CS 175: Project in Artificial Intelligence

Slides 3: Regression

Topic 5: Regression

Slides taken from Prof. Smyth and Prof Ihler
(with slight modifications)

Data in Matrix Form

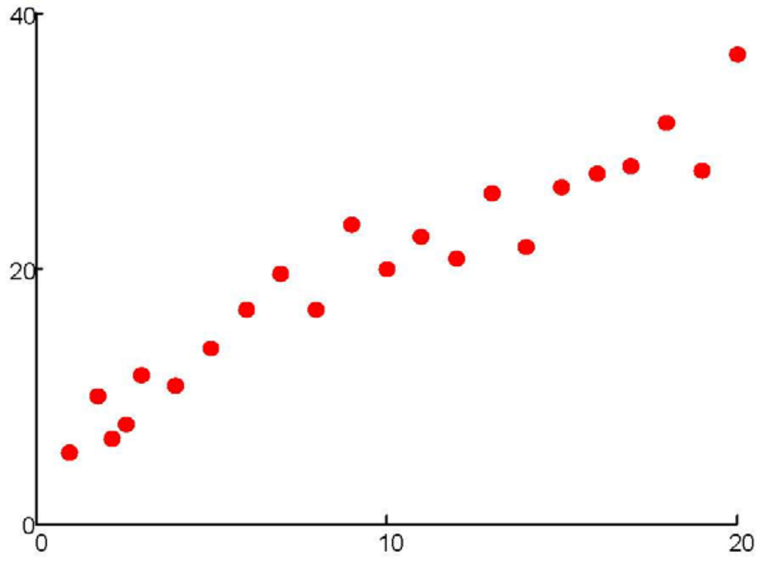
Measurements →

Entities ↓

ID	Income	Age	….	Monthly Debt	Good Risk?
18276	65,000	55	….	2200	Yes
72514	28,000	19	….	1500	No
28163	120,000	62	….	1800	Yes
17265	90,000	35	….	4500	No
…	…	…	….	…	…
…	…	…	….	…	…
61524	35,000	22	….	900	Yes

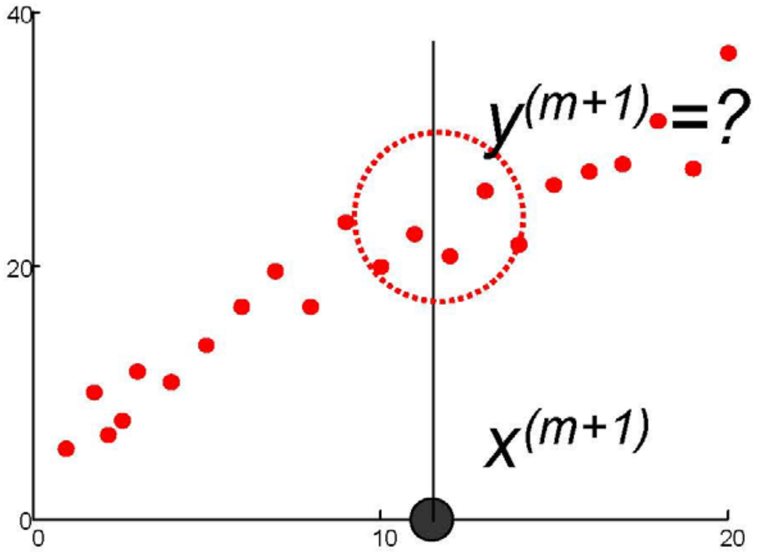
"Measurements" may be called "variables", "features", "attributes", "fields", etc

