

Predicting Arm Motion from Motorcortical Activity

Andrew H. Fagg
Symbiotic Computing Laboratory
School of Computer Science
University of Oklahoma

In collaboration with:
Nicholas Hatsopoulos (University of Chicago)
Lee Miller (Northwestern University)
Gregory W. Ojakangas (Drury University)
David Goldberg (OU)

Funded by:
National Institutes of Health
University of Oklahoma

Current Prosthetic Technology

Actuation unnatural:

- Activation of other muscles
- Use of other hand

Limited sensory feedback:

- Vision
- Tactile stimulation



From Shawnee News-Star

Future Prosthetic Technology



Future Prosthetic Technology

Direct connection
from brain to limb
carries motor
commands



Future Prosthetic Technology

Direct connection
from limb to brain
carries sensory
information



Our Focus Today

Controlling a prosthetic (robot) arm using the activation level of a set of neurons:

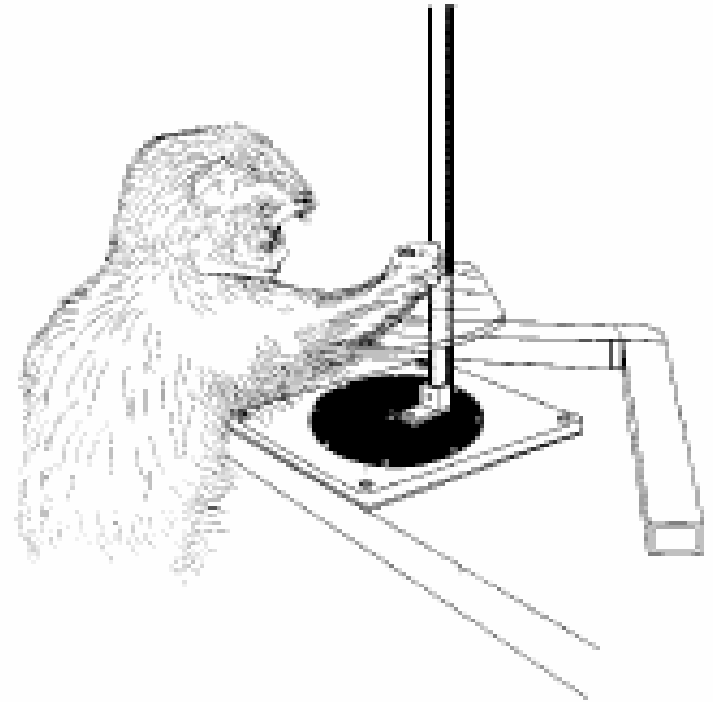
- Start by observing natural movements of the arm + the corresponding neural activity
- Construct a model that allows us to predict arm motion from neural activity
- (future) use these predictions to drive the robot arm

Fundamental Questions

- What is the best way to construct and evaluate our models?
- How do we describe the motion of the arm (or what is motor cortex encoding)?
 - Cartesian position of the wrist
 - Joint torques
 - (many others possible)

A Typical Experimental Setup

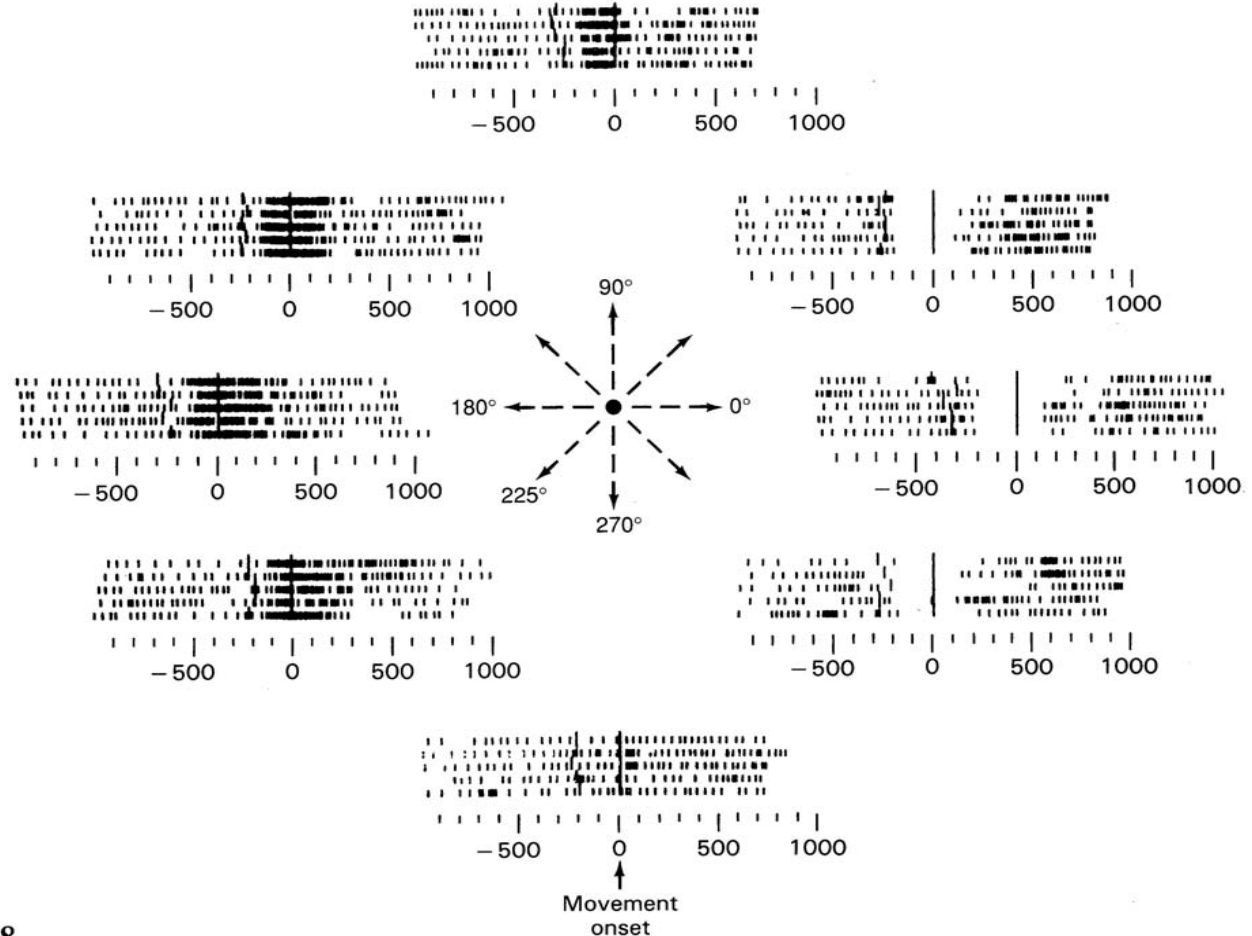
- Monkey makes planar arm movements
 - Hand, elbow, and shoulder are constrained to operate in a horizontal plane
- Arm movements are to lighted targets



Scott & Kalaska (1997)

Motor Cortex Activity During A Reach

- Data from a single neuron
- Neuron is most active for particular directions of movement!

