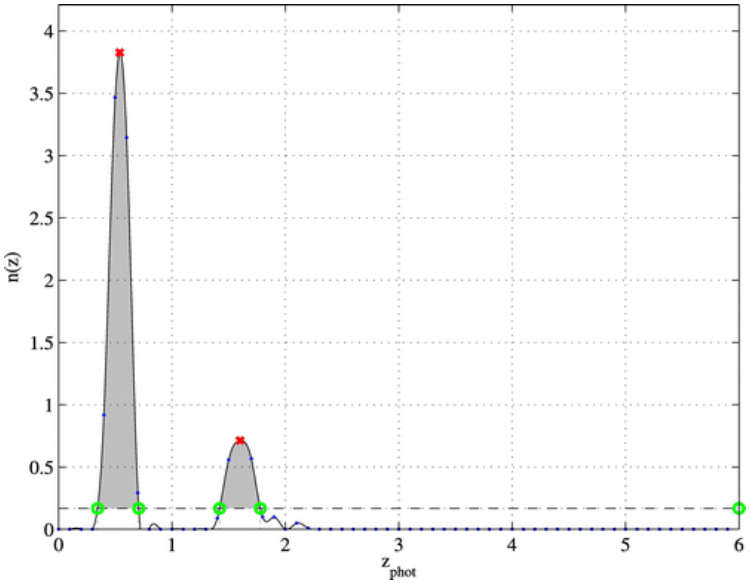
Luminous Red galaxy

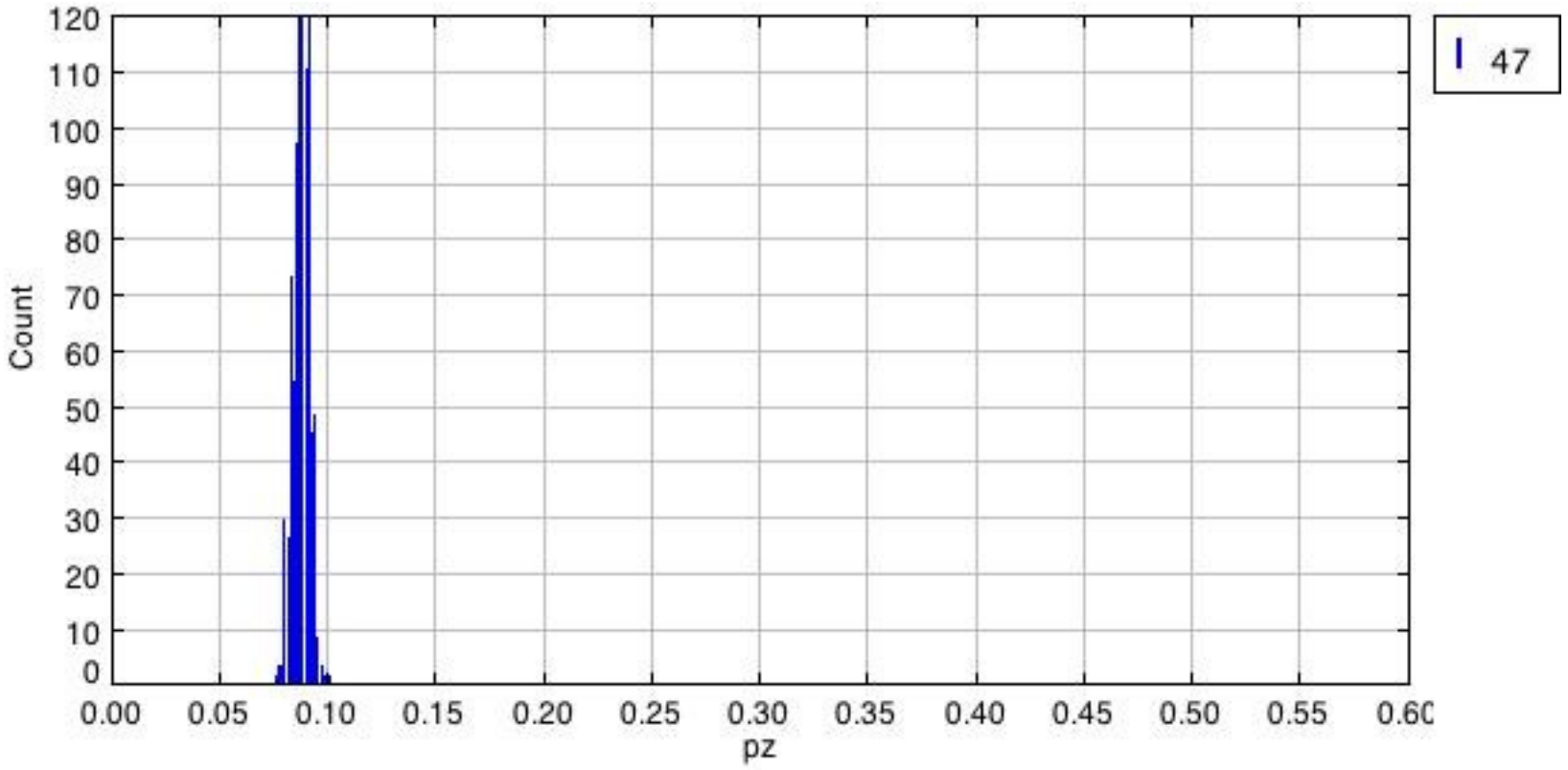


Single-peak QSO

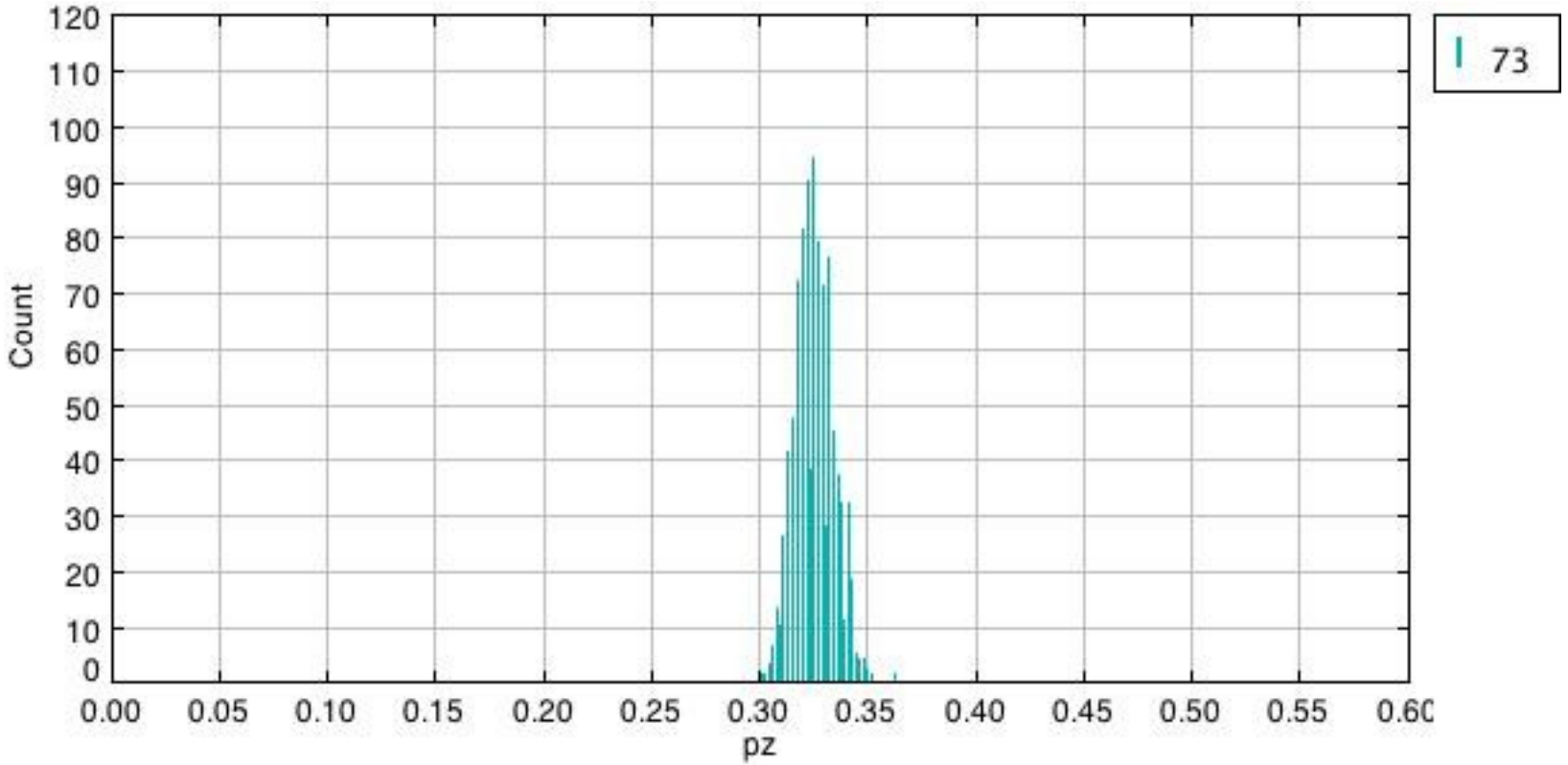


Dual-peak QSO





$Z_{spec} = 0.081$ $\quad$ $\langle z_{photo} \rangle = 0.0874$



$Z_{spec} = 0.306$ $\langle z_{photo} \rangle = 0.3248$