Scatter plots



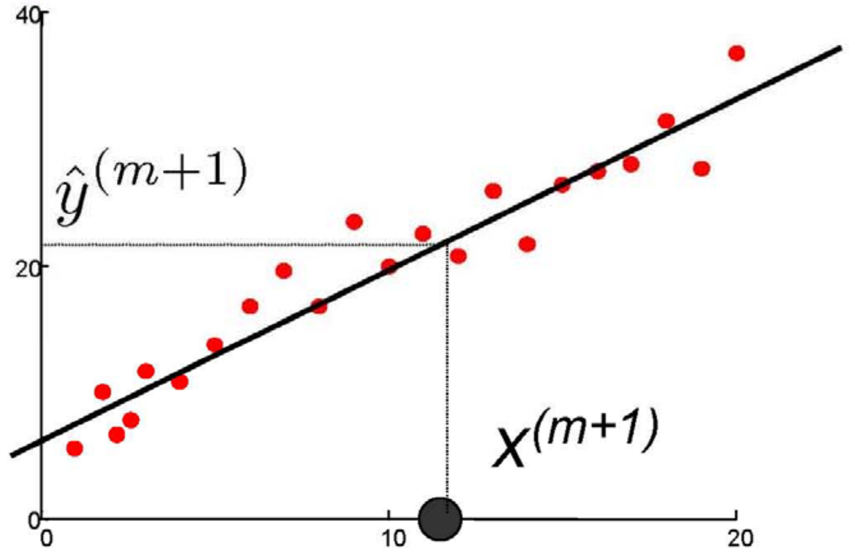
- Suggests a relationship between x and y
- *Prediction*: new x, what is y?

Predicting new examples



- Regression: given the observed data, estimate $y^{(m+1)}$ given new $x^{(m+1)}$

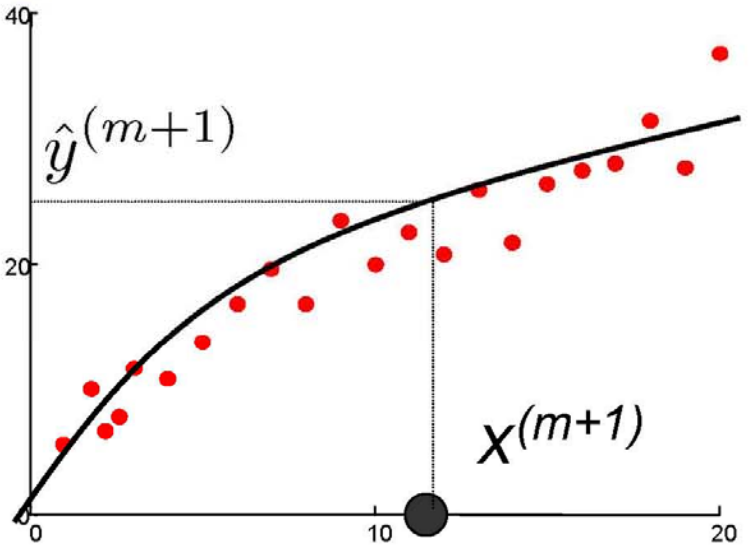
Regression functions



Linear function

- Regression: given the observed data, estimate $y^{(m+1)}$ given new $x^{(m+1)}$

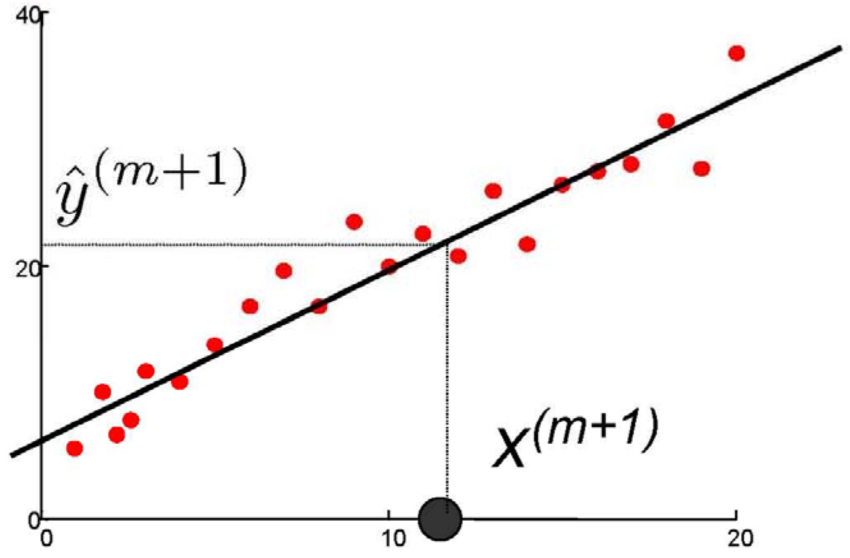
Regression functions



Nonlinear function

- Regression: given the observed data, estimate $y^{(m+1)}$ given new $x^{(m+1)}$