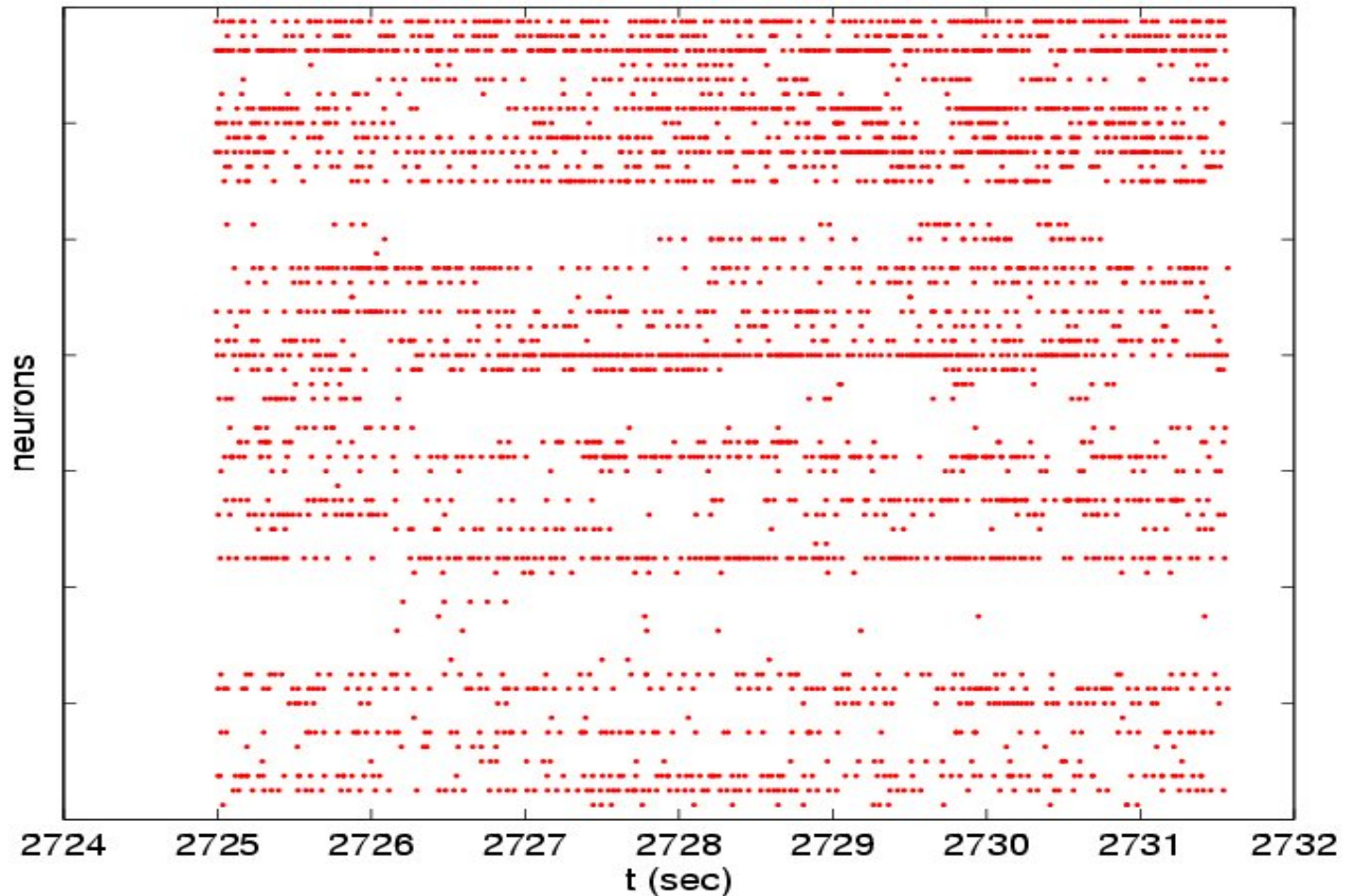


Our Experimental Setup

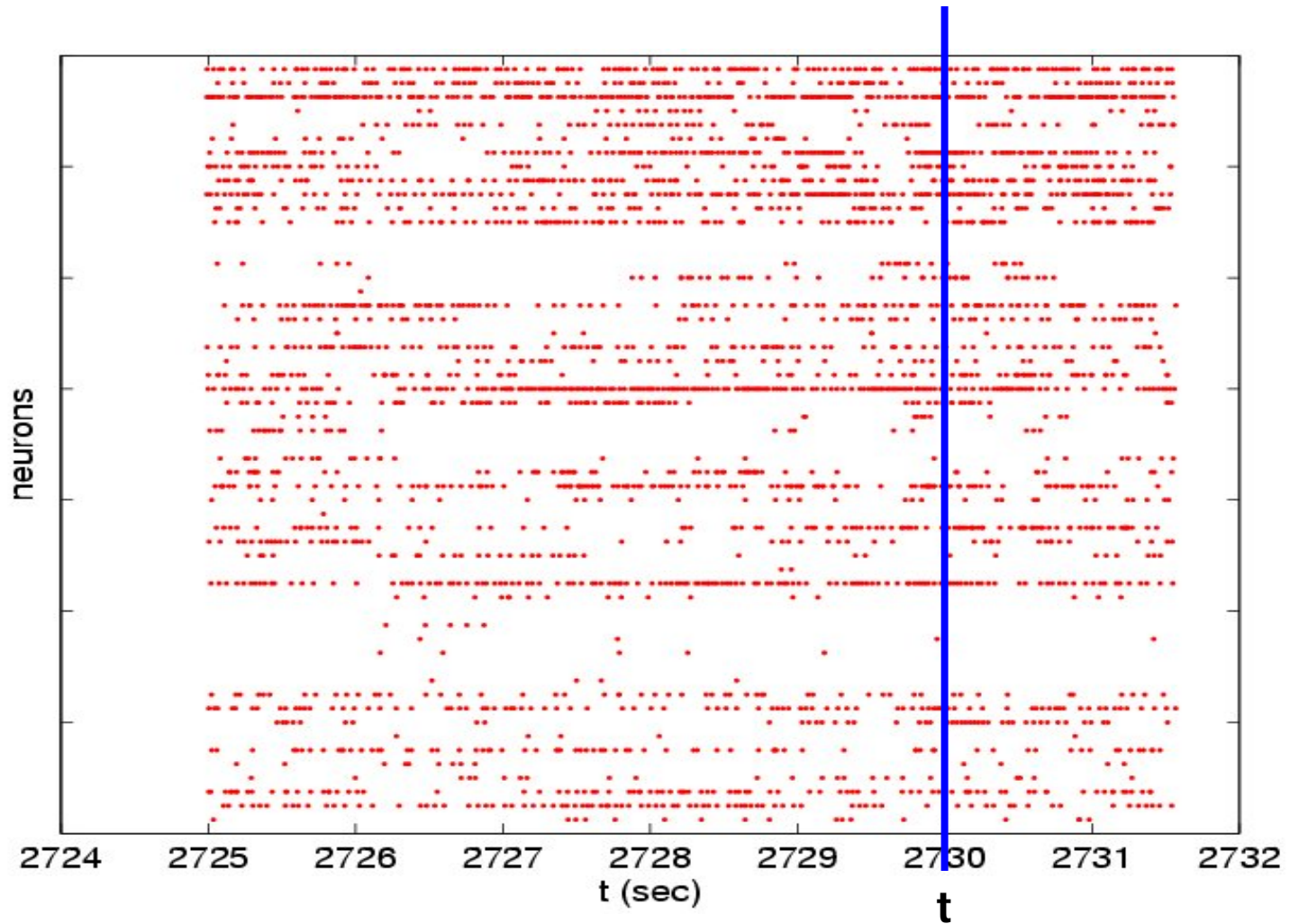
Hatsopoulos lab

- Monkey wears an “exoskeleton” robot
- Movements constrained to the horizontal plane
- Exoskeleton can drive the elbow and shoulder OR can operate in a passive mode
- Simultaneously record from ~50-100 neurons

