Statistical Learning

Or, From Nonparametric Regression to Deep Learning in an Hour

Chad M. Schafer
Department of Statistics & Data Science
Carnegie Mellon University
February 2018

Outline

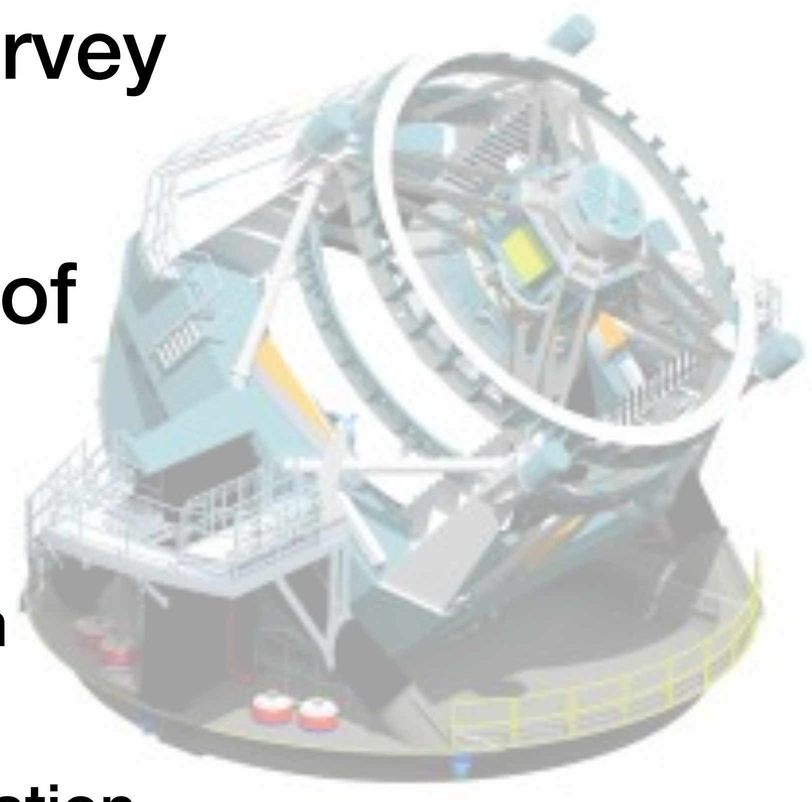
- Motivation: Representing Data
- Nonparametric Regression
- Curse of Dimensionality
- Additive Models
- Neural Nets
- Deep Learning

The LSST ISSC

Informatics & Statistics
Science Collaboration

- Informatics and Statistics one of eight LSST Science Collaborations
- Over 75 members and growing: data scientists and astronomers
- http://issc.science.lsst.org

LSST Basics



- 10-year photometric survey
- 3.2 Gigapixel camera
- 32 trillion observations of 40 billion objects
- Science Goals
  - Cataloging the Solar System
  - Exploring the Changing Sky
  - Milky Way Structure & Formation
  - Understanding Dark Matter and Dark Energy

Ivezić, et al. (2014)

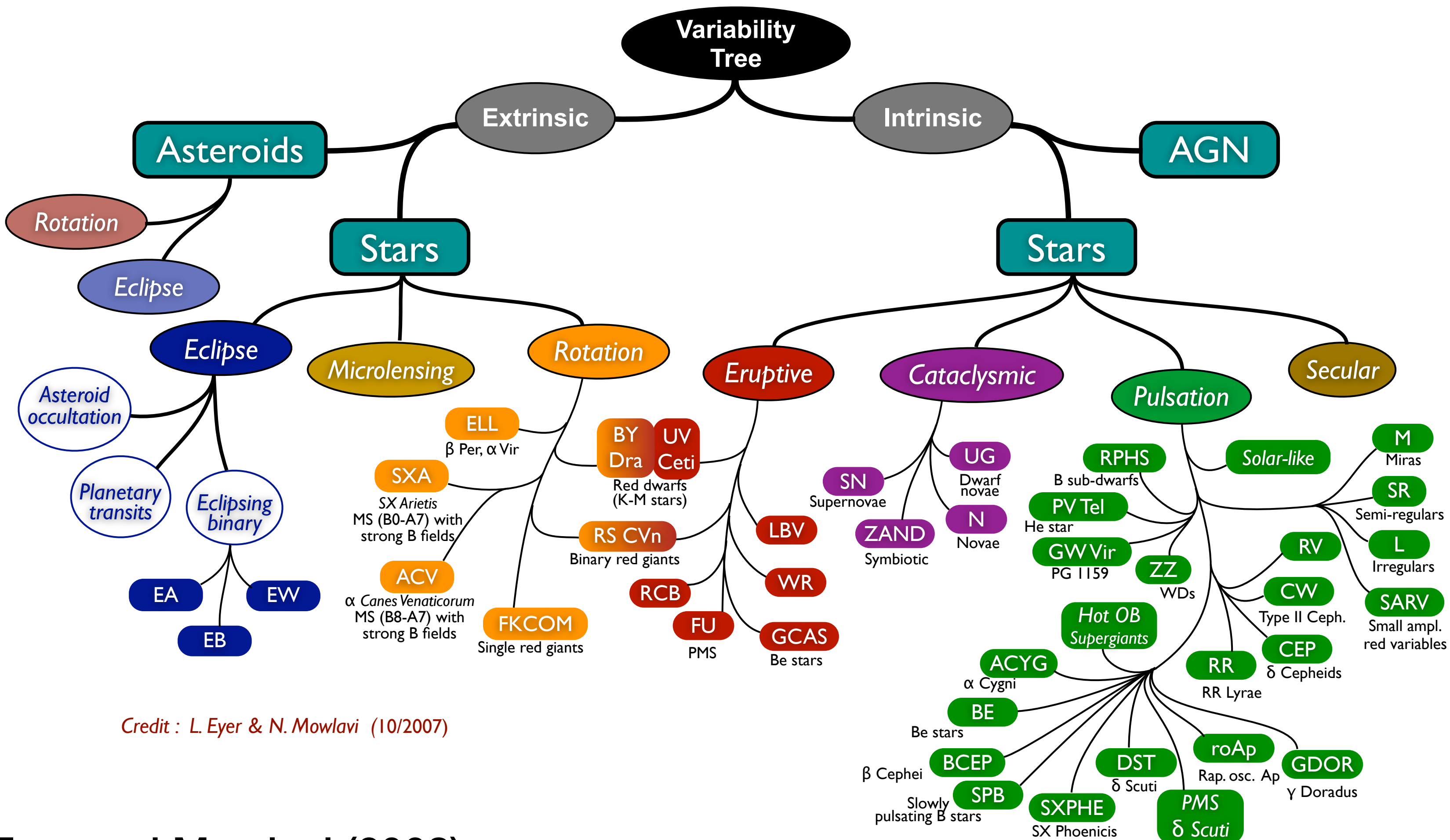
Common Themes

- General implementation challenges
- Existing procedures to LSST scales
- Expanding sophistication of analysis procedures in use
- Making the most of available data

Representations

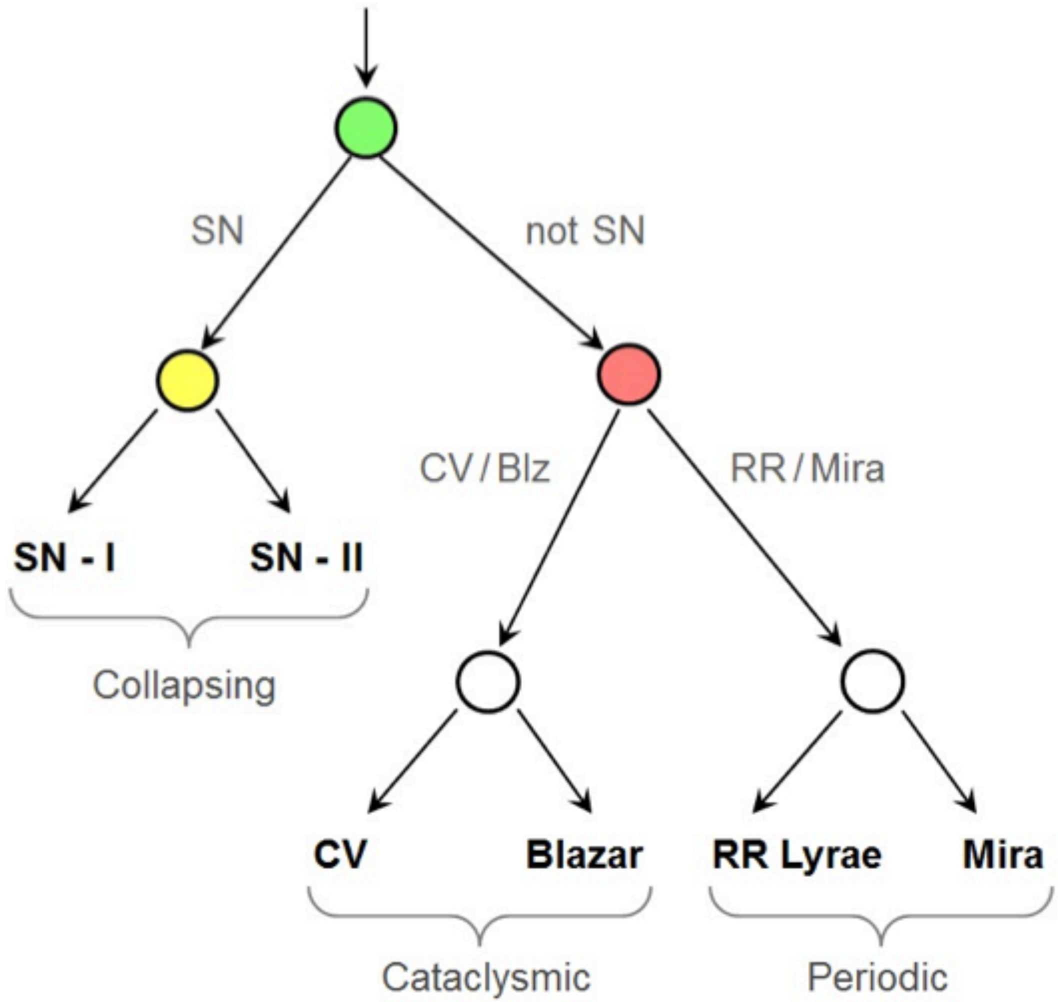
- A recurring challenge is representing observables in forms amenable to standard analysis tools
- The fundamental challenge of “Big Data”
- What summary statistic retains the important information in the data?
- Separating signal from noise

Classifying Variables



Eyer and Mowlavi (2008)

Classifying Variables



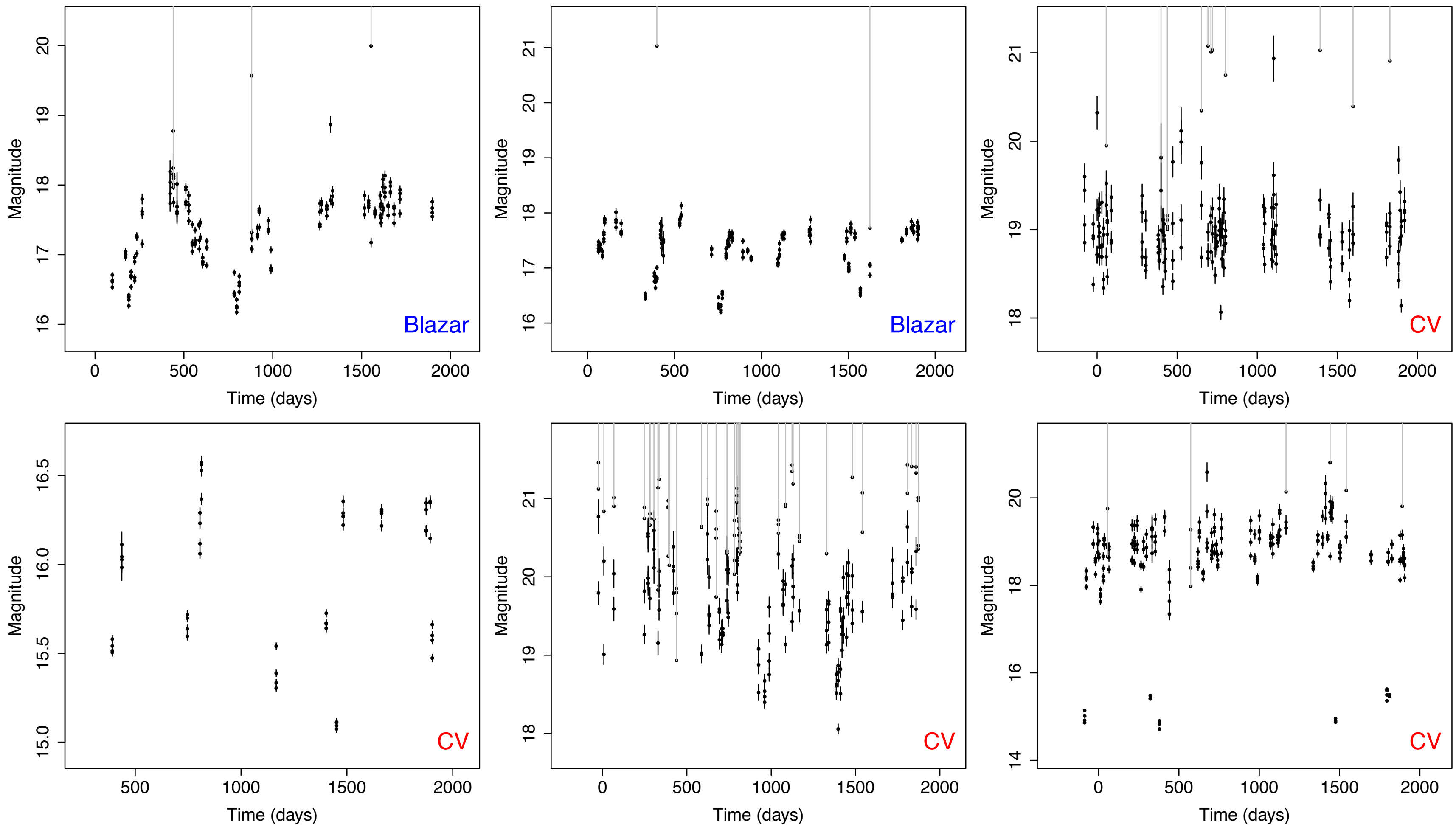
Blazars versus CVs

Cataclysmic Variables (CV) – binary system in Milky Way with matter transfer from secondary (normal) star to primary white dwarf

Blazars – Quasars with "jet" of energy pointed at Earth

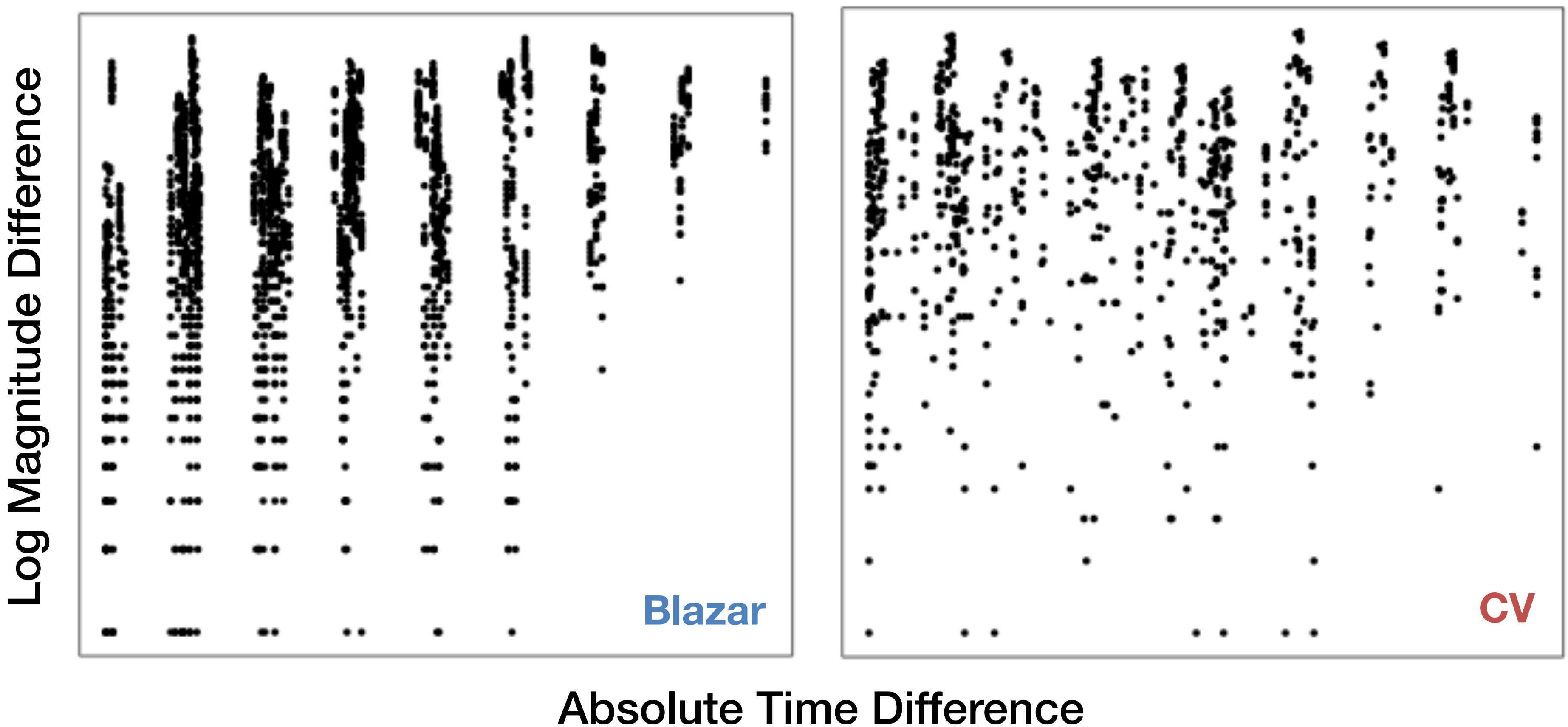
Both produce light curves with irregular variability, lacking periodic structure

Blazars versus CVs



Light Curves from Catalina Real-Time Transient Survey (Drake 2009)

Blazars versus CVs



Comparison of Structure Functions

Summarizing the SF

Typical to fit model to structure function

- Power Law Form (Schmidt et al.)
- Damped Random Walk (Kelly et al.)