Linear regression



Linear function

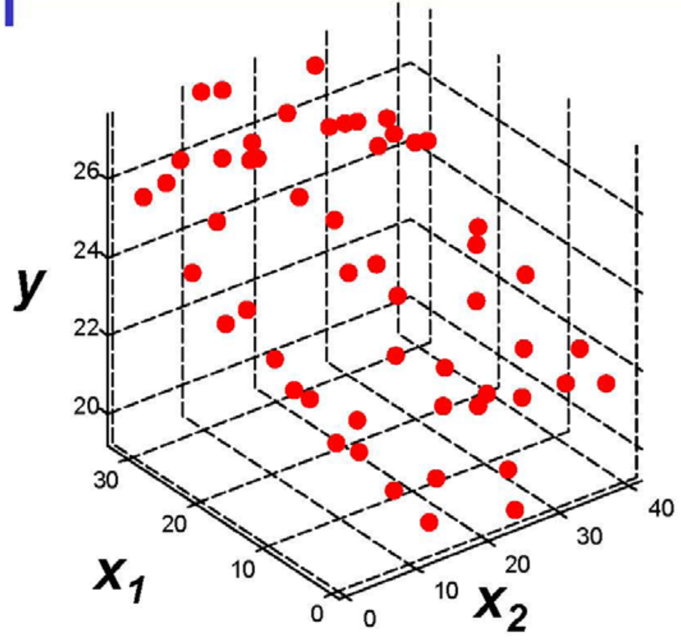
Line: y = ax + b

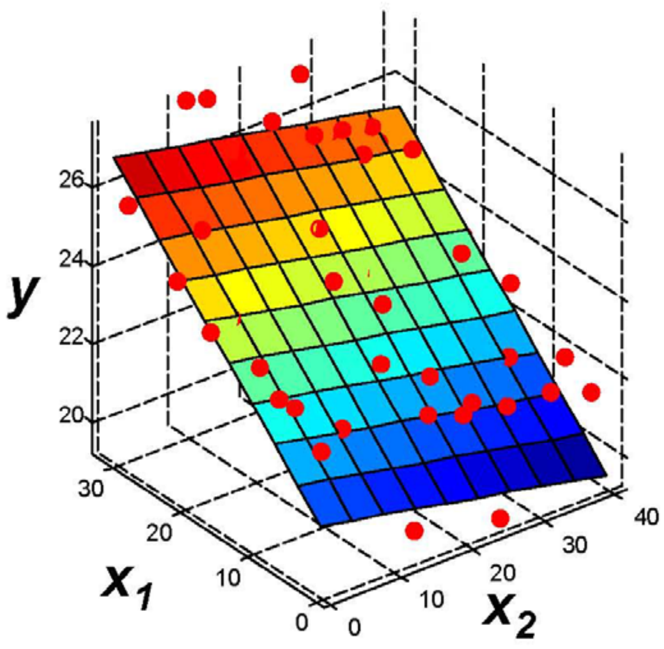
We'll write

$$\hat{y}(x) = \theta_0 + \theta_1 x$$

- Regression: given the observed data, estimate $y^{(m+1)}$ given new $x^{(m+1)}$

More dimensions?