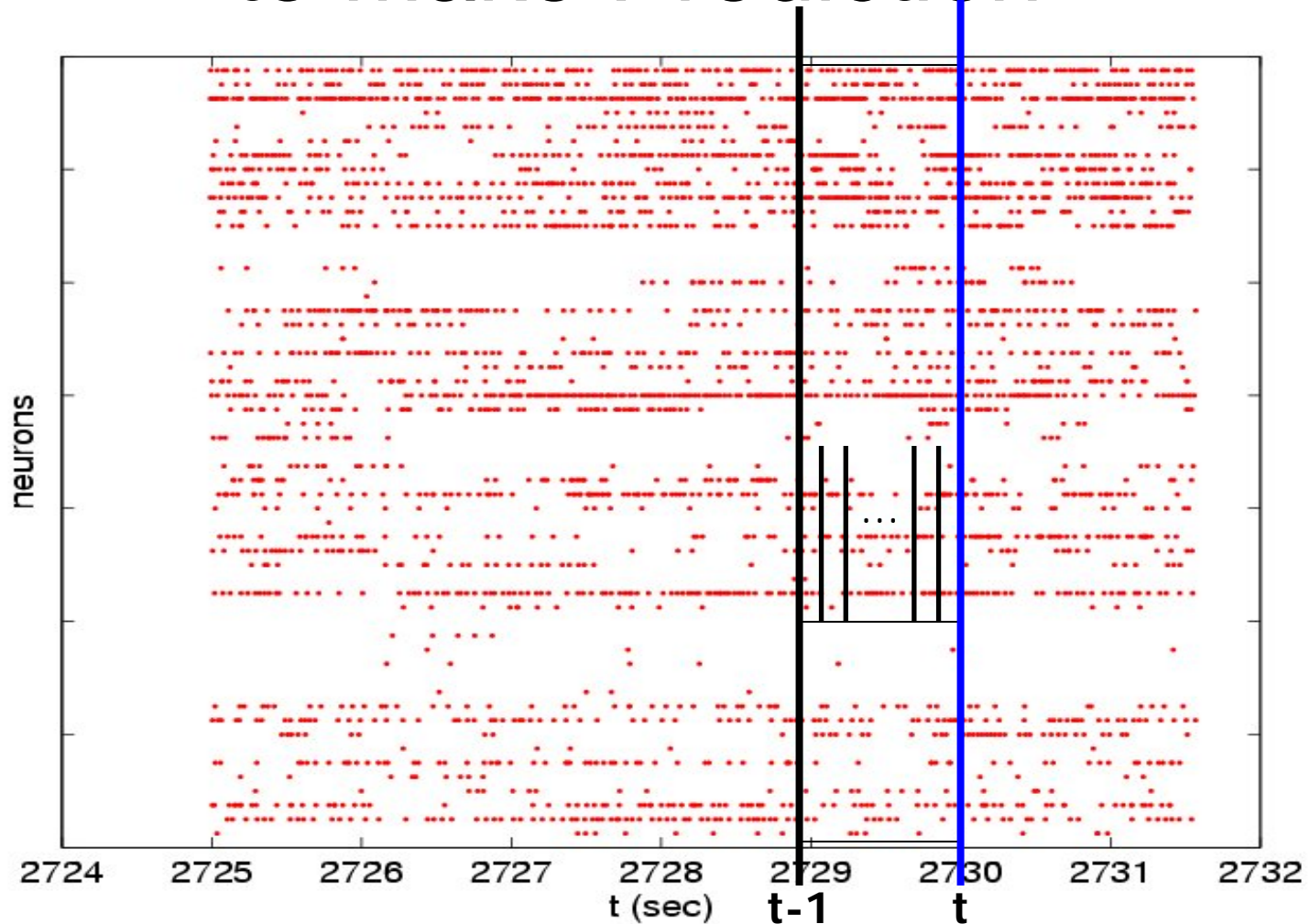
Spike Data From Single Trial



Predicting Arm Motion at Time t



Use One Second of Data to Make Prediction



One second of data is partitioned into 20 bins

Predictor Implementation

- One data set:
 - 55 neurons * 20 time bins = 1100 features
- Each feature encodes the firing rate of the neuron during the 50ms bin
- Construct a model that uses this feature vector to predict:
 - Cartesian location of the wrist, or
 - Torque applied to the joints

Models

What do we really mean by 'model'?

What Do We Really Mean by 'Model'?

One definition:

- A description of the relationship between observable variables
- In our case, we will use a **function** as an explicit description of this relationship

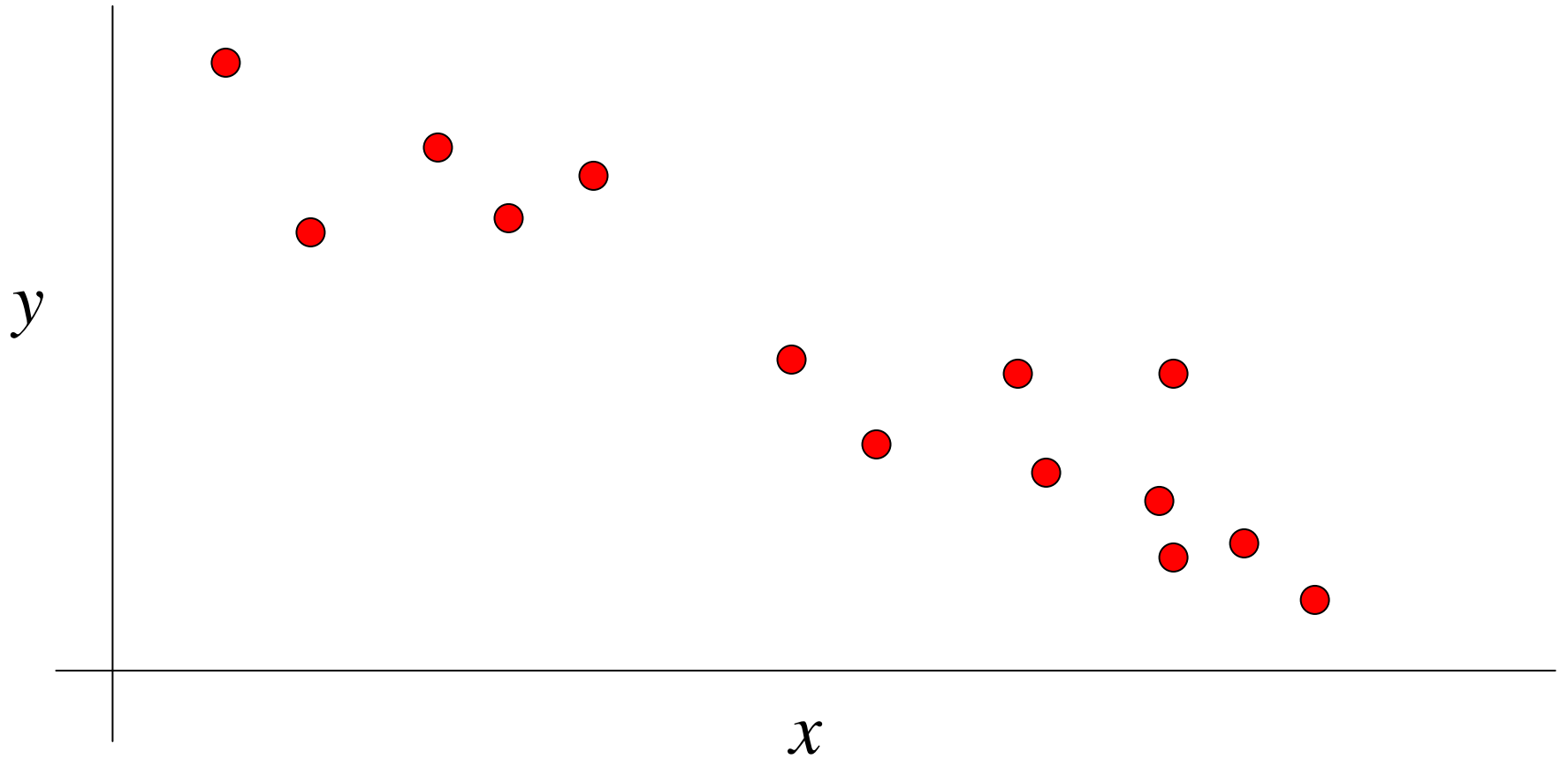
Where Do Models Come From?

Where Do Models Come From?

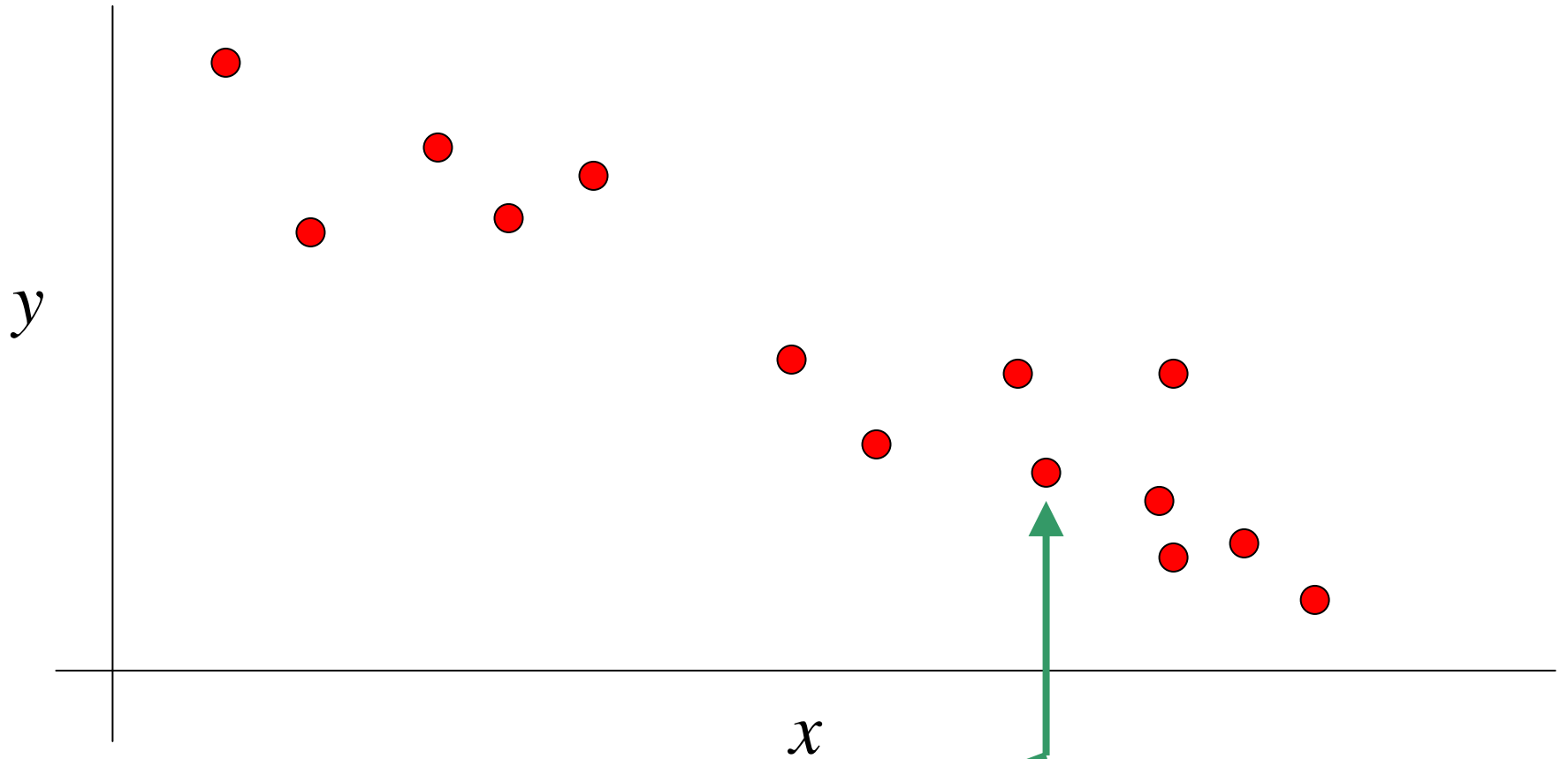
An empirical approach:

- Construct a model from a set of observations
- Each observation is an input/output (X/Y) pair
- Refer to this as the **training set**

An Example

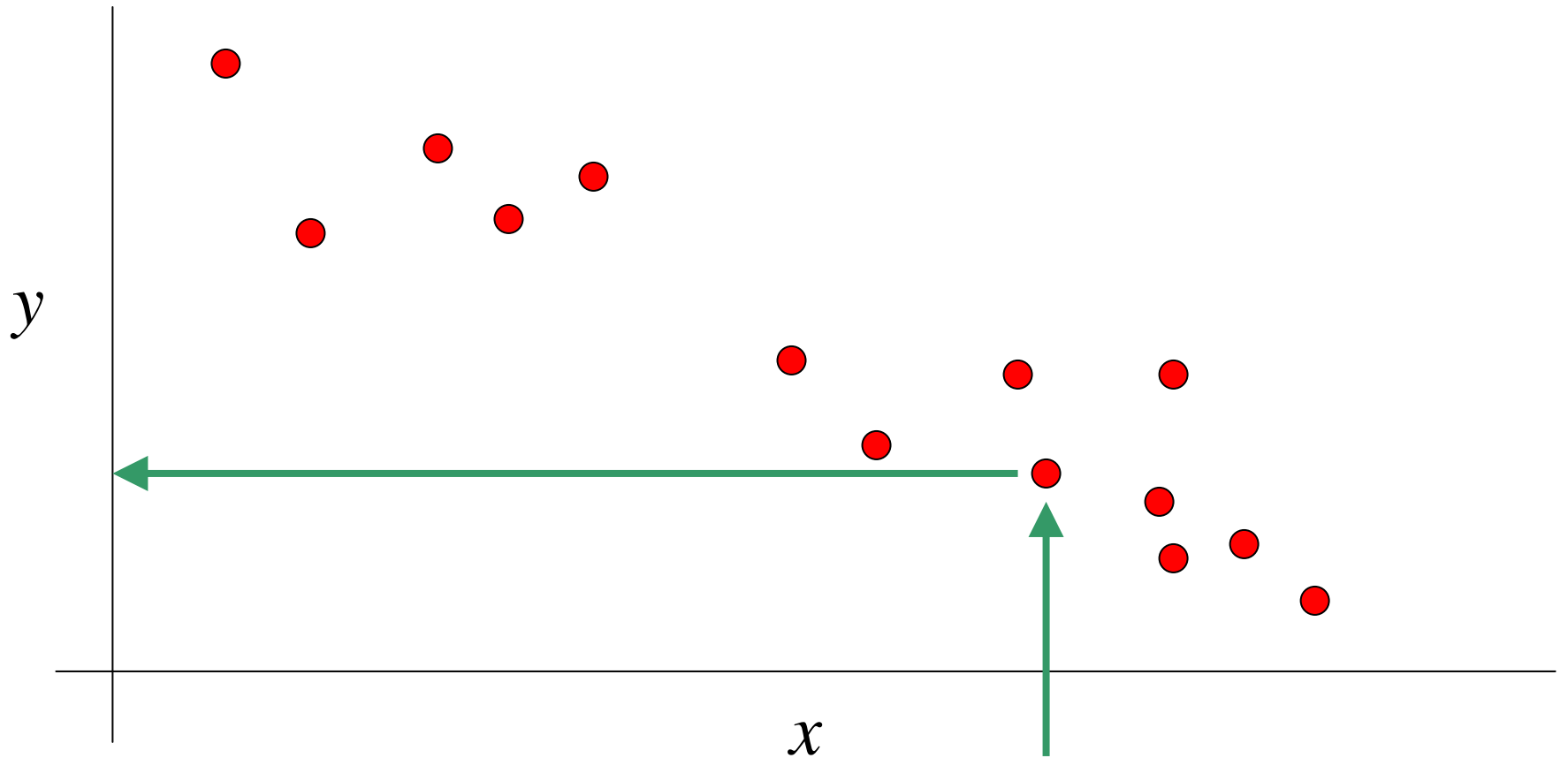


Querying a Model



A query: given some x_q ...

Querying a Model



A query: given some x_q produce a corresponding y_q

Generalization

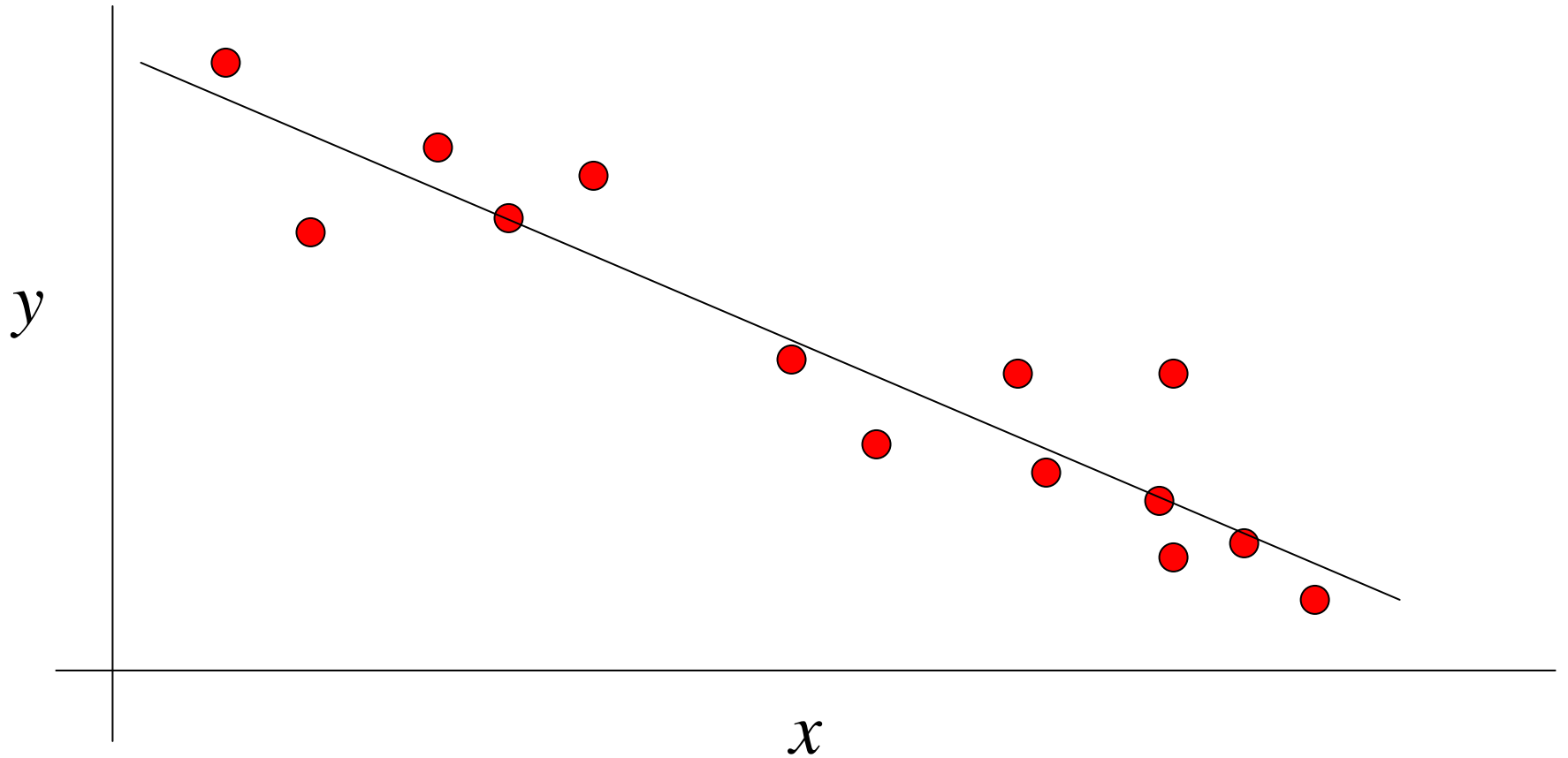
What happens when the query does not match a training input?