Effort to find a low-dimensional representation, avoiding the curse of dimensionality

Summarizing the SF

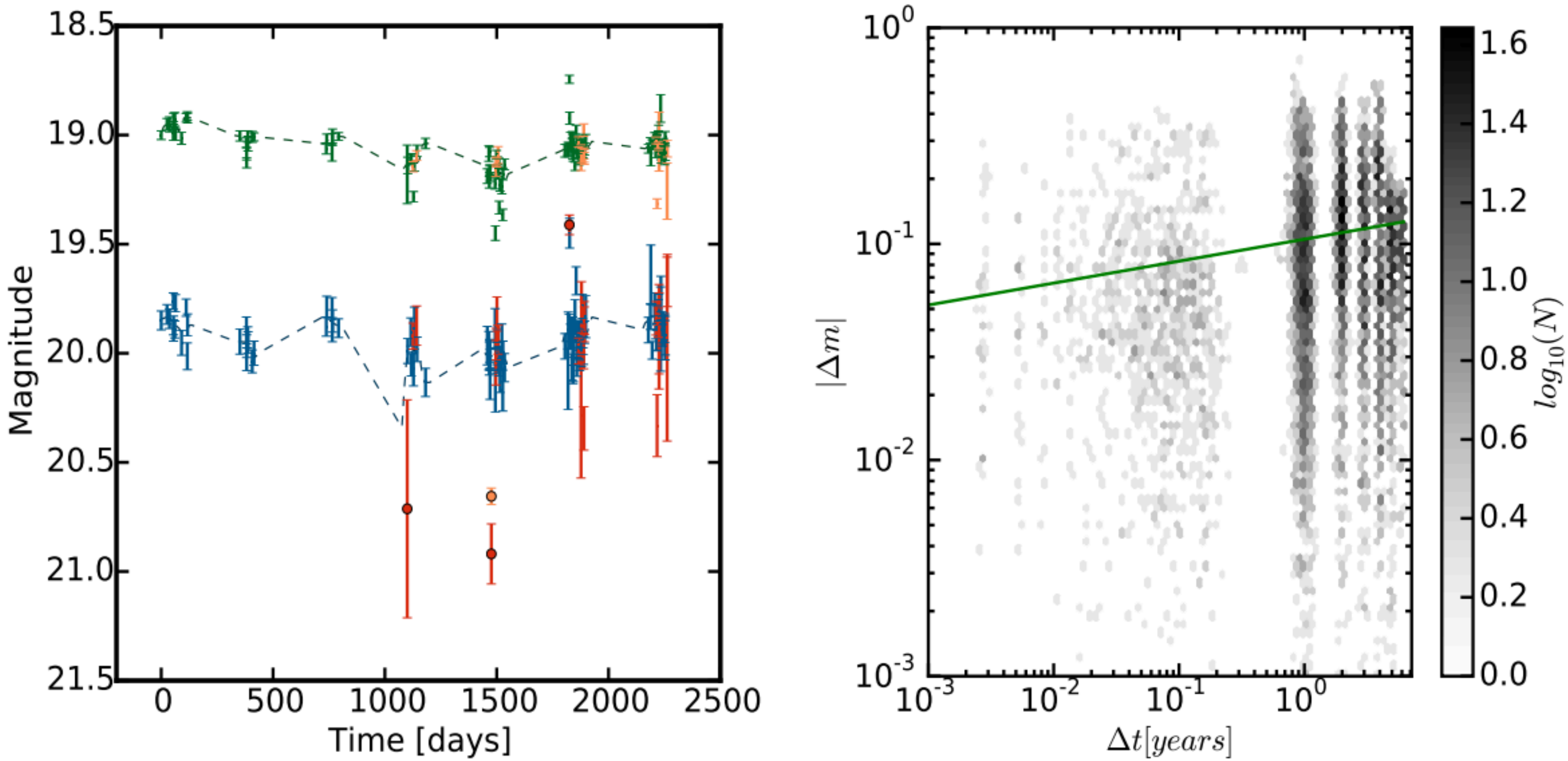


Figure 2 in Peters et al. Quasar light curve and SF

Summarizing the SF

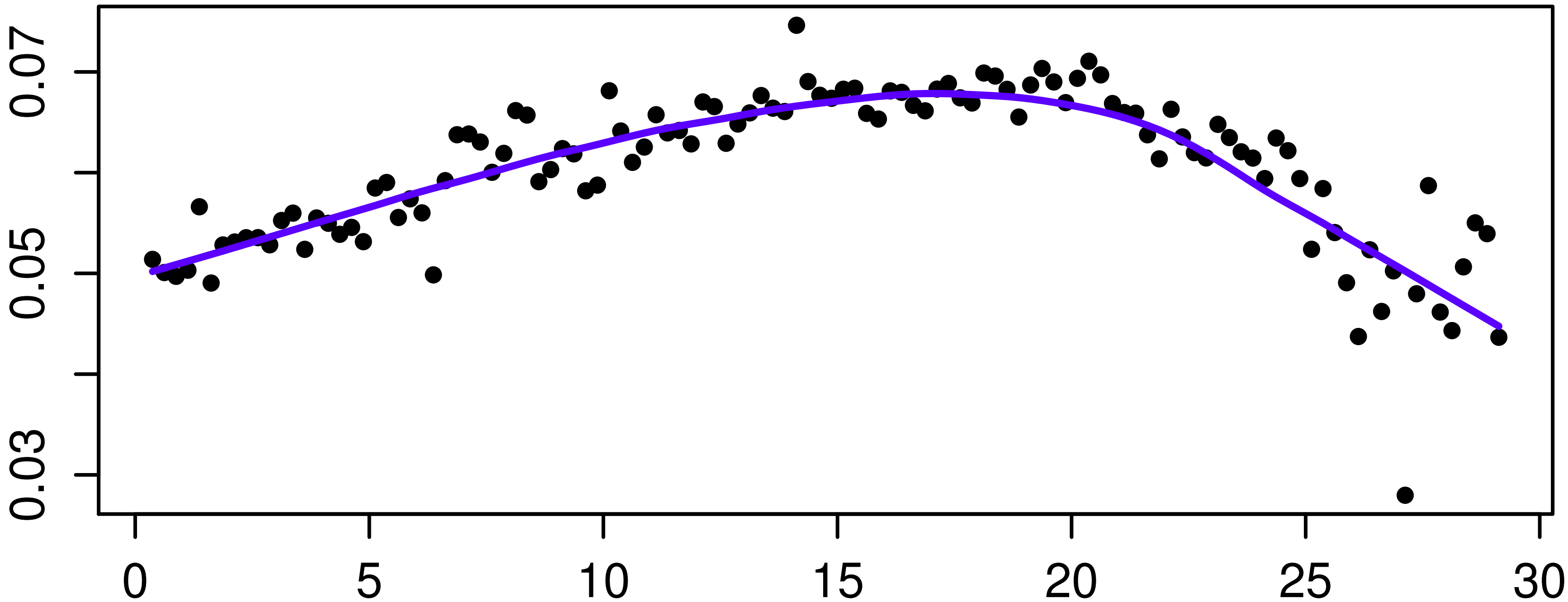
Typical to fit model to structure function

- Power Law Form (Schmidt et al.)
- Damped Random Walk (Kelly et al.)

Effort to find a low-dimensional representation, avoiding the curse of dimensionality

Ideally, could utilize higher-dimensional representation

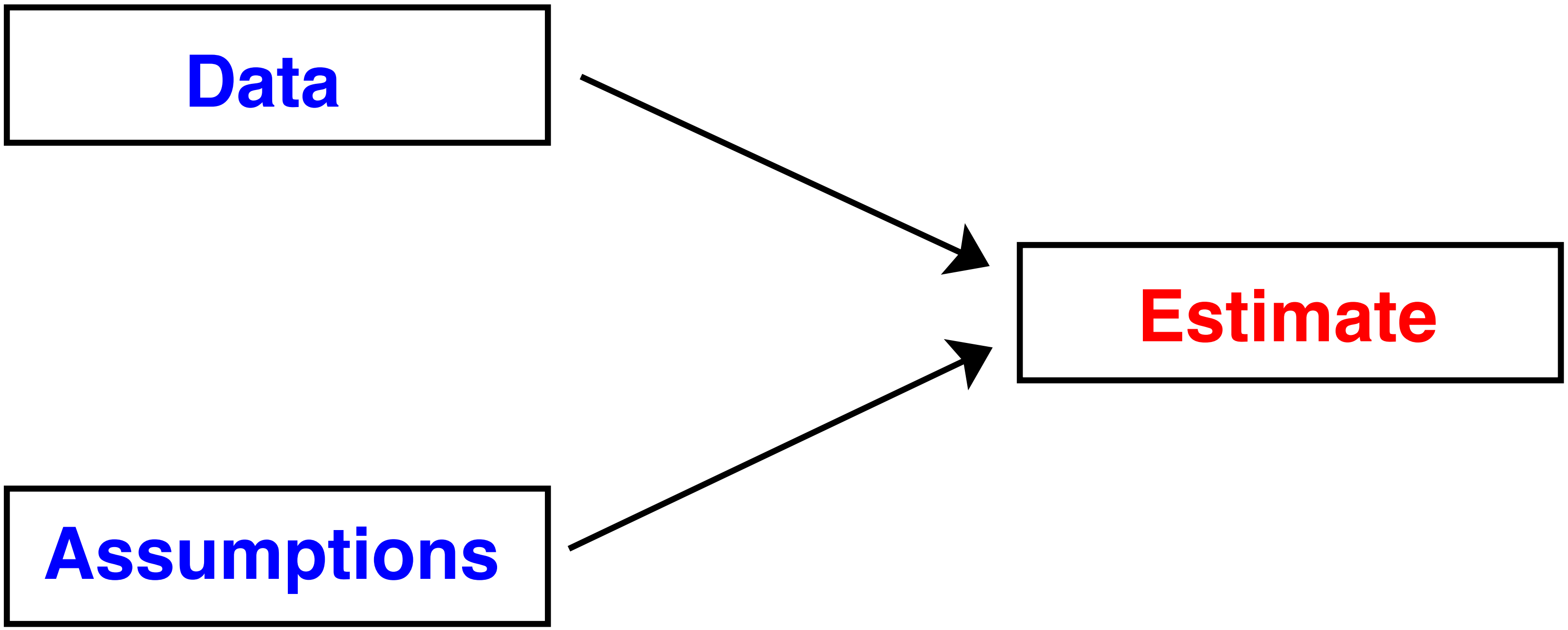
Nonparametric Regression



Important smoothing technique

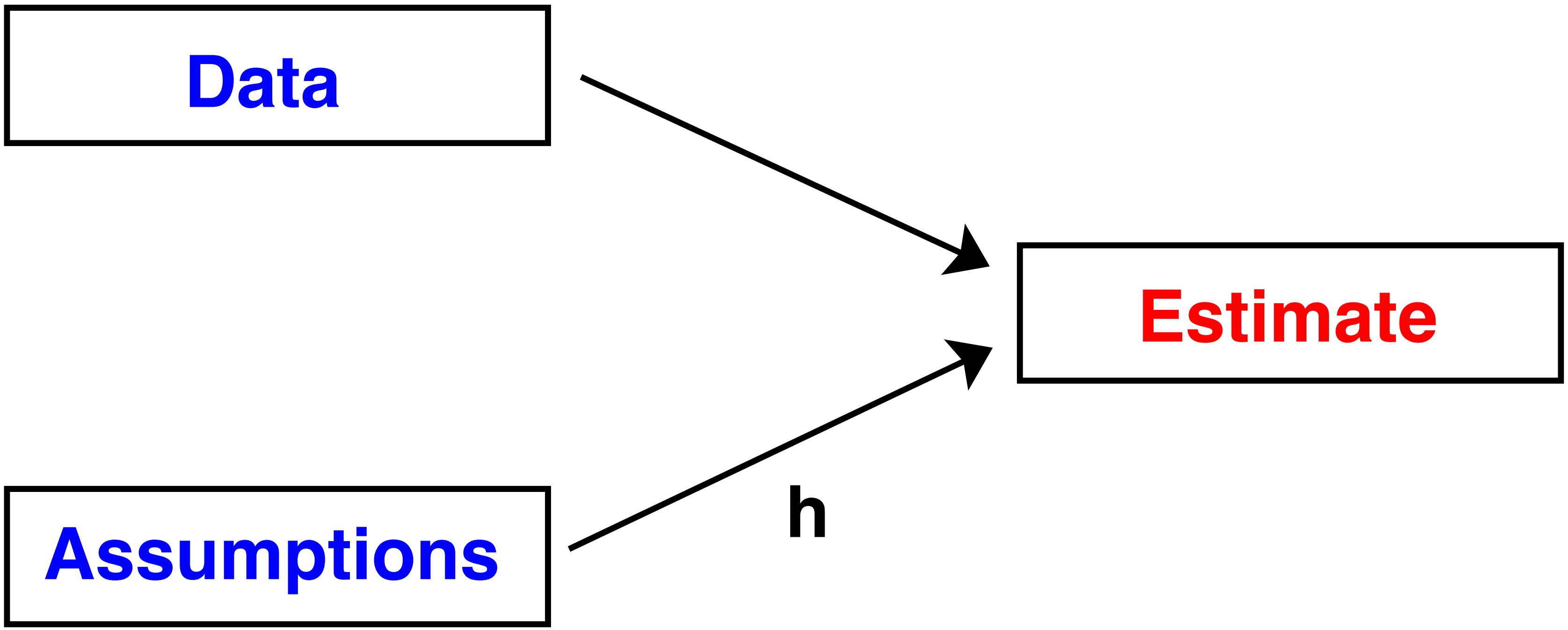
Regression as a summary tool

What is Nonparametric?