Generalization

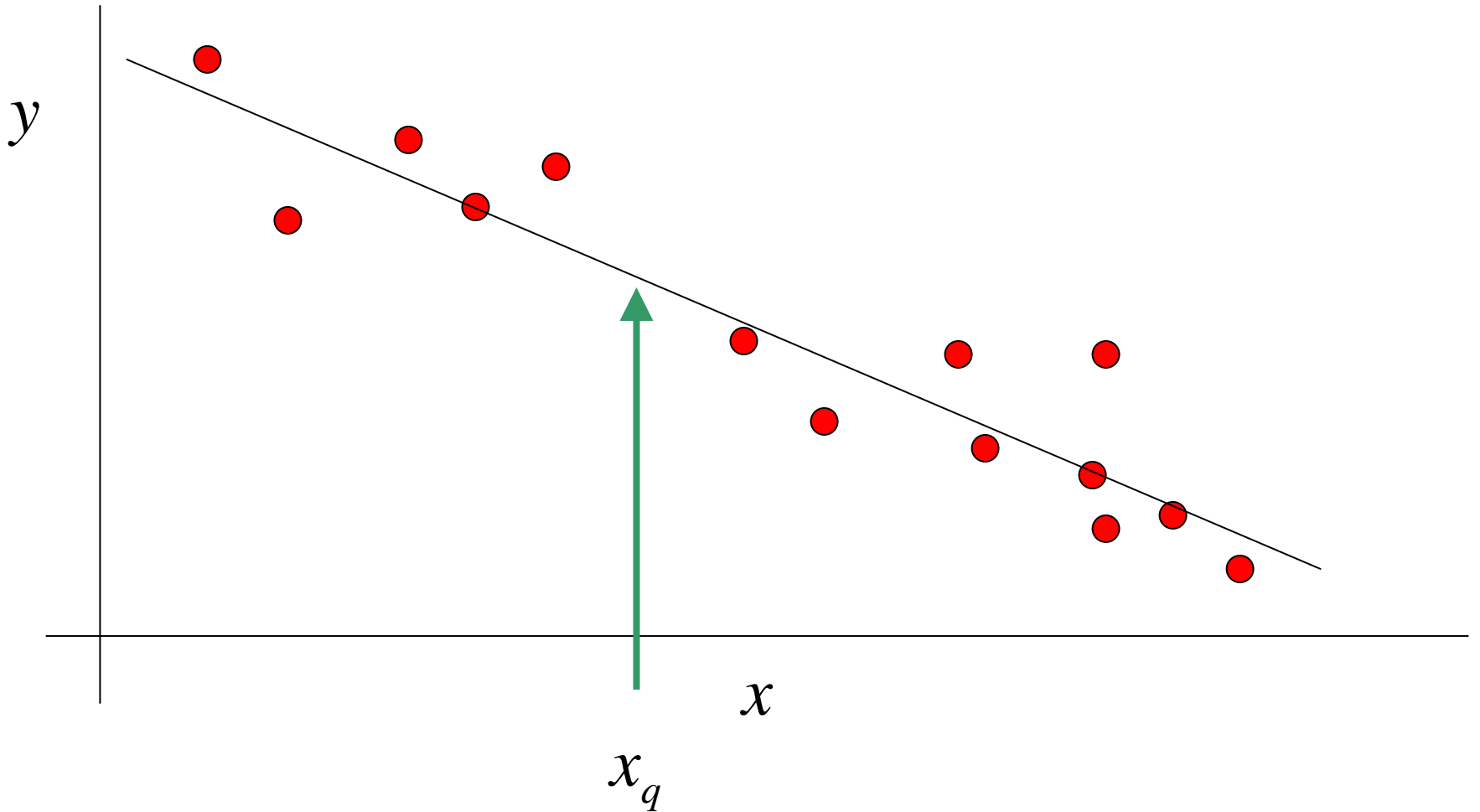
What happens when the query does not match a training input?

- We would like for our model to generalize (reasonably) to this novel situation
- We can do this by fitting a function to the data set...

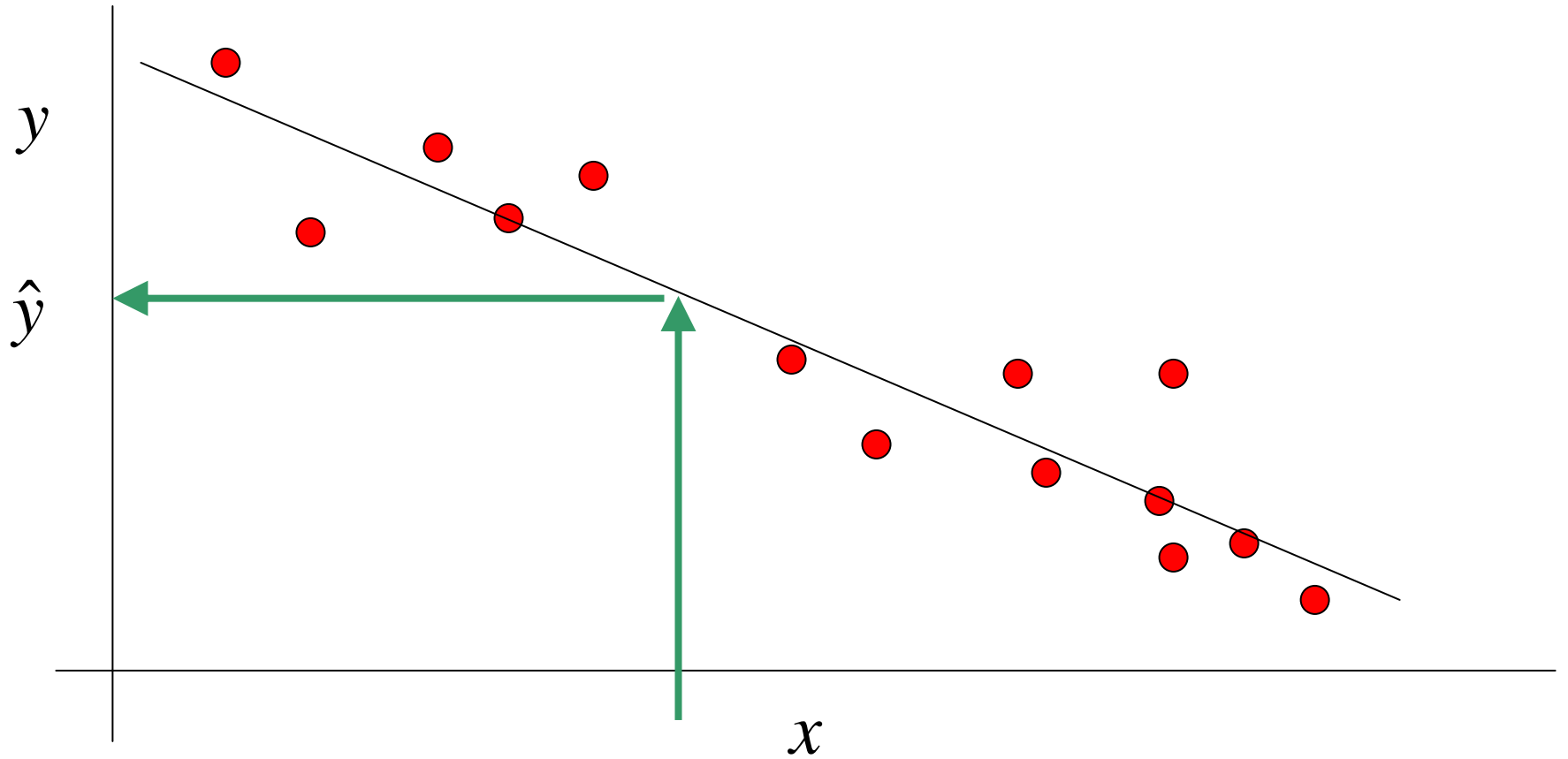
Fitting a Model to a Training Set



A Novel Query



A Novel Query



A Linear Model

For our example, the model is:

$$\hat{y} = f_W(x) = w_1 * x + w_0$$

A Linear Model

For our example, the model is:

$$\hat{y} = f_W(x) = w_1 * x + w_0$$

In the more general case:

$$\hat{y}_j = f_W^j(X) = w_{0j} + \sum_{i=1}^N w_{ij} x_i$$

A Linear Model

In the more general case:

$$\hat{y}_j = f_W^j(X) = w_{0j} + \sum_{i=1}^N w_{ij} x_i$$

And in vector-matrix notation:

$$\hat{Y} = f_W(X) = W X$$

A Linear Model

In the more general case:

$$\hat{y}_j = f_W^j(X) = w_{0j} + \sum_{i=1}^N w_{ij} x_i$$

And in vector-matrix notation:

$$\hat{Y} = f_W(X) = W X$$



Column vector of dimension m

A Linear Model

In the more general case:

$$\hat{y}_j = f_W^j(X) = w_{0j} + \sum_{i=1}^N w_{ij} x_i$$

And in vector-matrix notation:

$$\hat{Y} = f_W(X) = W X$$



Matrix of dimension $n \times m$

A Linear Model

In the more general case:

$$\hat{y}_j = f_W^j(X) = w_{0j} + \sum_{i=1}^N w_{ij} x_i$$

And in vector-matrix notation:

$$\hat{Y} = f_W(X) = W X$$



Column vector of dimension n

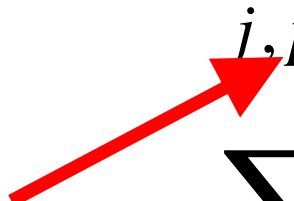
Measuring Degree of Model Fit

- Model performance is measured with an error function
- In this context, a common error function is:

$$\begin{aligned} E &= \sum_{j,p} \left(y_{jp} - \hat{y}_{jp} \right)^2 \\ &= \sum_{j,p} \left(y_{jp} - f_W^j \left(X_p \right) \right)^2 \end{aligned}$$

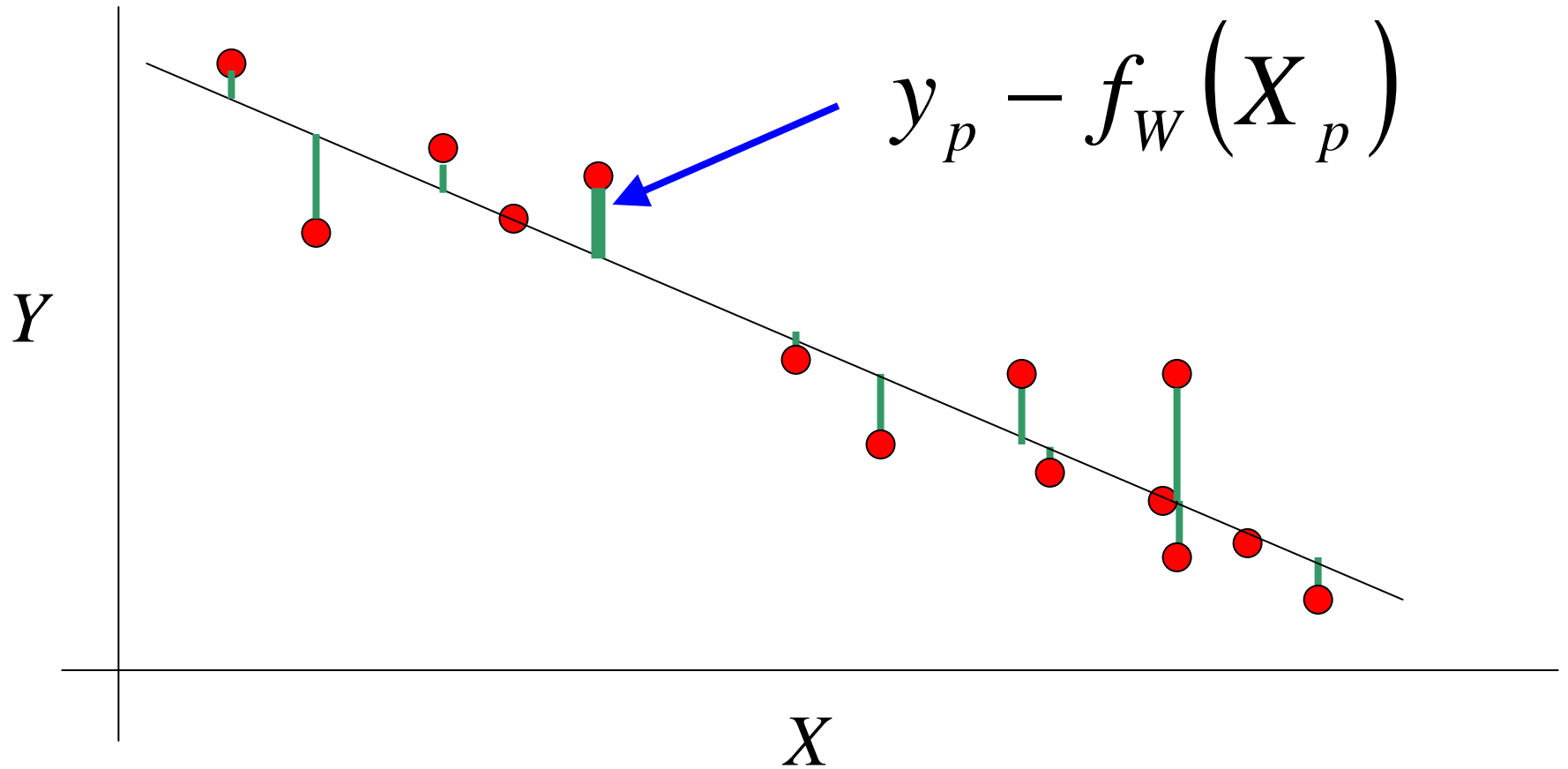
Measuring Degree of Model Fit

- Model performance is measured with an error function
- In this context, a common error function is:

$$E = \sum_{j,p} (y_{jp} - \hat{y}_{jp})^2$$
$$= \sum_{j,p} (y_{jp} - f_W^j(X_p))^2$$


One “p” for each
data point

Measuring Degree of Model Fit



Selecting Model Parameters

We often will select a set of model parameters (w) so as to minimize E

There are a variety of techniques available to us, including:

- Gradient methods
- Inverse methods (e.g., pseudo-inverse)

In statistics we call this **regression**

Pseudo-Inverse Method of Regression

Prediction for a single input (our vector-matrix equation again):

$$\hat{Y}_p = W X_p$$

Pseudo-Inverse Method of Regression

Prediction for a single input (our vector-matrix equation again):

$$\hat{Y}_p = W X_p$$

We can also perform all predictions simultaneously:

$$\left[\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_P \right] = W \left[X_1, X_2, \dots, X_P \right]$$

Pseudo-Inverse Method of Regression

$$\left[\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_P \right] = W \left[X_1, X_2, \dots, X_P \right]$$

Which we can simplify to:

$$\hat{Y} = W X$$

(where \hat{Y} and X are now matrices)

With the real Y would like to solve for W , but X is not square...

Pseudo-Inverse Method of Regression

We would like to solve for W , but X is not square

So: we use the Moore-Penrose pseudo-inverse:

$$Y X^+ = W X X^+$$

Pseudo-Inverse Method of Regression

We would like to solve for W , but X is not square

So: we use the Moore-Penrose pseudo-inverse:

$$Y X^+ = W X X^+$$

Under certain conditions (including this one):

$$X X^+ = I$$

Pseudo-Inverse Method of Regression

We would like to solve for W , but X is not square

So: we use the Moore-Penrose pseudo-inverse:

$$\hat{Y} X^+ = W X X^+$$

Under certain conditions (including this one):

$$X X^+ = I$$

Therefore: $W = Y X^+$

Pseudo-Inverse Method of Regression

$$W = Y X^+$$

This generally does not provide a perfect solution, but it does minimize our original error function:

$$E = \sum_{j,p} \left(y_{jp} - \hat{y}_{jp} \right)^2$$

Back to Our Arm Motion Prediction Problem ...

- X_t : description of neuron activation for a set of cells over the last 1 second
- Y_t : a description of the state/motion of the arm (e.g., torque applied to the shoulder and elbow joints)
 - (this is a direct observation)
- \hat{Y}_t : a prediction of arm motion

Back to Our Arm Motion Prediction Problem ...

- The monkey will perform many trials through the course of an experiment (~200)
- For each trial, the monkey makes ~6-7 pointing movements
- We will take samples for X_t and Y_t every 50 msec

Constructing Our Model

- Given a set of samples (on the order of 20,000), we can use our pseudo-inverse method to estimate W
- We can then measure the resulting prediction performance on this **training set**
- What is missing?