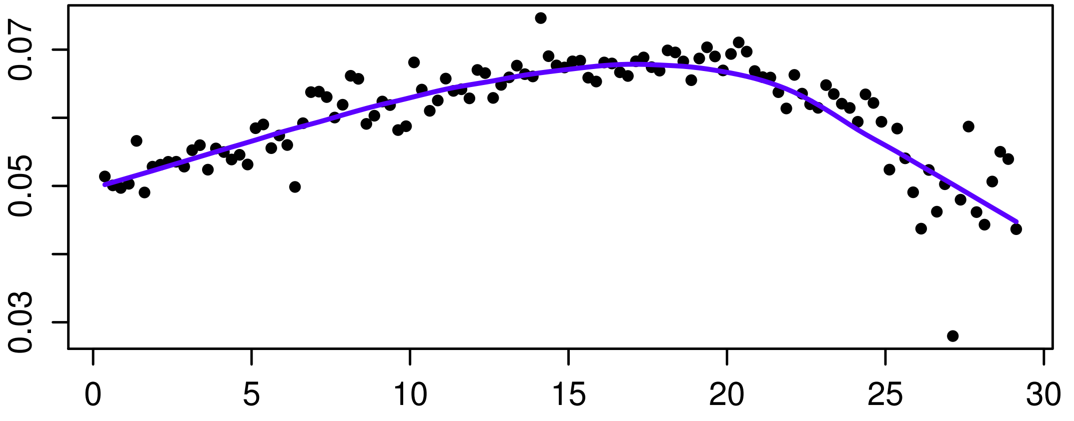
In the parametric case, the influence of assumptions is fixed

What is Nonparametric?



In the nonparametric case, the influence of assumptions is controlled by smoothing parameter h which shrinks with more data

Nonparametric Regression

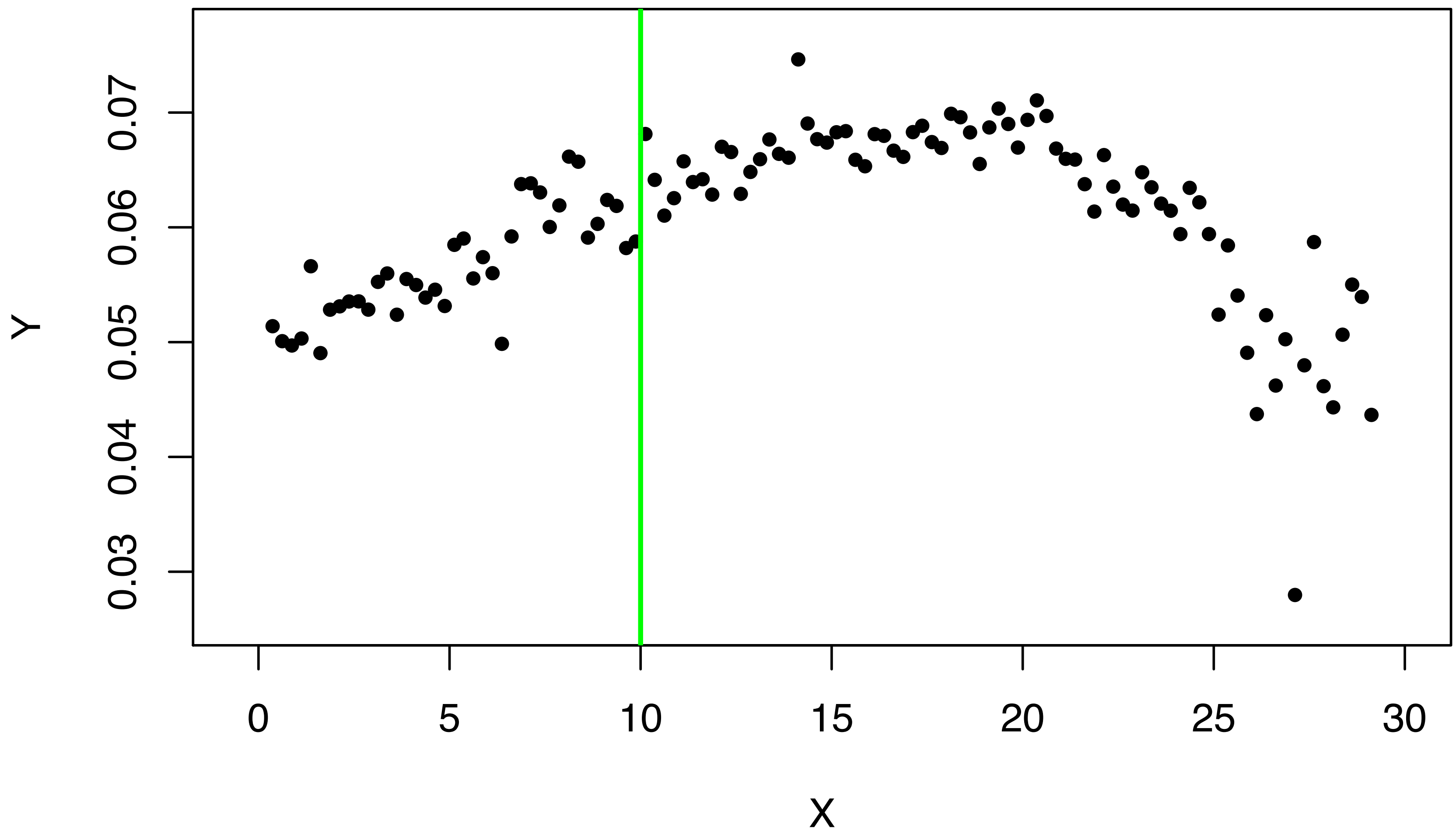


Here we consider local linear regression

- Fits a sequence of local linear models
- Each local model only fits within a neighborhood. Size of neighborhood is the smoothing parameter

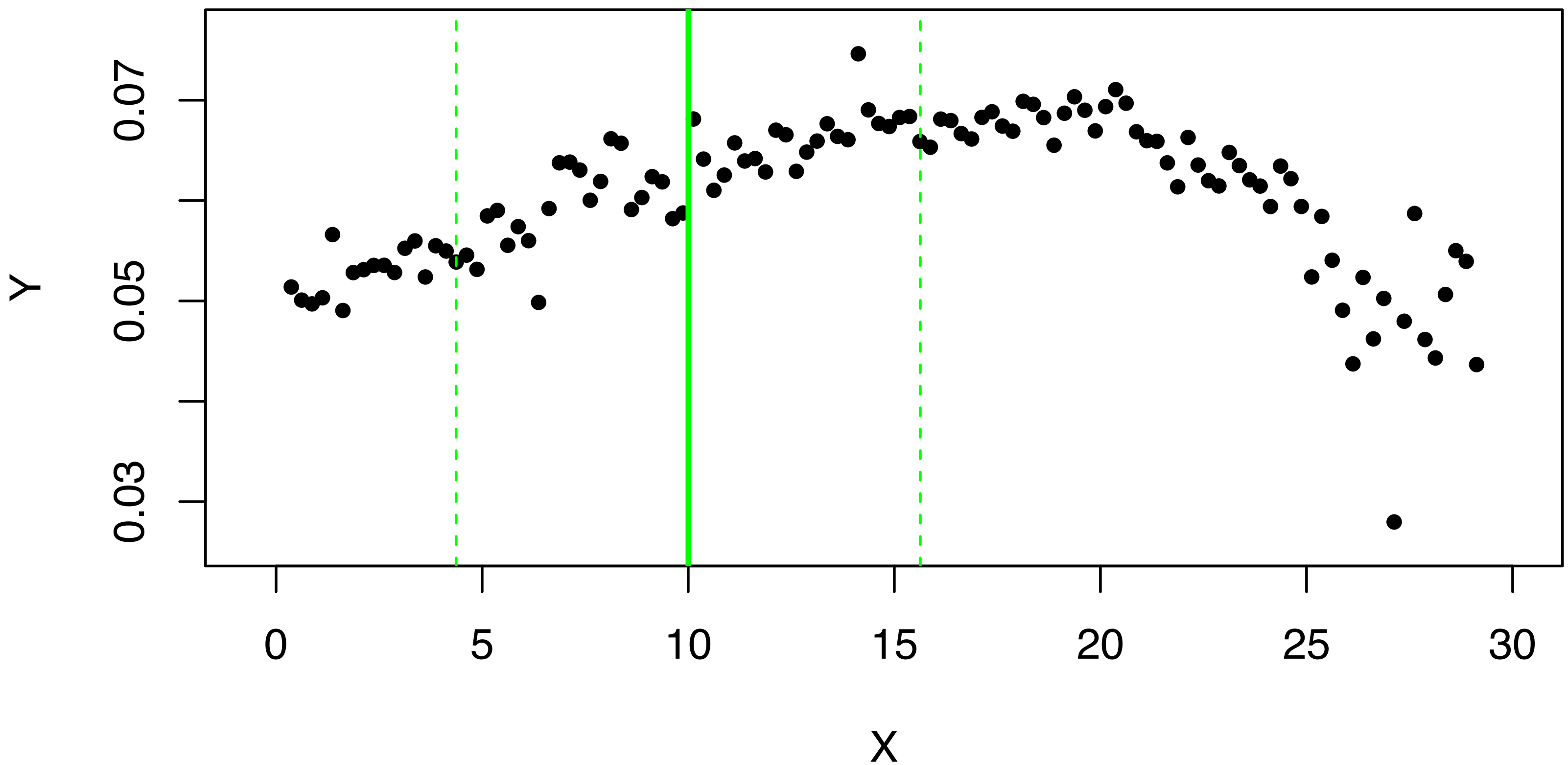
Step One: Fix the target point x_0.

Our objective is to estimate the regression function at x_0.



Step Two: Create the neighborhood around x_0.

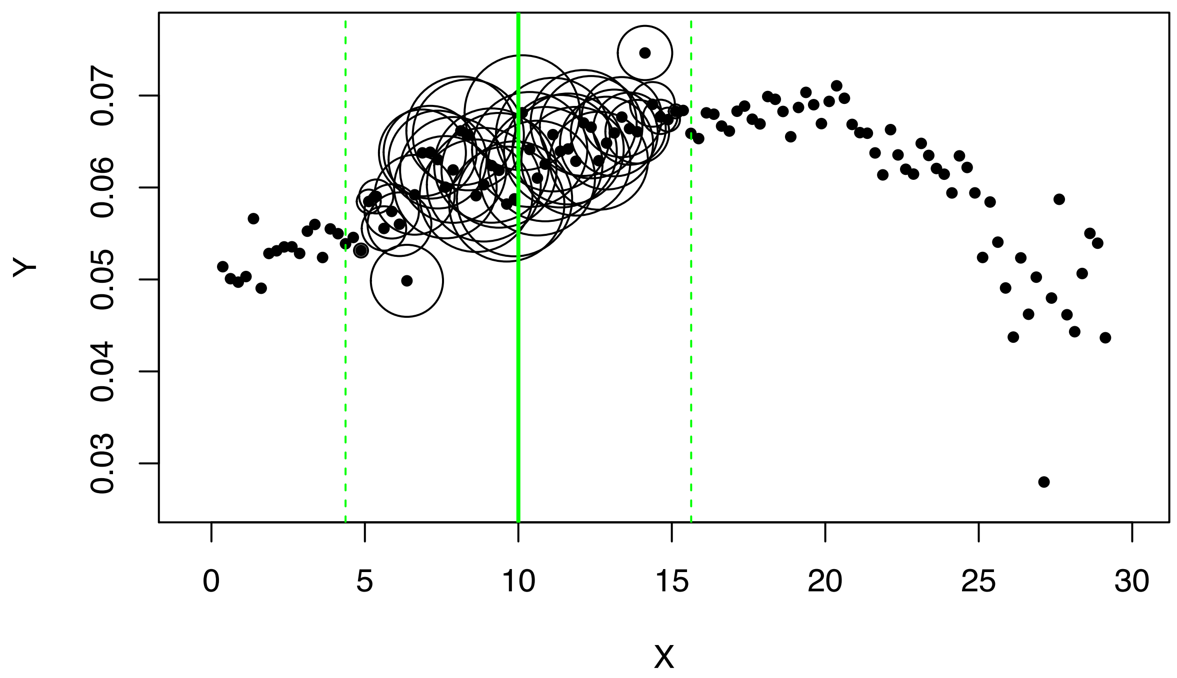
A common way to choose the neighborhood size is to choose is large enough to capture proportion α of the data. This parameter α is often called the span. A typical choice is $\alpha \approx 0.5$.



Step Three: Weight the data in the neighborhood.

Values of x which are close x_0 will receive a larger weight than those far from x_0. Denote by w_i the weight placed on observation i. The default choice is the tri-cube weight function:

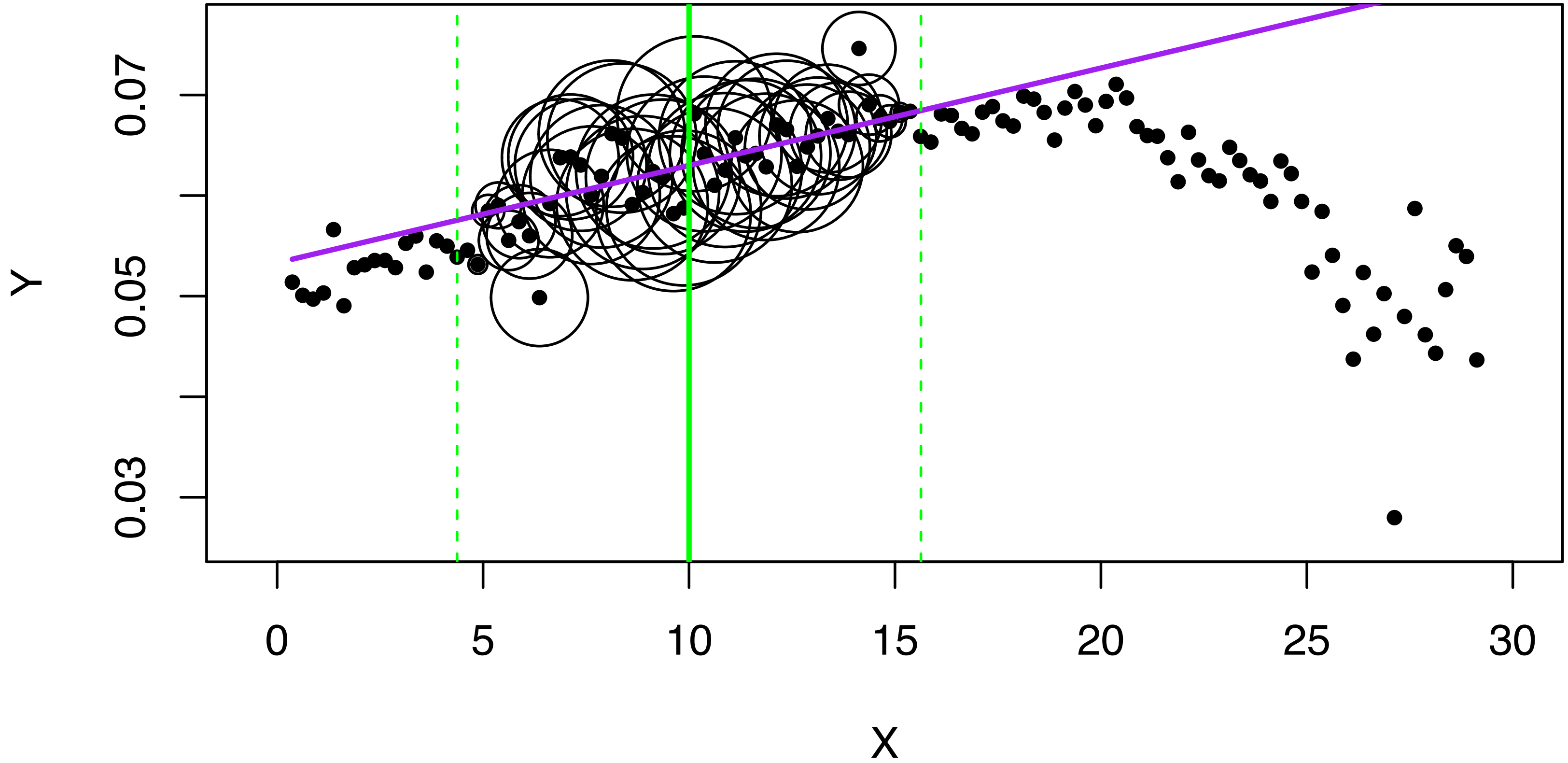
$$w_i = \begin{cases} \left(1 - \left|\frac{x_i - x_0}{\text{max dist}}\right|^3\right)^3, & \text{if } x_i \text{ in the neighborhood of } x_0 \\ 0, & \text{if } x_i \text{ is not in neighborhood of } x_0 \end{cases}$$



Step Four: Fit the local regression line.

This is done by finding β_0 and β_1 to minimize the weighted sum of squares

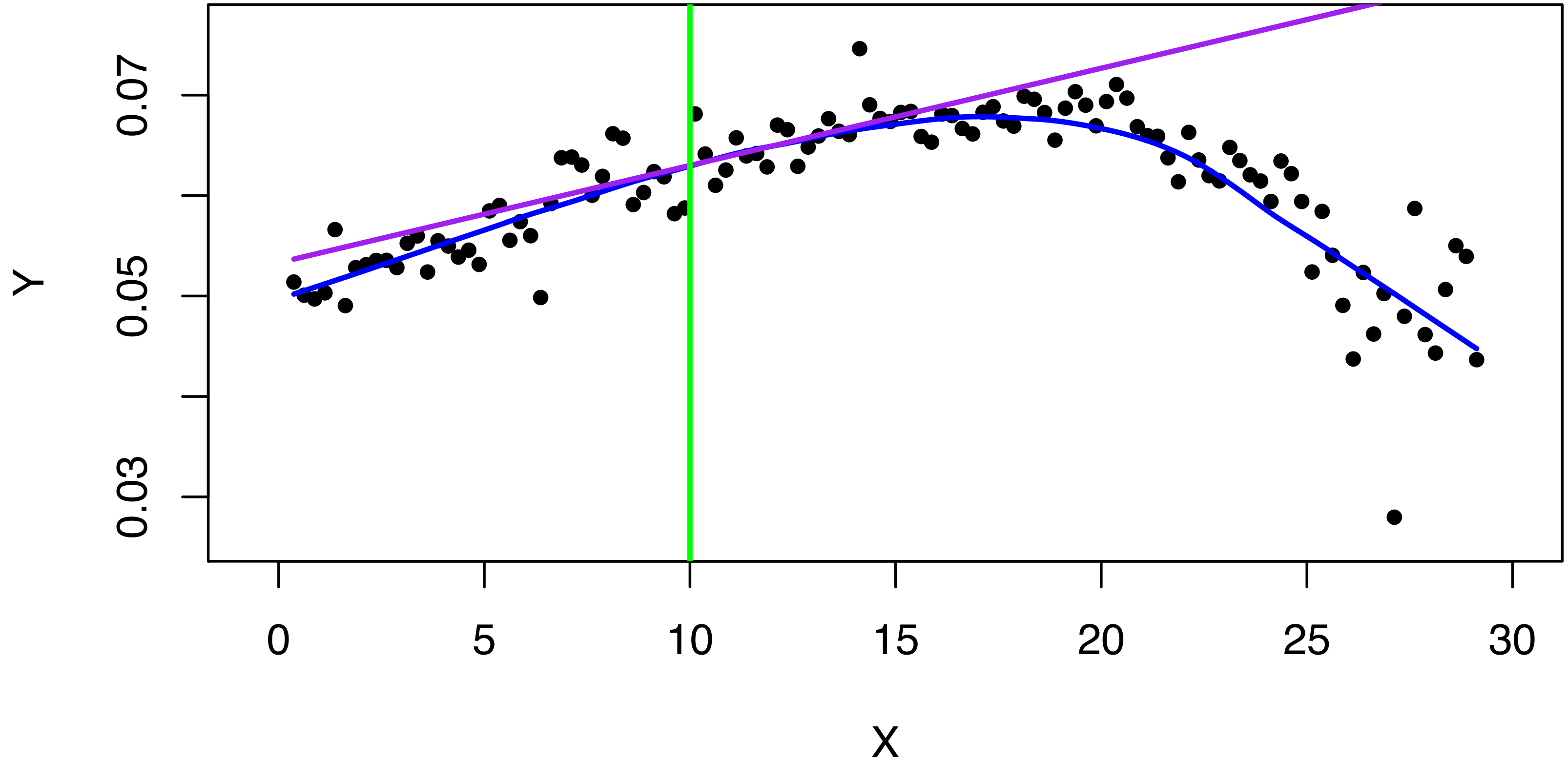
$$\sum_{i=1}^{n} w_i \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$



Step Five: Estimate $f(x_0)$.

This is done using the fitted regression line to estimate the regression function at x_0:

$$\widehat{f}(x_0) = \widehat{\beta}_0 + \widehat{\beta}_1 x_0$$



Bias-Variance Tradeoff

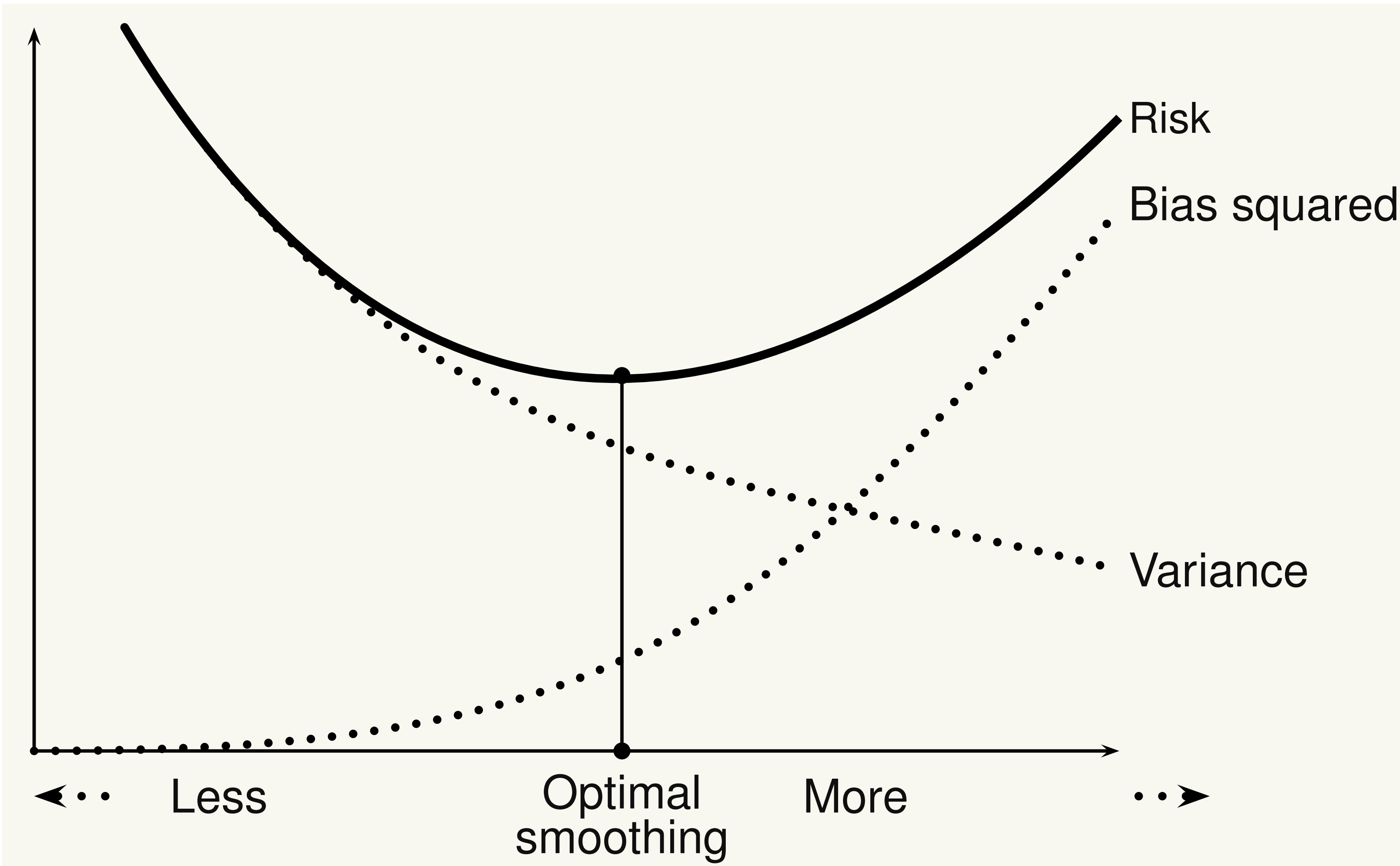
Choose smoothing parameter to achieve balance between

- Too simple – high bias: precise but not accurate
- Too complex – high variance: accurate, but not precise

This is the classic bias-variance tradeoff

Could be called the accuracy-precision tradeoff

Bias-Variance Tradeoff



Bias-Variance Tradeoff

Estimate the risk

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}(\widehat{f}(X_i) - f(X_i))^2$$

with the **leave-one-out cross-validation score**:

$$CV = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{f}_{(-i)}(X_i))^2$$

where $\widehat{f}_{(-i)}$ is the estimator obtained by omitting the i^{th} pair (X_i, Y_i).

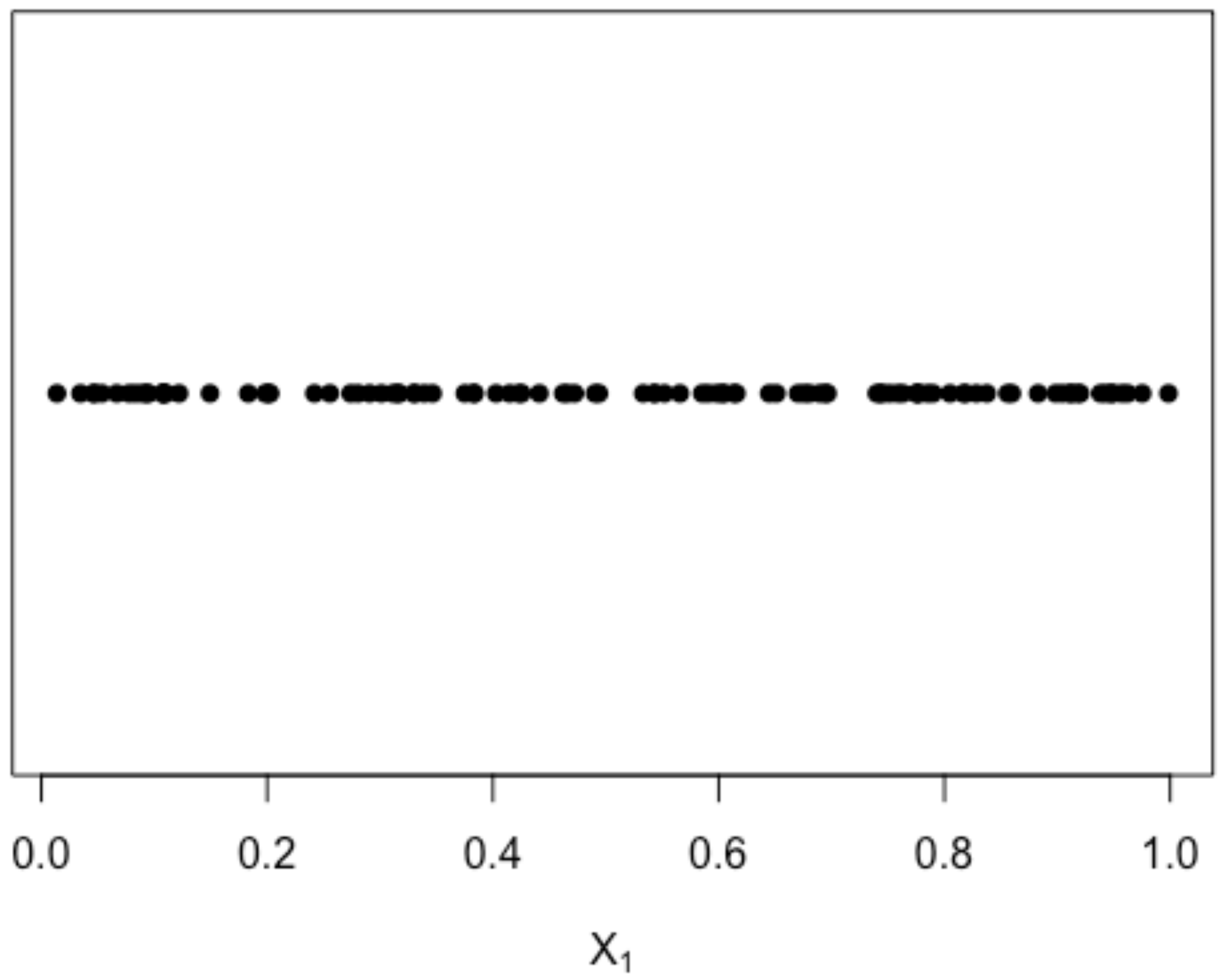
Curse of Dimensionality

Despite the promise of nonparametrics, fitting models in high dimensions is a challenge

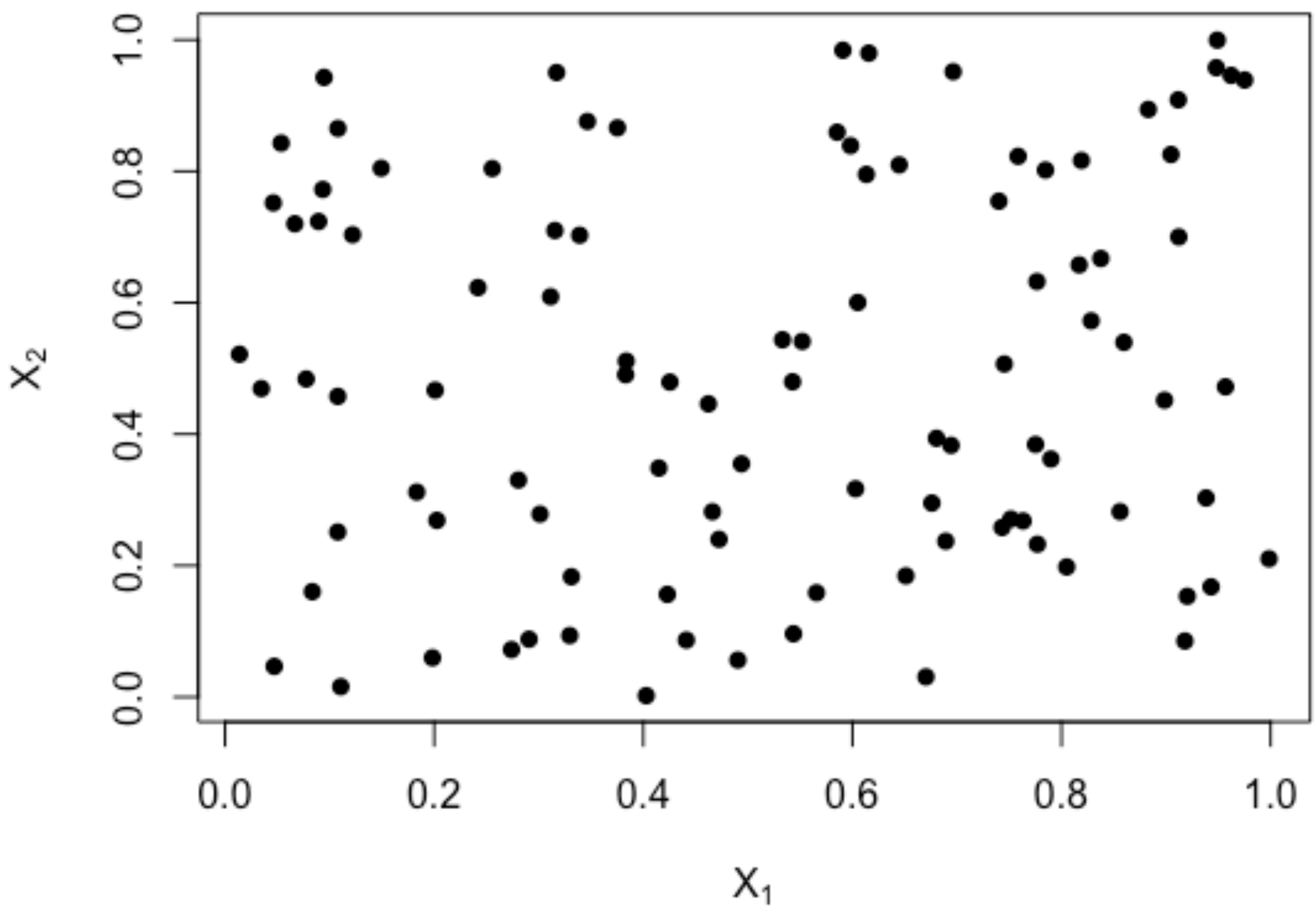
These fits require ample data in the "neighborhood" to be reliable, and data become sparse in high dimensions