Evaluating our Model

What is missing?

- Fundamentally, we want to be able to talk about the performance of the predictor on future data sets
- Approach: withhold some of our data from the training process & use it only for measuring performance
 - We call this a **test set**

N-Fold Crossvalidation

- Standard procedure for evaluating a model building approach
- Explicitly addresses the problem of having too little data (remember that our input vector is ~ 1100 dimensions)
- First step: split the data you have into N independent sets (we call them **folders**)

N-Fold Crossvalidation

We will build N different models

- First model:
 - Use folds 1 ... $N-1$ to train the model
 - Evaluate the resulting model using the N th fold
- Second model:
 - Use folds 1 ... $N-2, N$ to train the model
 - Evaluate the resulting model using fold $N-1$
- Etc ...

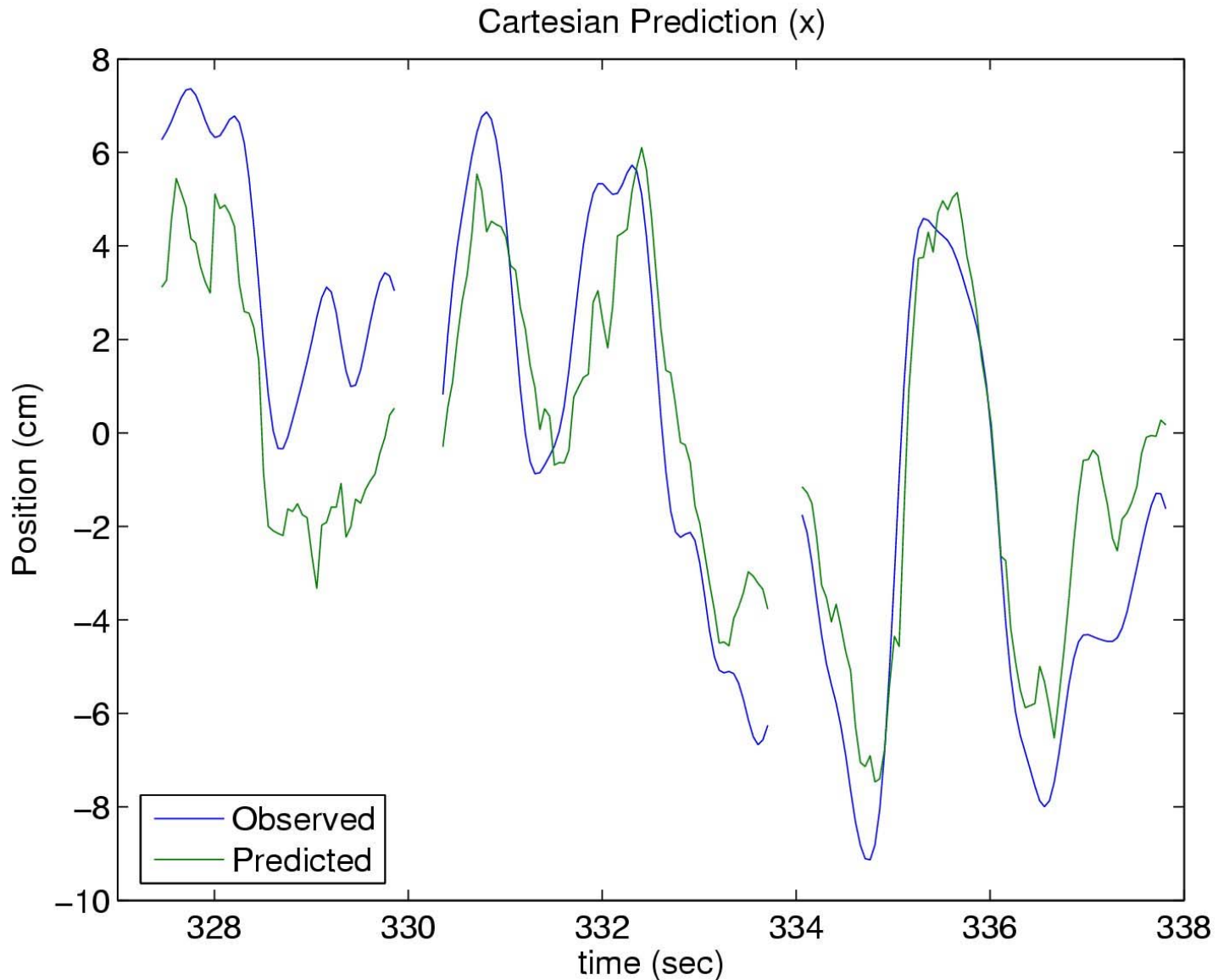
N-Fold Crossvalidation

This procedure gives us N different performance measures

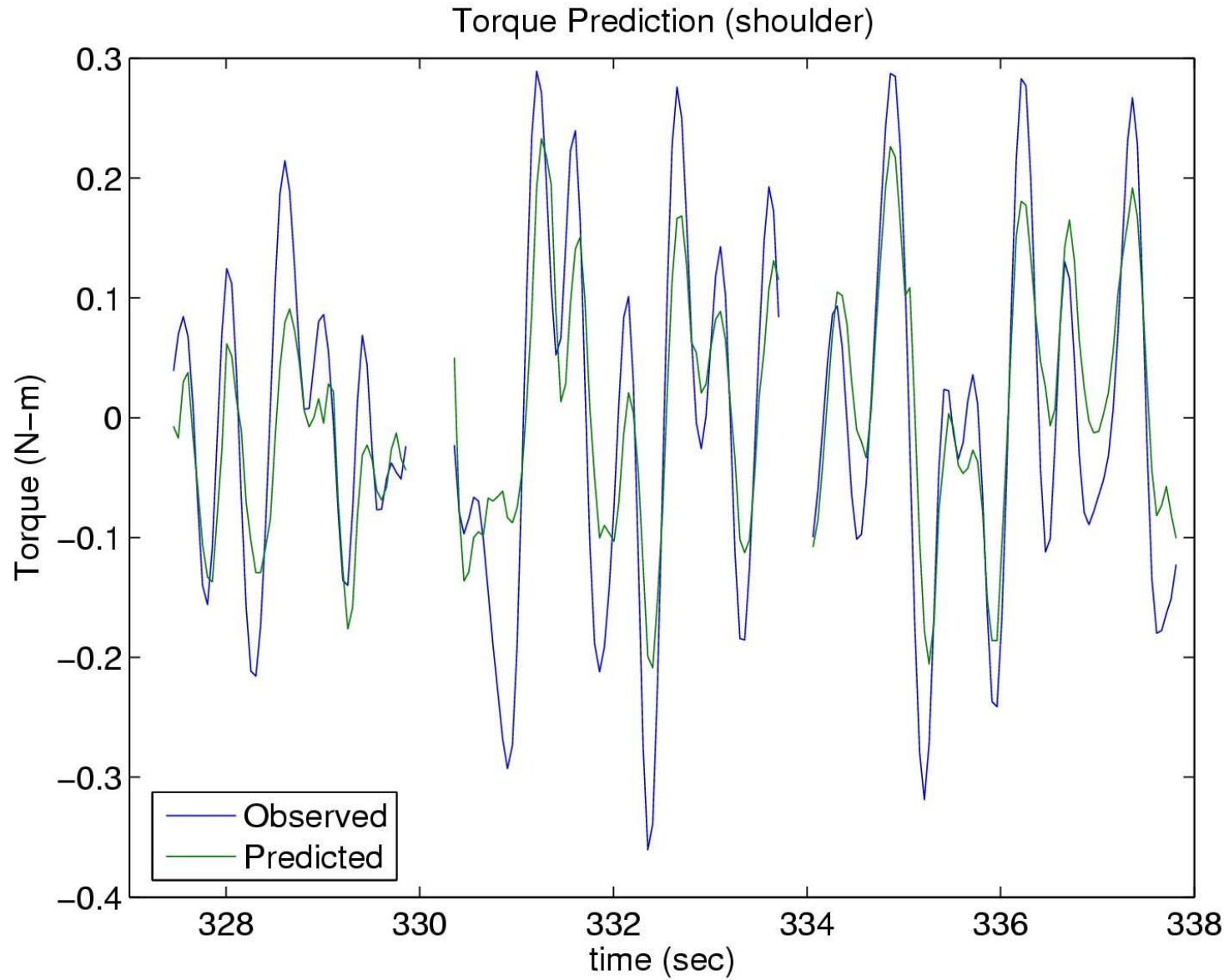
- We can then look at the mean & standard deviation of these measures