Choosing neighborhoods larger reduces the value of the approach

Curse of Dimensionality



Curse of Dimensionality



Curse of Dimensionality

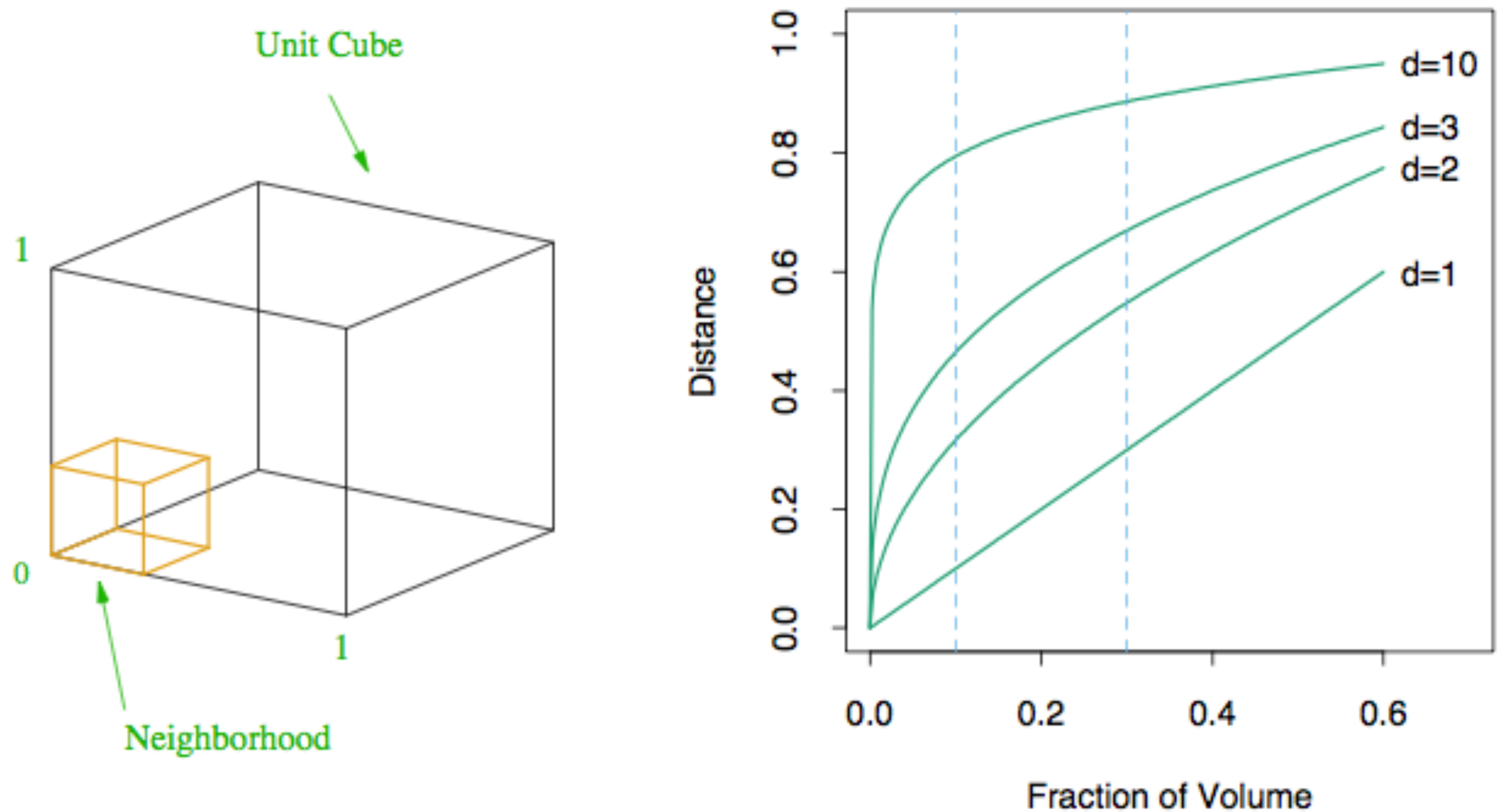
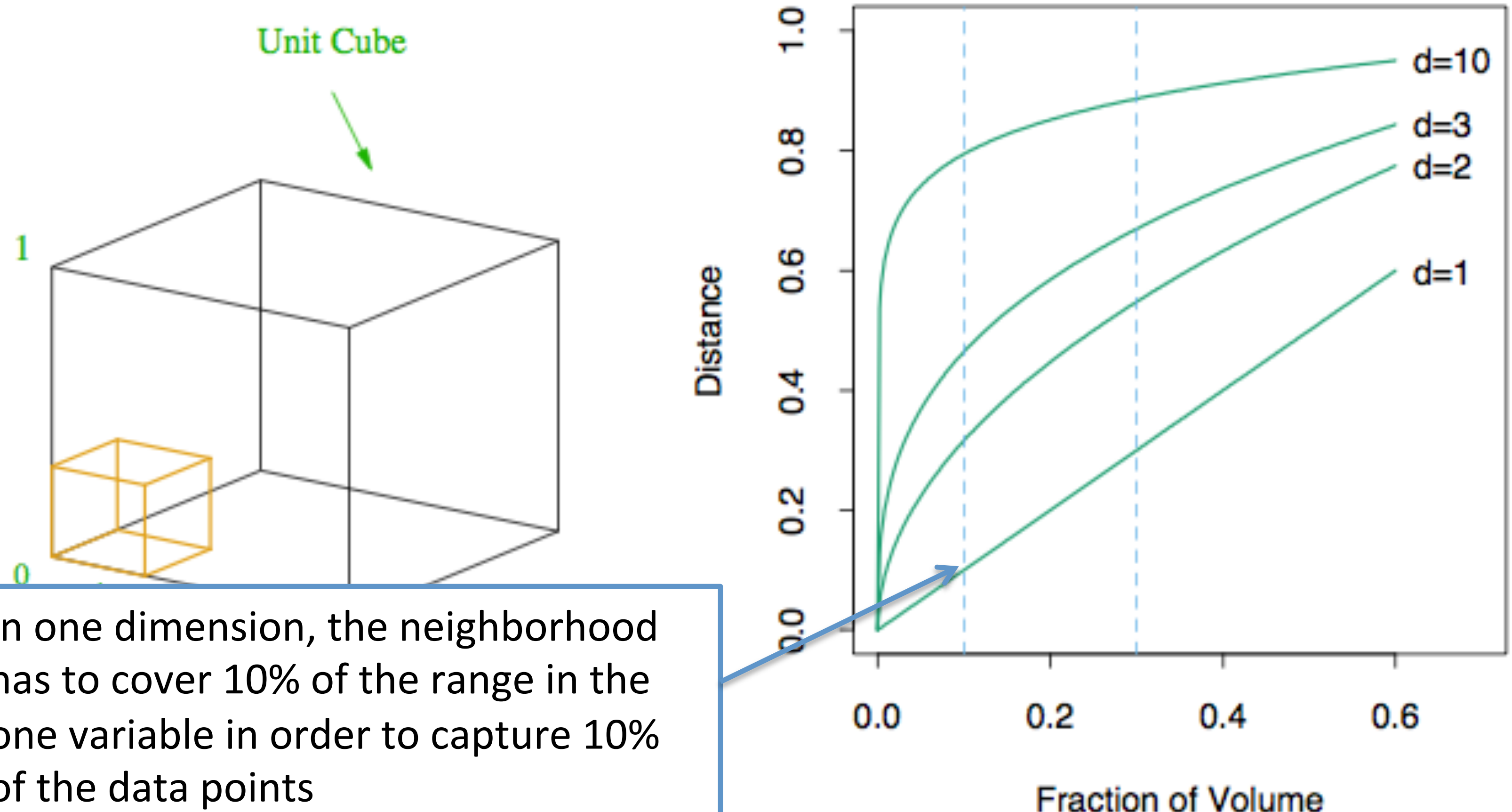


Figure 2.6 from Hastie, Tibshirani, and Friedman

Curse of Dimensionality



In one dimension, the neighborhood has to cover 10% of the range in the one variable in order to capture 10% of the data points

Figure 2.6 from Hastie, Tibshirani, and Friedman

Curse of Dimensionality

In ten dimensions, the neighborhood has to cover 80% of the range in each variable in order to capture 10% of the data points

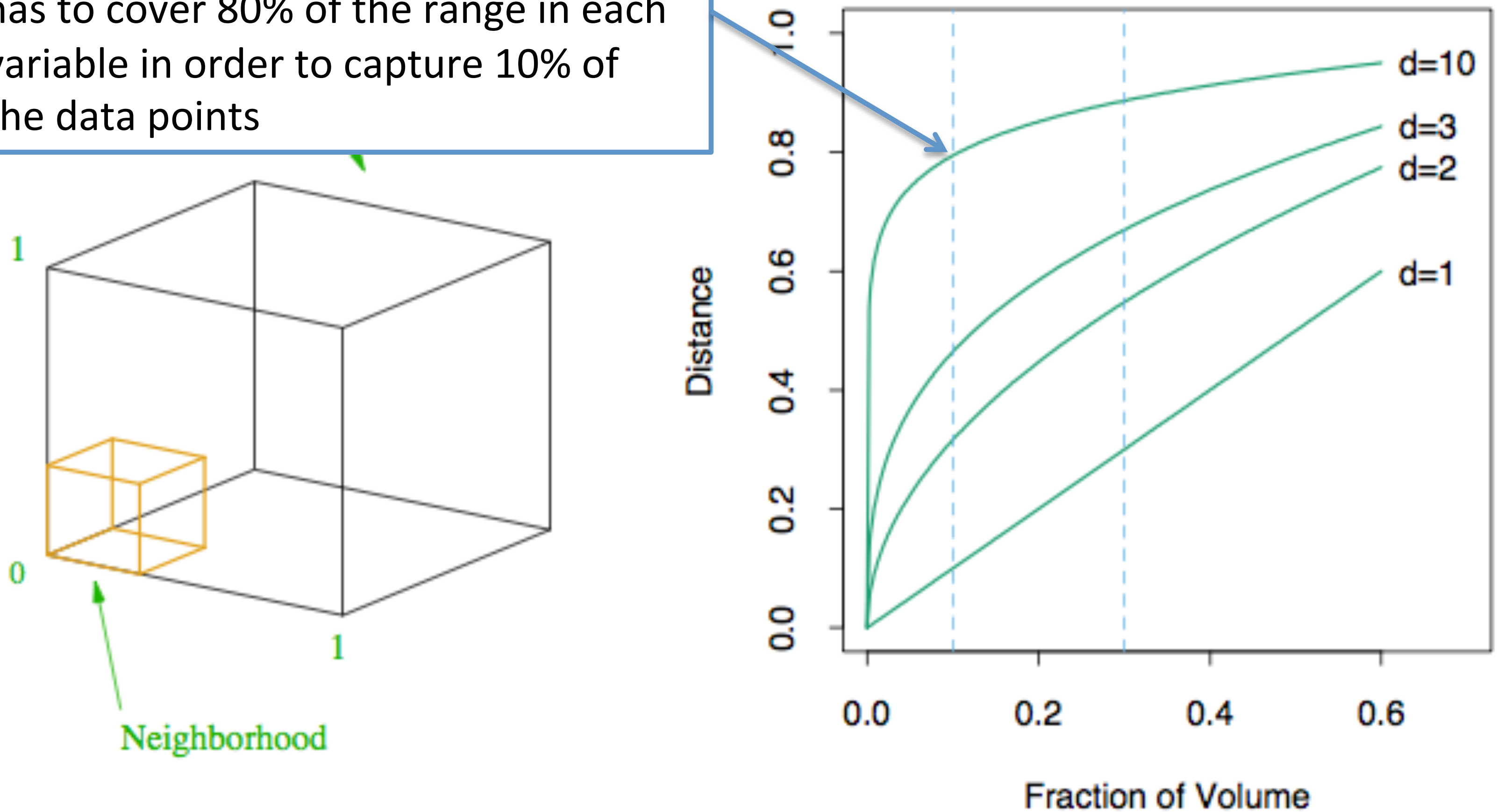


Figure 2.6 from Hastie, Tibshirani, and Friedman

Additive Models

Additive models avoid the curse of dimensionality by making a strong, but not overly restrictive assumption regarding the relationship between the response and predictors

Additive Models

The fully nonparametric model:

$$Y = f(x_1, x_2, \ldots, x_p) + \epsilon$$

with the f **estimated from the data**.

The additive nonparametric model:

$$Y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \epsilon$$

with each of the f_i **estimated from the data**. (Each f_i is shifted so that it is centered around zero. The intercept β_0 accounts for the overall mean of the response.)

Additive Models

The general estimation strategy is called **backfitting**.

In this process, each f_k is estimated nonparametrically, in a rotation, and the process is repeated until there is convergence.

When estimating $f_k(\cdot)$, the other $f_j(\cdot)$ are held fixed at their current best estimates, and we set up a one-dimensional nonparametric estimation problem, on which one could use either local linear regression, smoothing splines, or other approach.

Additive Models

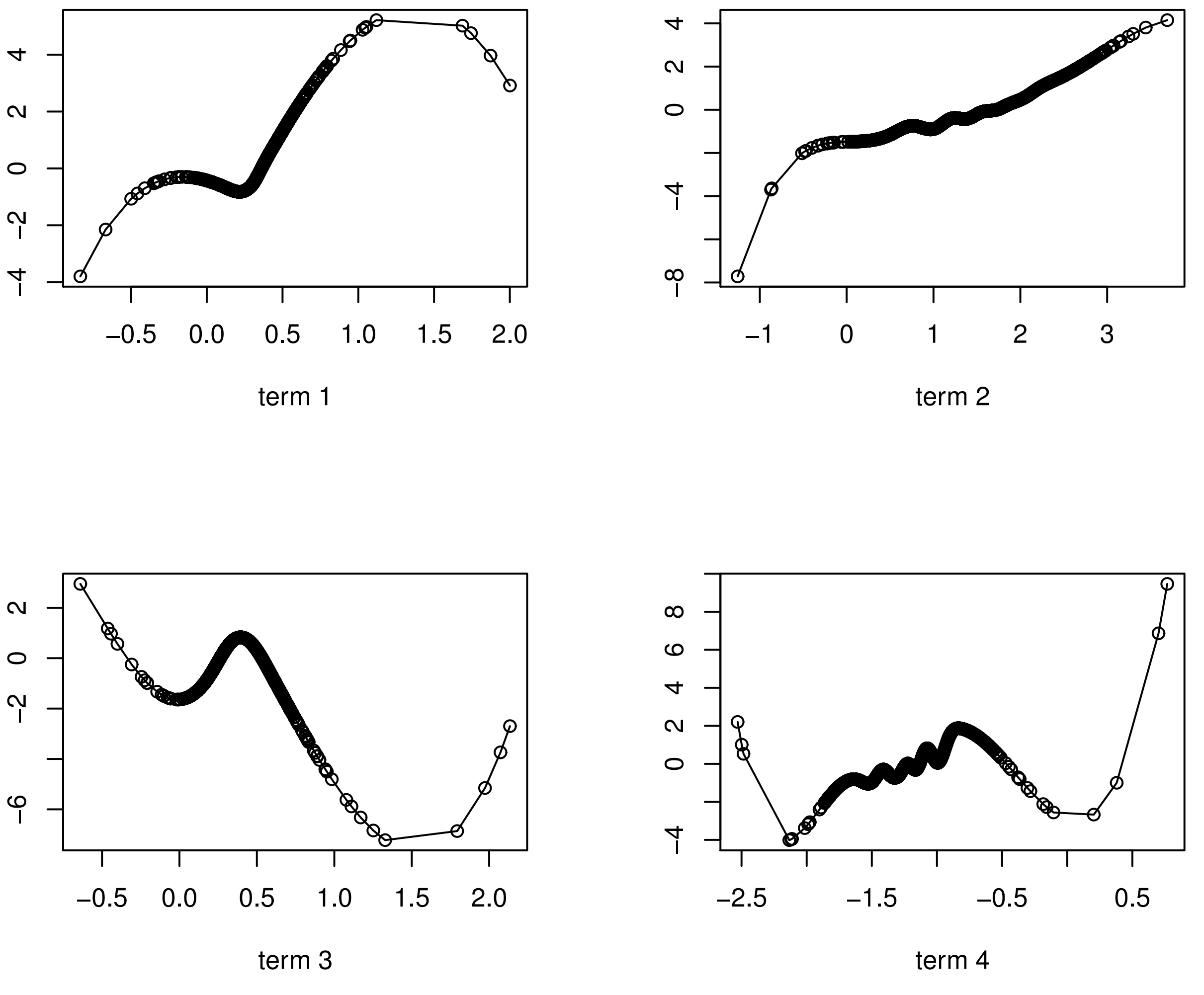
A generalization of this model is the **projection pursuit regression** model, which can be written as

$$Y = \beta_0 + \sum_{k=1}^{M} \beta_k f_k\left(\boldsymbol{\alpha}_k^T \mathbf{x}\right) + \epsilon$$

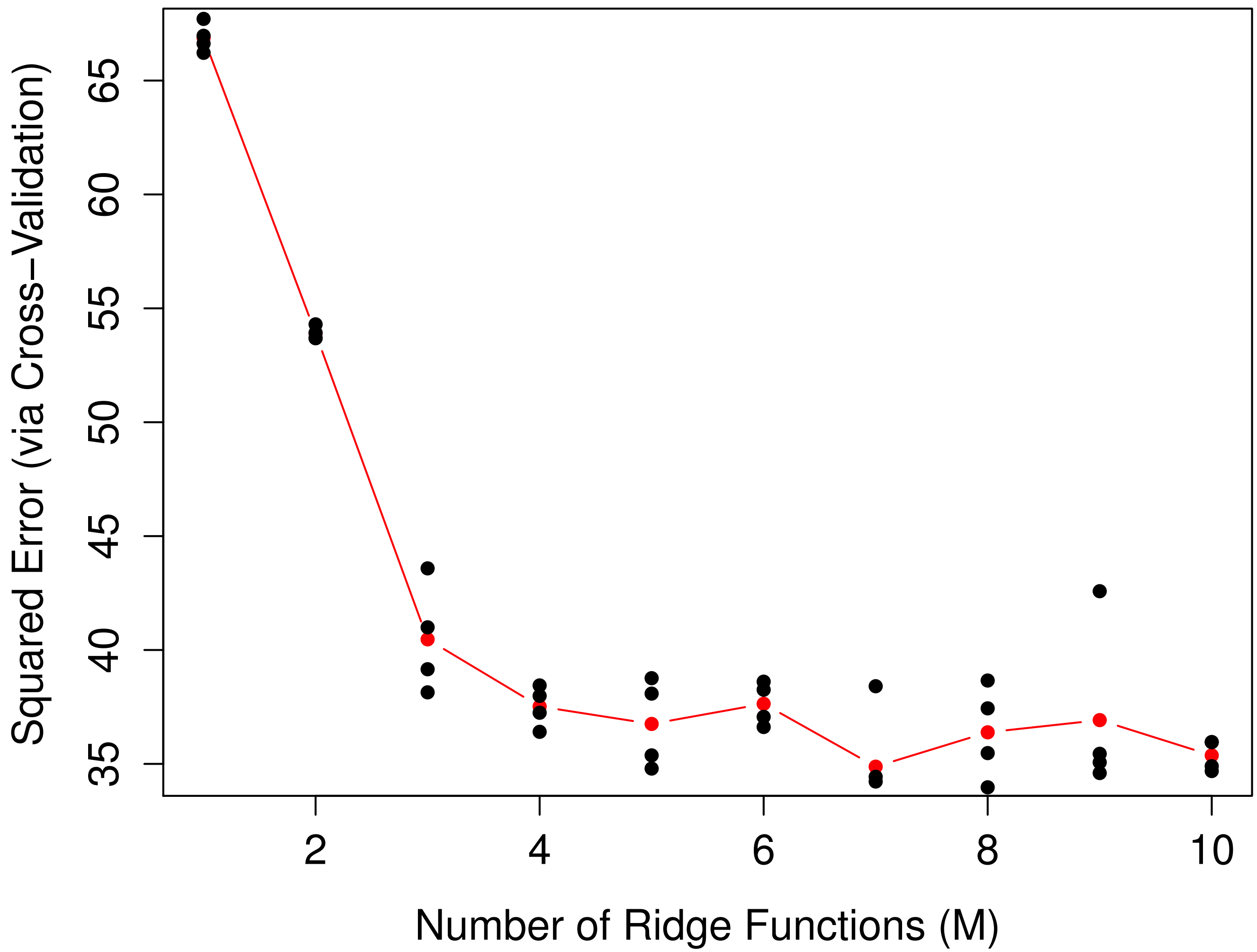
where each of the $\boldsymbol{\alpha}_k$ are a vector of length p. These $\boldsymbol{\alpha}_k$ are the **projection direction vectors**.

The functions f_k, called the **ridge functions**, are estimated nonparametrically. These functions are scaled to have mean zero and variance one when applied to the observed sample.

Additive Models



Additive Models



Neural Networks

The term **neural network** has evolved to encompass a large class of models and learning methods. Here we describe the most widely used "vanilla" neural net, sometimes called the **single hidden layer back-propogation network,** or **single layer perceptron**. There has been a great deal of *hype* surrounding neural networks, making them seem magical and mysterious. As we make clear in this section, they are just nonlinear statistical models, much like the projection pursuit regression model discussed above.

Quote from Hastie, Tibshirani, and Friedman

Neural Networks

The single hidden layer back-propogation network regression model can be written as follows:

$$Y = \beta_0 + \sum_{k=1}^{M} \beta_k \phi\big(\alpha_{0k} + \boldsymbol{\alpha}_k^T \mathbf{x}\big) + \epsilon$$

where the β and α are parameters to be estimated, but the function ϕ is **not** estimated.

Neural Networks

Some terms commonly used in conjunction with neural networks:

- The function ϕ is called the **activation function**. The standard choice is the **sigmoid function**

$$\phi(u) = \frac{1}{1 + \exp(-u)}.$$

- The elements $\phi\big(\alpha_{0k} + \boldsymbol{\alpha}_k^T \mathbf{x}\big)$ for $k = 1, 2, \ldots, M$ comprise the **hidden layer**.

- The intercept terms α_{0k} are called the **biases**.

- The entire collection of α and β parameters are called the **weights**.

Neural Networks

If least squares is used alone, however, the solution is unstable, and the nonconvex optimization problem that is solved is sensitive to the starting values used in the iterative search algorithm.

Hence, a **regularization penalty** is often added onto the residual sum of squares. A standard choice is the same penalty used in ridge regression, i.e., minimize

$$\mathrm{RSS} + \lambda \sum_{k=1}^{M} \left[\beta_k^2 + \sum_{i=0}^{p} \alpha_{ik}^2 \right].$$

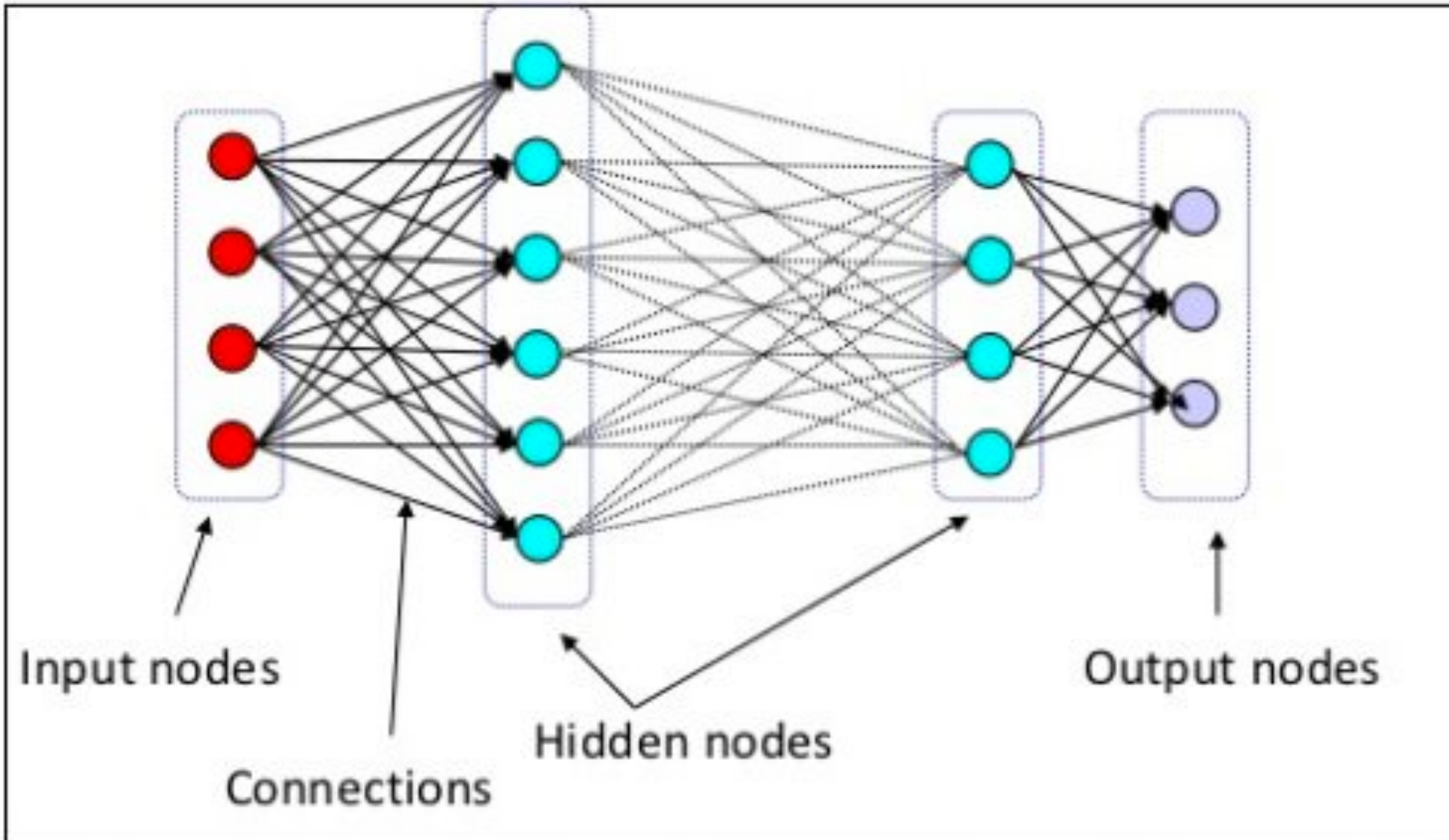
The parameter λ is commonly called the **decay parameter**.

Neural Networks

Of course, now we have two parameters that control the complexity of the model: M and λ. These should both be chosen carefully in order to avoid overfitting. Like other nonparametric estimation methods, neural networks have trememdous capacity to overfit to the observed data.

We will use k-fold cross-validation to choose the tuning parameters M and λ (the decay parameter).

Adding Layers



Deep Learning

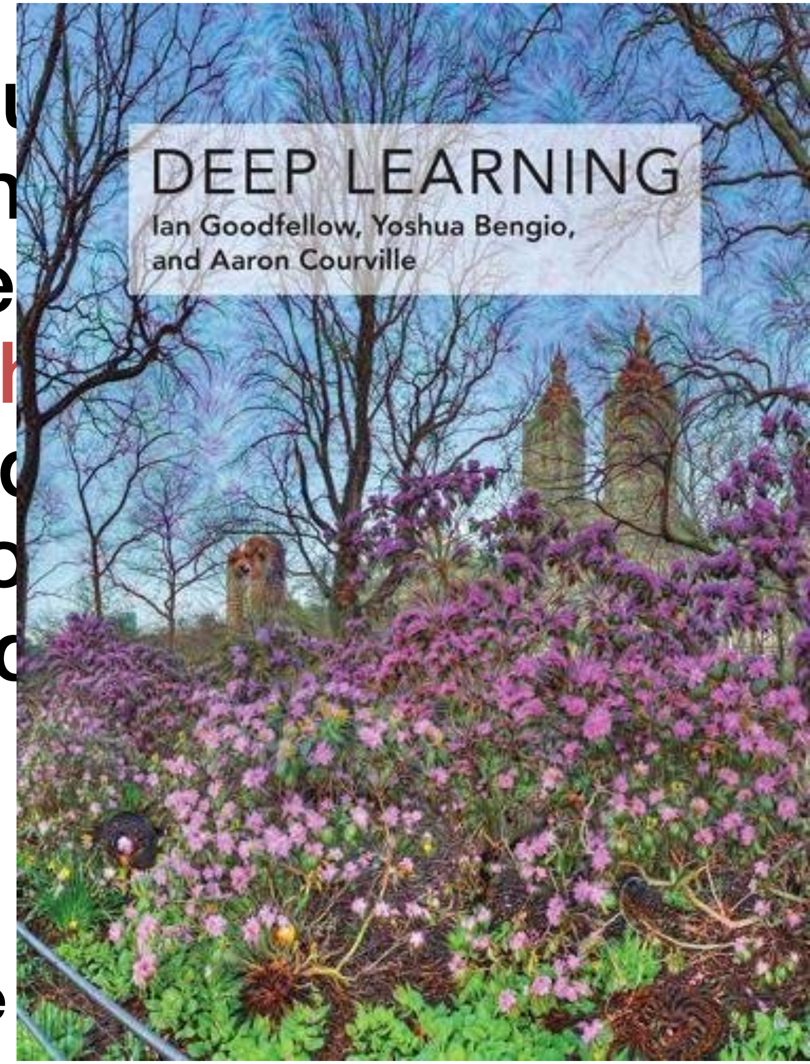
“Deep learning is a particular kind of machine learning that achieves great power and flexibility by representing the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.”

--Page 8 in *Deep Learning,*
Goodfellow, Bengio, and Courville

Deep Learning

“Deep learning is a partic
machine learning that ach
power and flexibility by re
world as a nested hierarch
with each concept definec
simpler concepts, and mo
representations computec
less abstract ones.”

--Page 8 in *Deep Learning,*
Goodfellow, Bengio, and Courville



www.deeplearningbook.org

Deep Learning

What makes it “deep?”

Deep Learning

What makes it "deep?"

The number of hidden layers is typically large, allowing for the modeling of complex relationships.

This can be viewed as an extension/resurrection of neural networks

Resurgence of NN

Multiple factors contributed to growth of interest in Deep Learning:

- Increase in training set sizes
- Improved algorithms for training deeper networks (e.g., Hinton, et al. in 2006)
- Growth in computational resources
- Successes

Flexibility

A primary appeal of the approach is the flexibility in constructing the layers

- How many units are there in each layer?
- What is the mapping from one layer to the next?
- How is the output constructed from the final hidden layer?

Flexibility

A primary appeal of the approach is the flexibility in constructing the layers

- How many units are there in each layer?
- What is the mapping from one layer to the next?
- How is the output constructed from the final hidden layer?