In our work, we are using 20-fold crossvalidation

Results: Predicting the Cartesian Position of the Wrist



Results: Predicting Joint Torque



Comparative Evaluation

Want to be able to compare prediction of joint torque versus Cartesian position

- Use **fraction of variance accounted for**:

$$vaf_j = 1 - \frac{\sum_{p=1}^P (y_{jp} - \hat{y}_{jp})^2}{\sum_{p=1}^P (y_{jp} - \bar{y}_j)^2}$$

\hat{y} = prediction

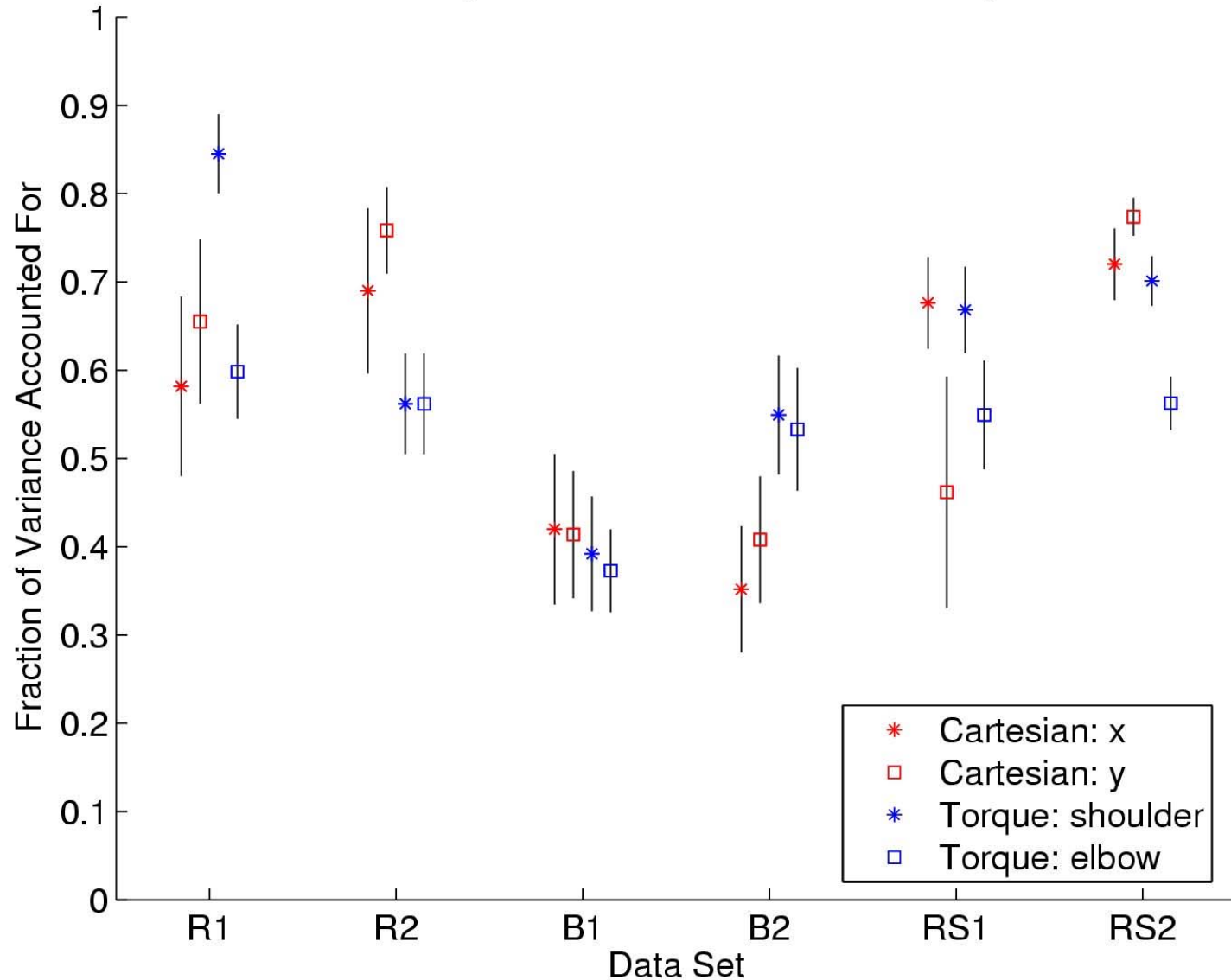
y = observation

\bar{y} = mean observation

Comparative Results

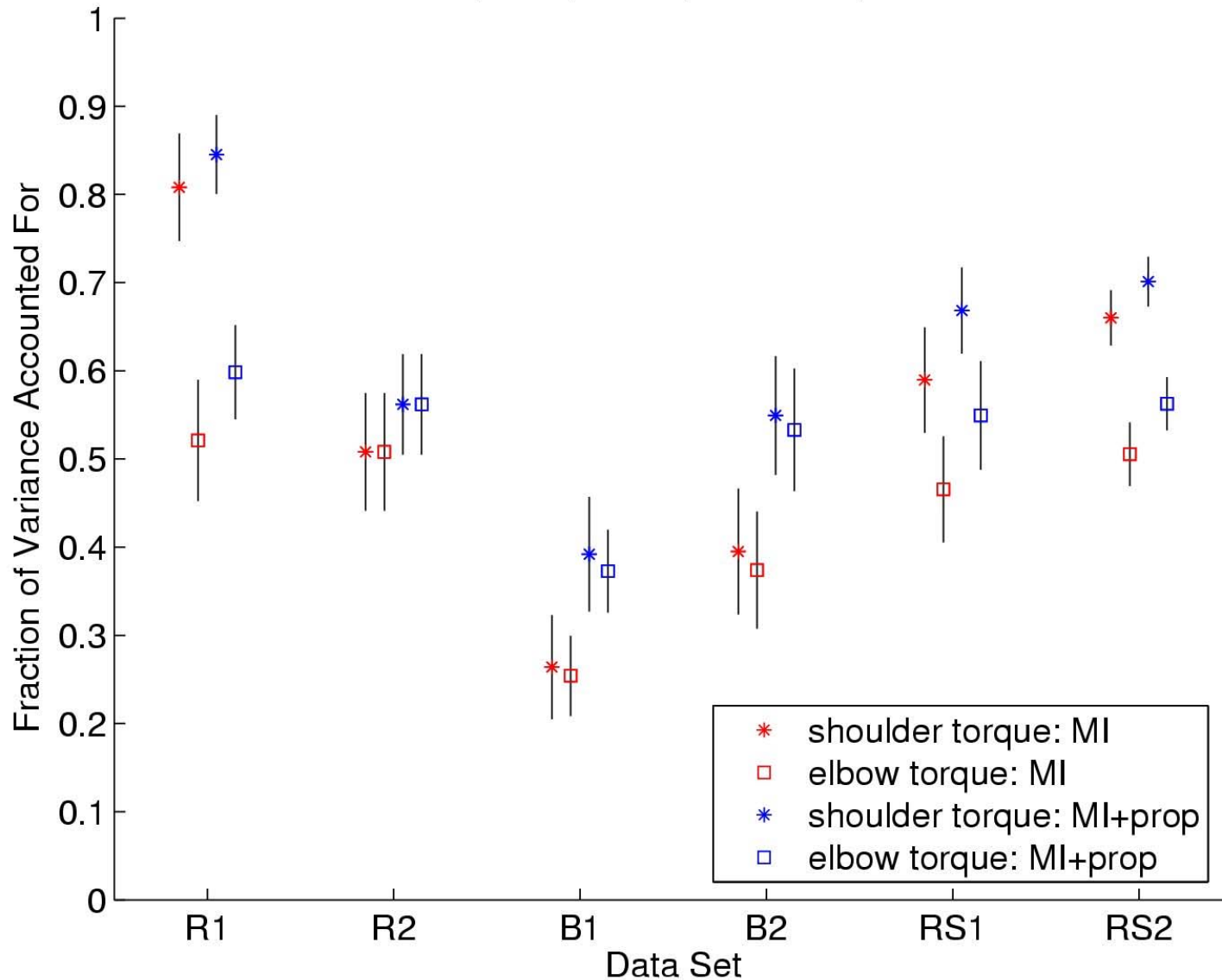
Predicting Cartesian Location vs Joint Torque

6 independent
data sets



Adding Joint State to Torque Predictor

Effect of Proprioceptive Input on Torque Prediction



Conclusions: Methods

- Linear models can often take us pretty far
 - Nice mathematical properties + we can use the pseudo-inverse to derive a solution
 - ... But nonlinear models are often necessary
- Independent training and testing
- Crossvalidation allows us to deal with small amounts of data

Conclusions:

Interpreting Neural Activity

- With 50-100 neurons we are able to predict arm motions to a reasonable degree (but still want to do better)
- What information is being encoded in the primary motor cortex?
 - Our results **suggest** that the representation is closer to one that captures intrinsic and dynamic information (ie torque)