Fully Connected Layer

A standard mapping is a fully connected layer, simply a linear combination of the input (either the data or the output of the preceding layer)

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$$b + \sum_{i=1}^{d} w_i x_i$$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$$b + \mathbf{w}^T\mathbf{x}$$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$$b_1 + \mathbf{w}_1^T\mathbf{x} \qquad b_2 + \mathbf{w}_2^T\mathbf{x}$$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

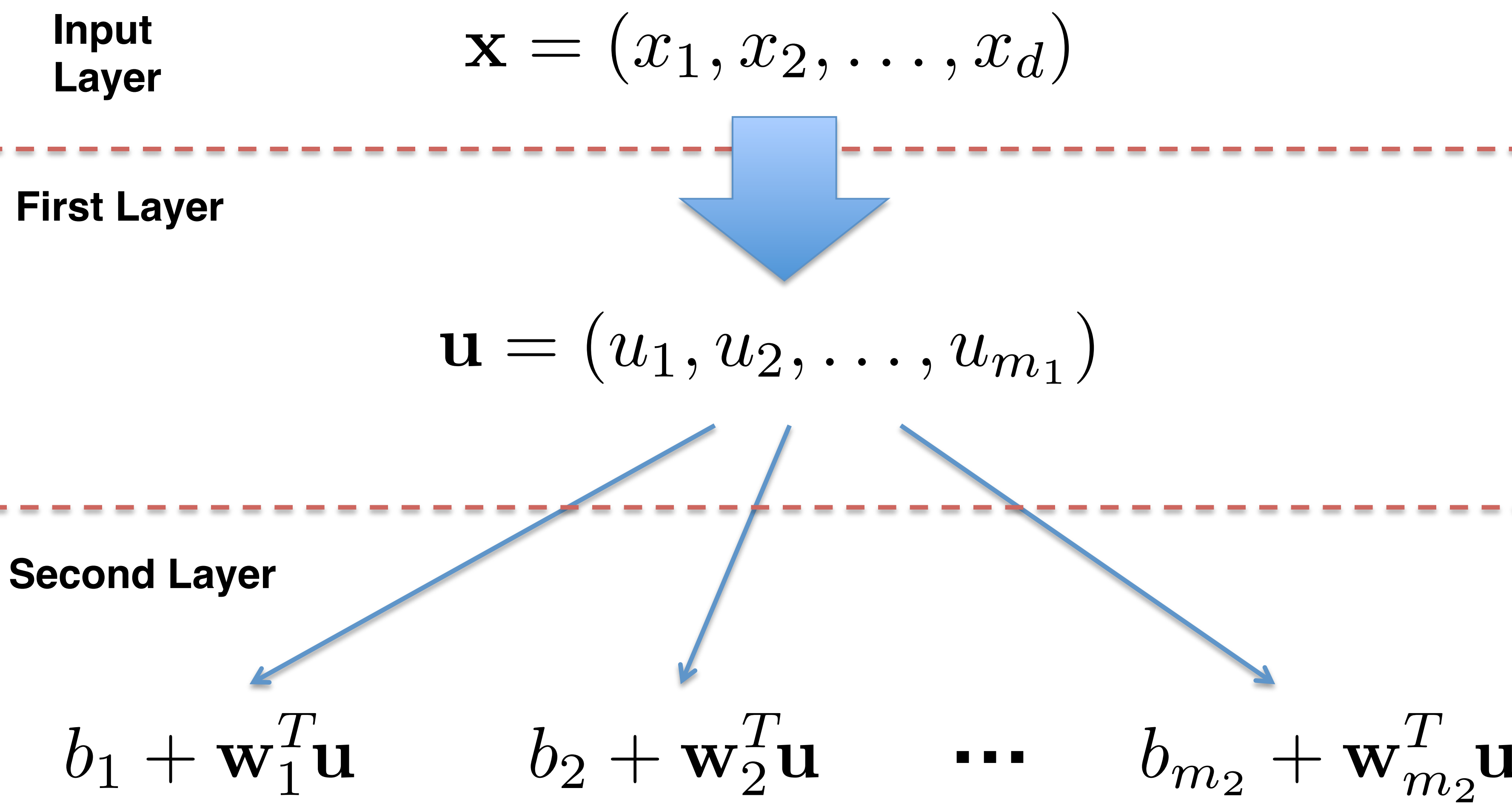
$$b_1 + \mathbf{w}_1^T\mathbf{x} \qquad b_2 + \mathbf{w}_2^T\mathbf{x} \qquad \cdots \qquad b_m + \mathbf{w}_m^T\mathbf{x}$$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

First Layer

$$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$$

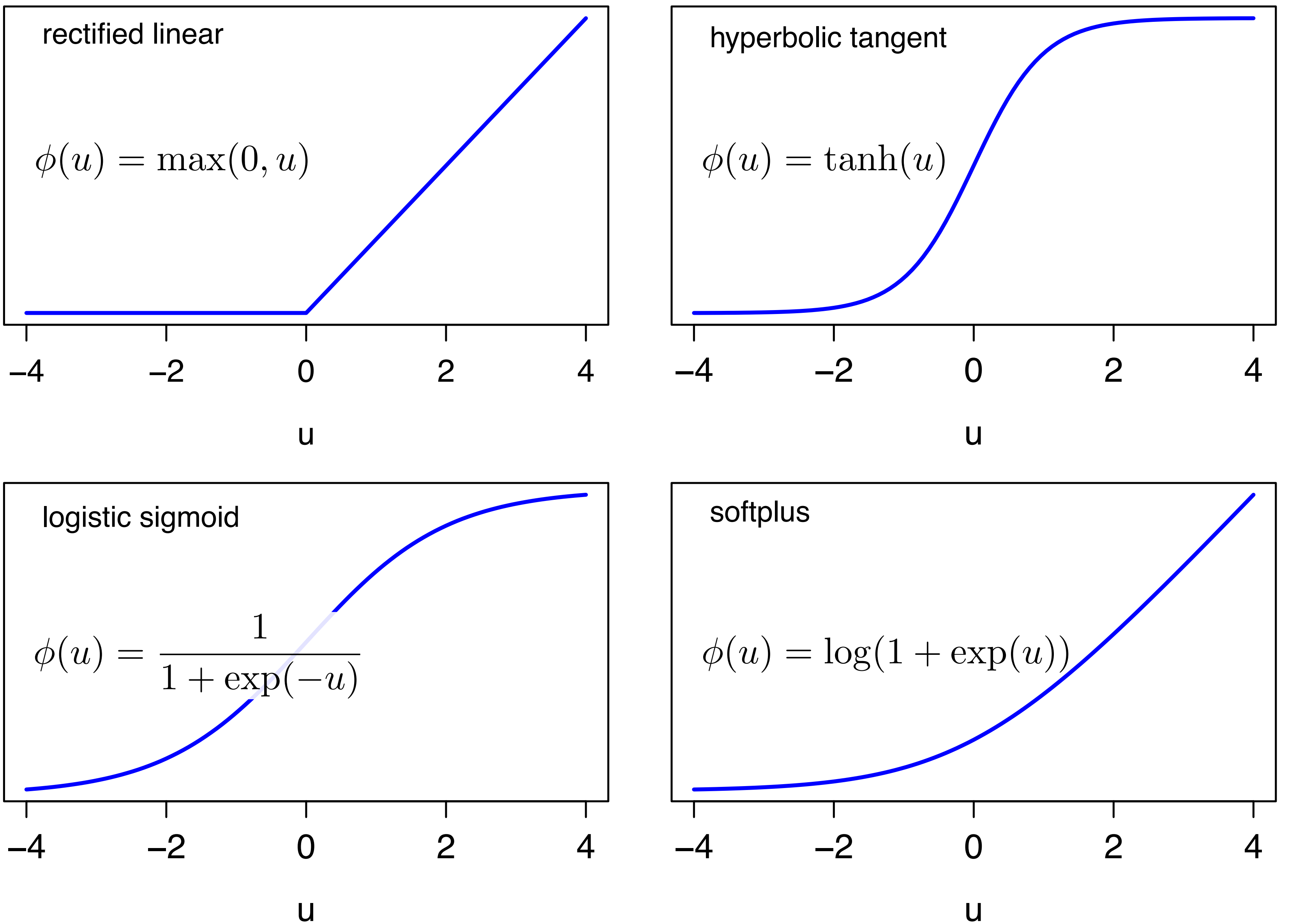

Input Layer
$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
First Layer
$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$
Second Layer
$b_1 + \mathbf{w}_1^T \mathbf{u}$
$b_2 + \mathbf{w}_2^T \mathbf{u}$
…
$b_{m_2} + \mathbf{w}_{m_2}^T \mathbf{u}$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$$\phi(b_1 + \mathbf{w}_1^T\mathbf{x}) \qquad \phi(b_2 + \mathbf{w}_2^T\mathbf{x}) \qquad \cdots \qquad \phi(b_m + \mathbf{w}_m^T\mathbf{x})$$

$\phi(\cdot)$ is the activation function, a simple nonlinear mapping


rectified linear
$\phi(u) = \max(0, u)$
hyperbolic tangent
$\phi(u) = \tanh(u)$
logistic sigmoid
$\phi(u) = \dfrac{1}{1 + \exp(-u)}$
softplus
$\phi(u) = \log(1 + \exp(u))$
−4
−2
0
2
4
u

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

$$\phi(b_1 + \mathbf{w}_1^T\mathbf{x}) \qquad \phi(b_2 + \mathbf{w}_2^T\mathbf{x}) \qquad \cdots \qquad \phi(b_m + \mathbf{w}_m^T\mathbf{x})$$

Input Layer

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

First Layer

$$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$$

Input Layer

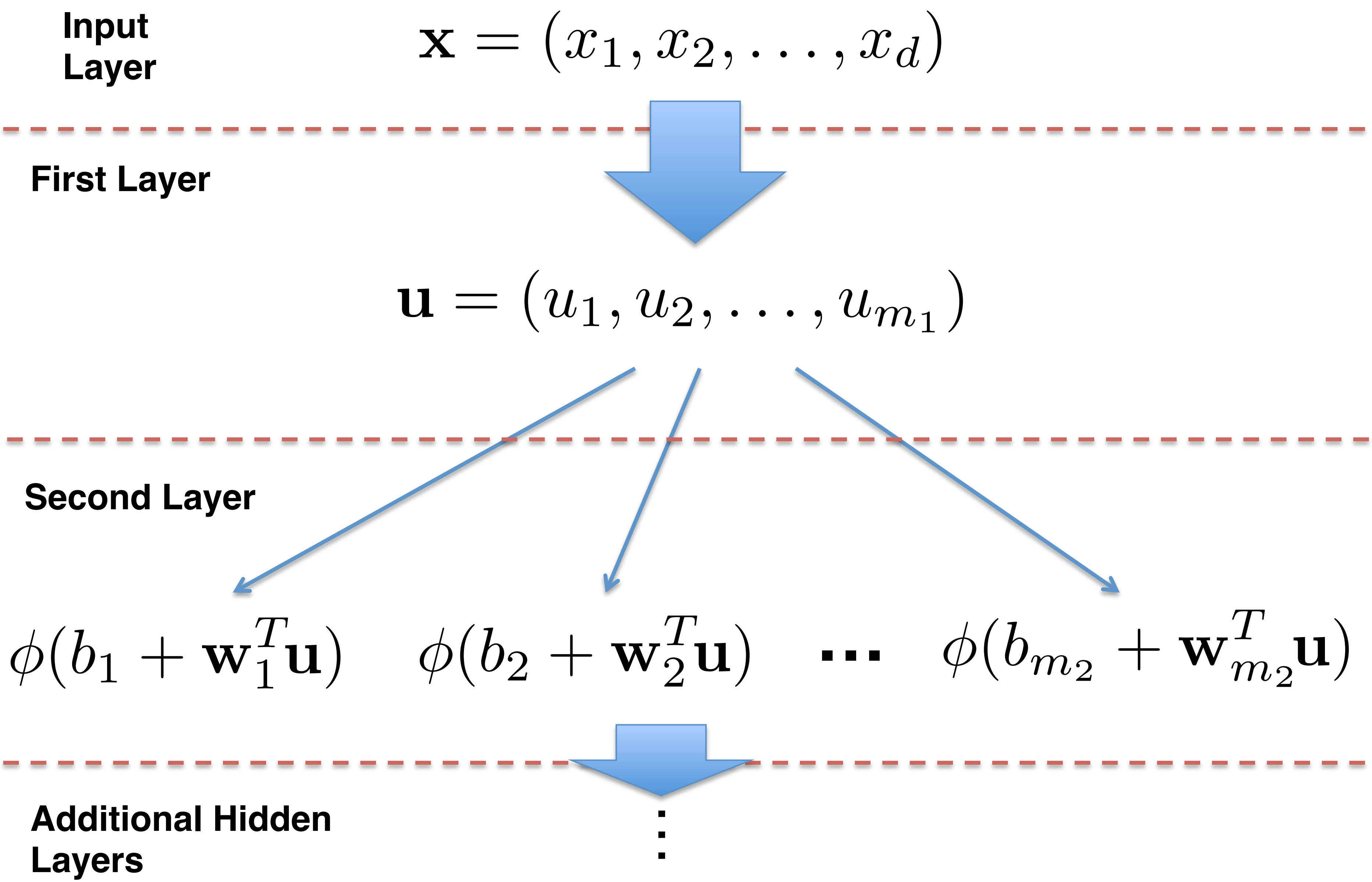
$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

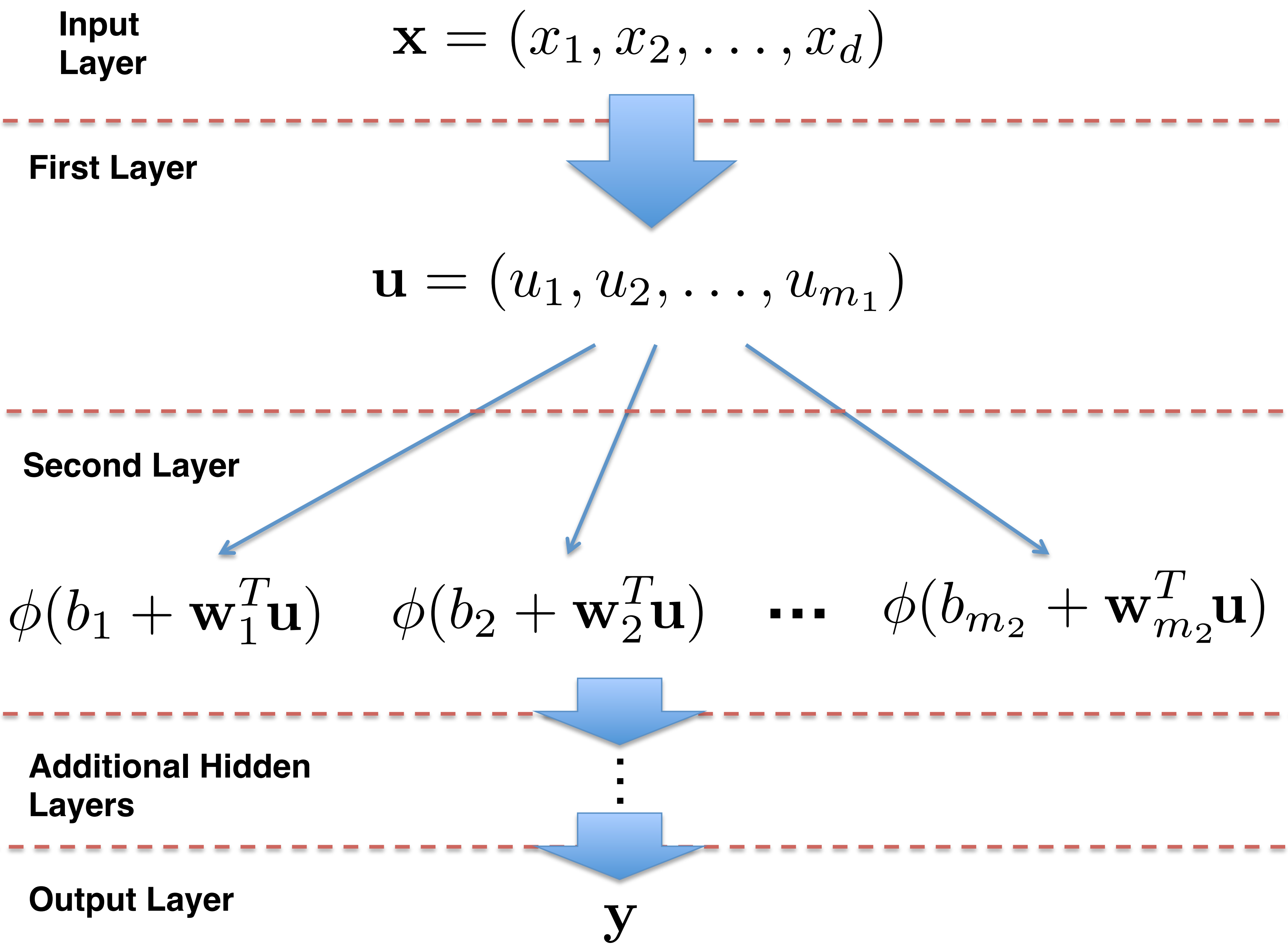
First Layer

$$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$$

Second Layer

$$\phi(b_1 + \mathbf{w}_1^T \mathbf{u}) \quad \phi(b_2 + \mathbf{w}_2^T \mathbf{u}) \quad \cdots \quad \phi(b_{m_2} + \mathbf{w}_{m_2}^T \mathbf{u})$$


Input Layer
$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
First Layer
$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$
Second Layer
$\phi(b_1 + \mathbf{w}_1^T \mathbf{u})$
$\phi(b_2 + \mathbf{w}_2^T \mathbf{u})$
$\cdots$
$\phi(b_{m_2} + \mathbf{w}_{m_2}^T \mathbf{u})$
Additional Hidden Layers
$\vdots$



Input Layer
$\mathbf{x} = (x_1, x_2, \ldots, x_d)$
First Layer
$\mathbf{u} = (u_1, u_2, \ldots, u_{m_1})$
Second Layer
$\phi(b_1 + \mathbf{w}_1^T \mathbf{u})$
$\phi(b_2 + \mathbf{w}_2^T \mathbf{u})$
$\cdots$
$\phi(b_{m_2} + \mathbf{w}_{m_2}^T \mathbf{u})$
Additional Hidden Layers
Output Layer
$\mathbf{y}$

Output Layer

There are standard choices for generating the output from the final hidden layer

Output Layer

There are standard choices for generating the output from the final hidden layer

If the output is continuous, then simply taking a linear combination is typical:

$$\mathbf{y} = b + \mathbf{w}^T \mathbf{u}$$

Output Layer

There are standard choices for generating the output from the final hidden layer

If the output is continuous, then simply taking a linear combination is typical:

$$\mathbf{y} = b + \mathbf{w}^T \mathbf{u}$$

Result of final hidden layer

Output Layer

If the output is binary, then transformation to a probability is done via the logistic sigmoid function:

$$\mathbf{y} = \frac{1}{1 + \exp(-(b + \mathbf{w}^T \mathbf{u}))}$$

Output Layer

If the output is multinomial, then transformation to a probability is done via the softmax function:

$$\mathrm{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

where

$$\mathbf{z} = \mathbf{W}^T\mathbf{u} + \mathbf{b}$$

Some Code

R using package mxnet:

```
fc1 = mx.symbol.FullyConnected(data, name="fc1", num_hidden=128)

act1 = mx.symbol.Activation(fc1, name="relu1", act_type="relu")

fc2 = mx.symbol.FullyConnected(act1, name="fc2", num_hidden=128)

act2 = mx.symbol.Activation(fc2, name="relu2", act_type="relu")

fc3 = mx.symbol.FullyConnected(act2, name="fc3", num_hidden=2)

fullnetwork = mx.symbol.SoftmaxOutput(fc3, name="sm")
```

Flexibility

A primary appeal of the approach is the flexibility in constructing the layers

- How many units are there in each layer?
- What is the mapping from one layer to the next?
- How is the output constructed from the final hidden layer?

Flexibility

A primary appeal of the approach is the flexibility in constructing the layers

- How many units are there in each layer?
- What is the mapping from one layer to the next?
- How is the output constructed from the final hidden layer?

There are alternatives to fully connected layers, e.g. convolutional networks and recurrent networks

How Does it Work?

Instead of carefully constructing a model to relate the input to the output, deep learning exploits a large collection of simple components to make a prediction

What is the role of expert knowledge?

How Does it Work?

Universal Approximation Theorem (Hornik, et al.): With enough units, a single hidden layer can approximate to arbitrary precision any “nice” function.

But: Deeper networks use units more efficiently, are easier to fit, and generalize better

How Does it Work?

But: Deeper networks use units more efficiently, are easier to fit, and generalize better

Montufar, et al.: “[f]or deep models, the maximal number of linear regions grows exponentially fast with the number of parameters, whereas, for shallow models, it grows polynomially fast with the number of parameters.”

Fitting the Model

A cost function is optimized to estimate the parameters (weights)

Choose cost function to maximize appropriate likelihood

Stochastic gradient descent with back propagation to estimate gradient

Regularization

Overfitting is a huge concern

Approaches to regularization (smoothing) manage the bias/variance tradeoff

The model is parametric, so L^2 (ridge) or L^1 (lasso) penalties on the cost function are commonly used

Regularization

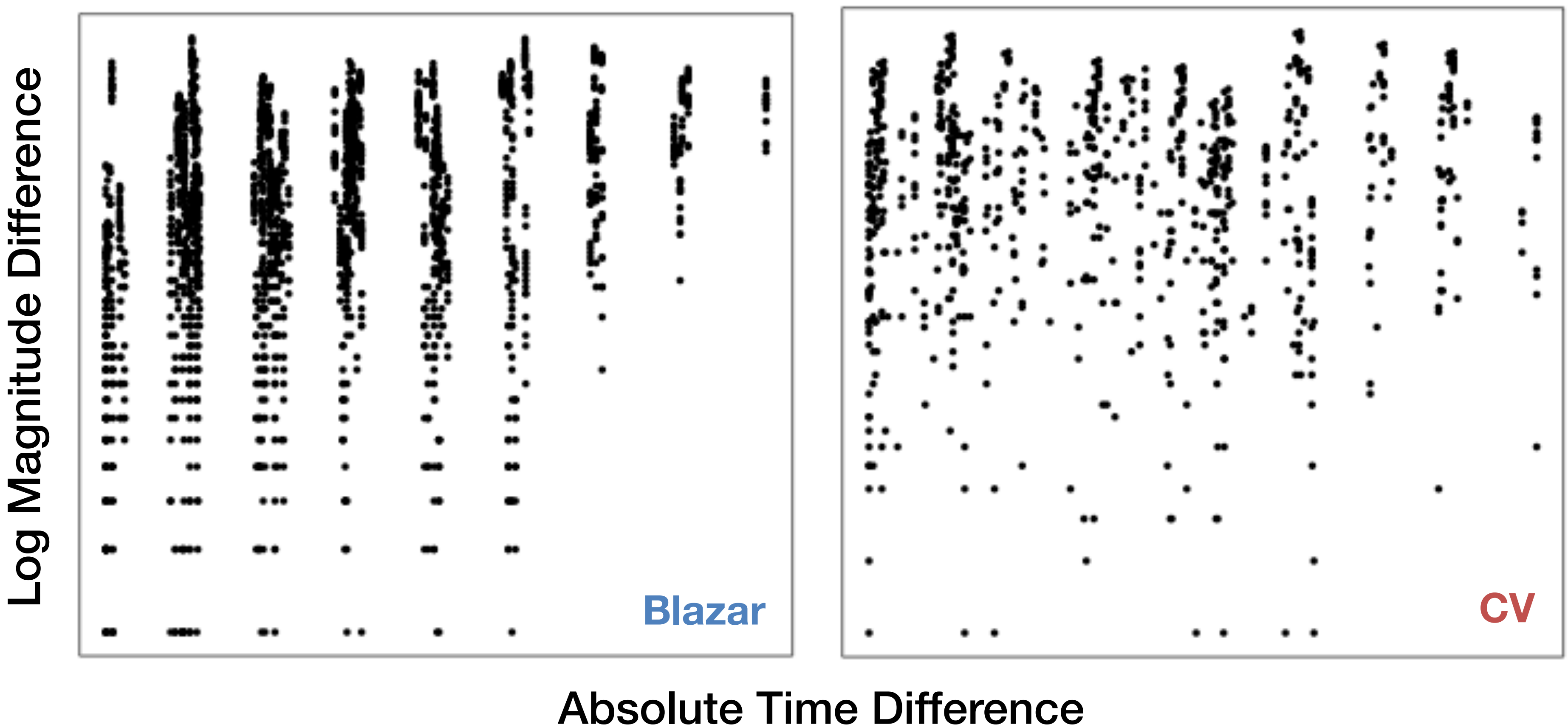
Dropout is a novel approach to regularization

Units are randomly included/excluded during training, approximating averaging over all possible submodels

Variant of bagging

Reduces potential influence of any individual unit

Blazars versus CVs



Comparison of Structure Functions

Summarizing the SF

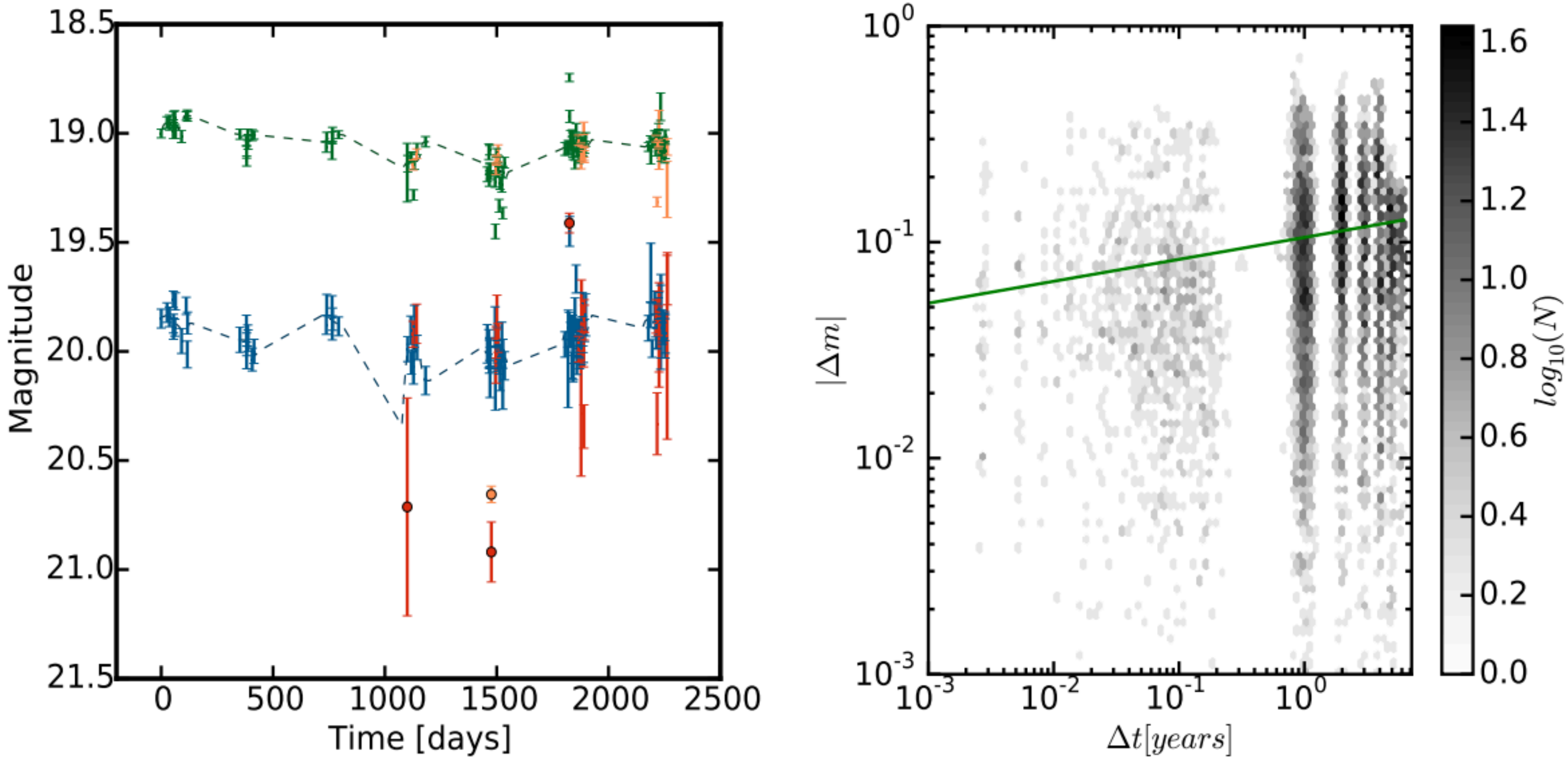


Figure 2 in Peters et al. Quasar light curve and SF

Summarizing the SF

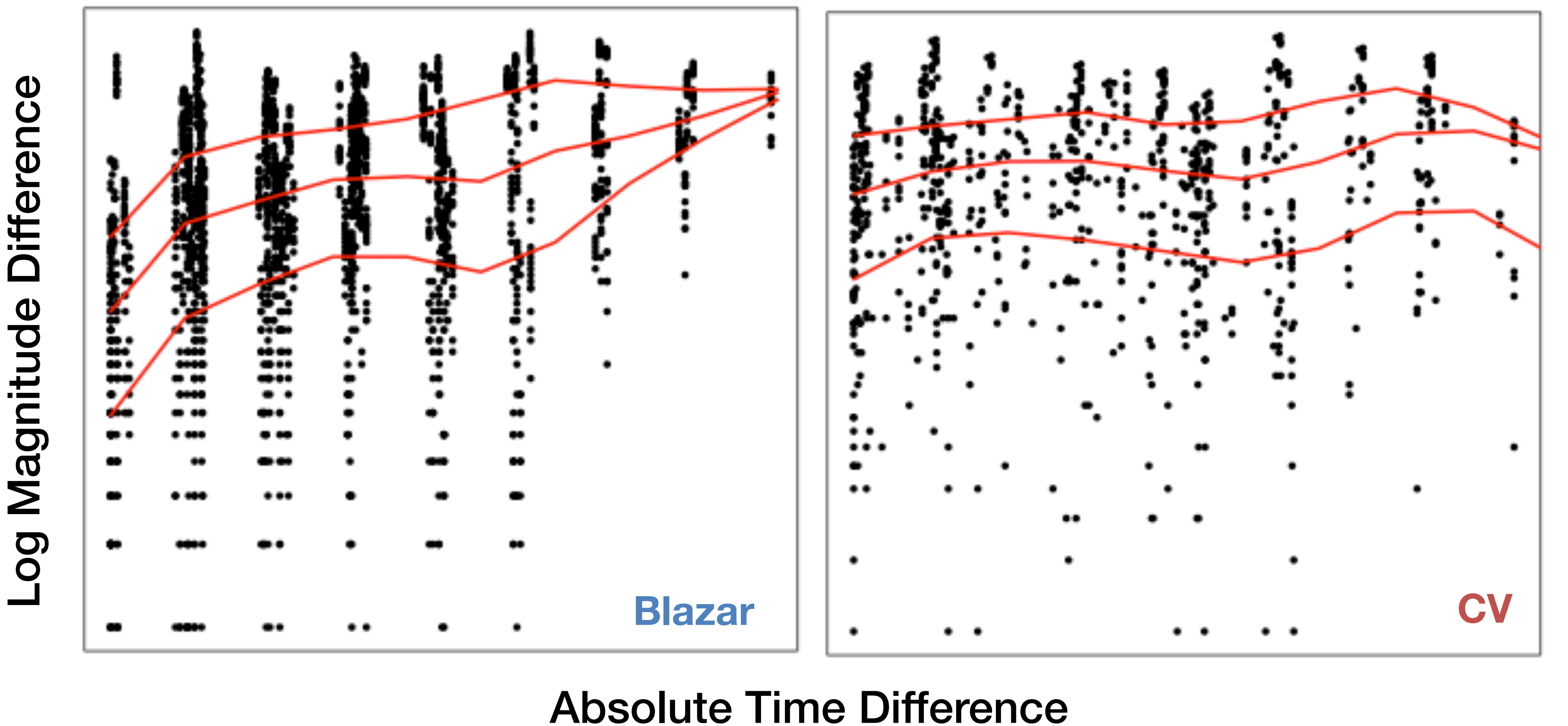
Typical to fit model to structure function

- Power Law Form (Schmidt et al.)
- Damped Random Walk (Kelly et al.)

Effort to find a low-dimensional representation, avoiding the curse of dimensionality

Ideally, could utilize higher-dimensional representation

Blazars versus CVs



Quantile regression fits

Blazar versus CV

Fit model with three hidden layers, using Dropout

128 nodes per layer

Rectified linear units as the activation functions

958 CVs, 318 Blazars from Catalina Real-Time Transient Survey

Blazar versus CV

Performance on test set:

Prediction \ Truth	Blazar	CV
Blazar	18	10
CV	8	91

Blazar versus CV

Performance on test set:

Deep Learning

Prediction \ Truth	Blazar	CV
Blazar	18	10
CV	8	91

Random Forest

Prediction \ Truth	Blazar	CV
Blazar	12	8
CV	14	93

Potential of Deep Learning

Best suited to situations where high-dimensional input is required

Avoid the curse of dimensionality

Seems particularly relevant for classification challenges

Can be extended to unsupervised learning - autoencoders

Summary

- Motivation: Representing Data
- Nonparametric Regression
- Curse of Dimensionality
- Additive Models
- Neural Nets
- Deep Learning

References

Drake, AJ, et al. ApJ (696): 870

Eyer L, and Mowlavi, N. 2008. Variable Stars across the Observational HR Diagram. Journal of Physics Conference Series

Goodfellow, Bengio, and Courville. *Deep Learning*

Hastie, Tibshirani, and Friedman, *Elements of Statistical Learning*

Hornik, et al. Neural Networks (3): 551

Kelly, et al. ApJ (698): 895

Montufar, et al. *NIPS 2014*

Peters, et al. ApJ (811): 95

Schmidt et al. ApJ (714): 1194