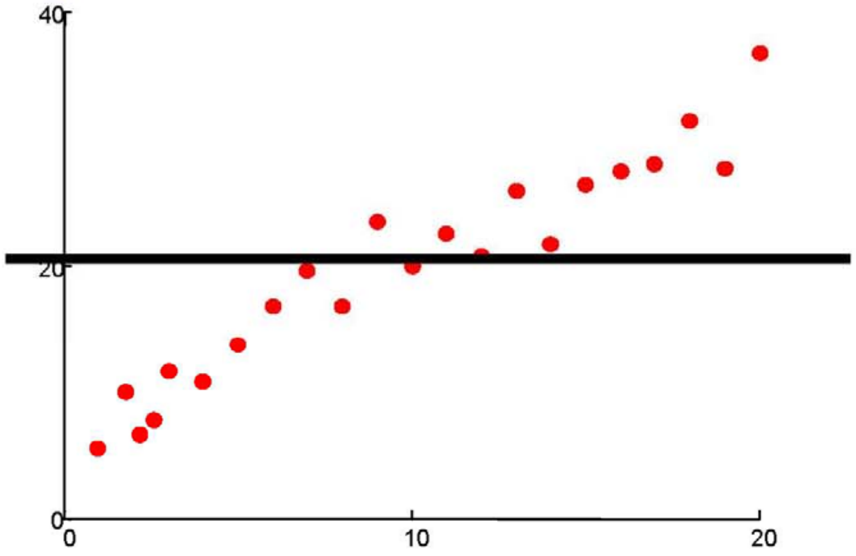
$$\hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$\underline{\theta} = [\theta_0 \ \theta_1 \ \theta_2]$$

$$\hat{y}(x) = \underline{\theta} \cdot \underline{x}^T$$

$$\underline{x} = [1 \ x_1 \ x_2]$$

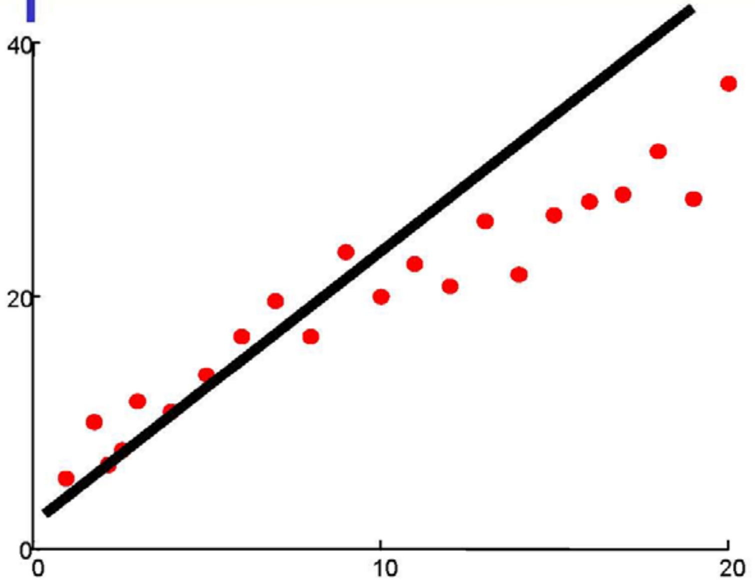
Measuring error



$$\hat{y}(x) = \theta_0 + \theta_1 x$$

- What makes a good predictor?

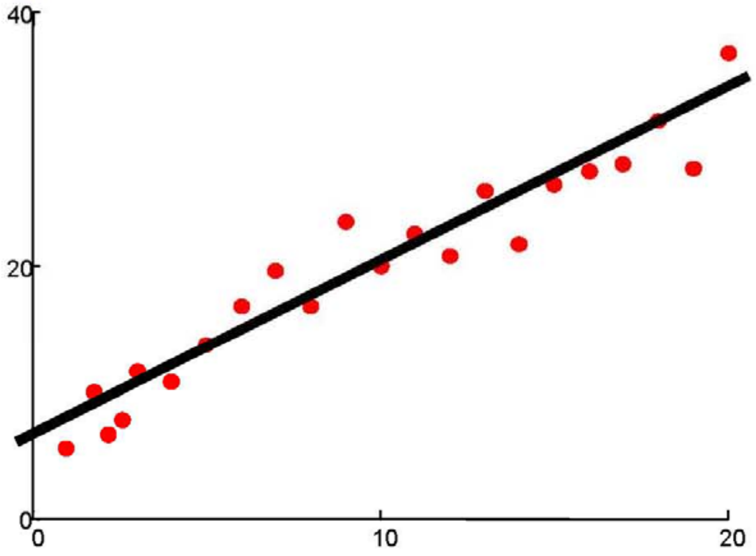
Measuring error



$$\hat{y}(x) = \theta_0 + \theta_1 x$$

- What makes a good predictor?

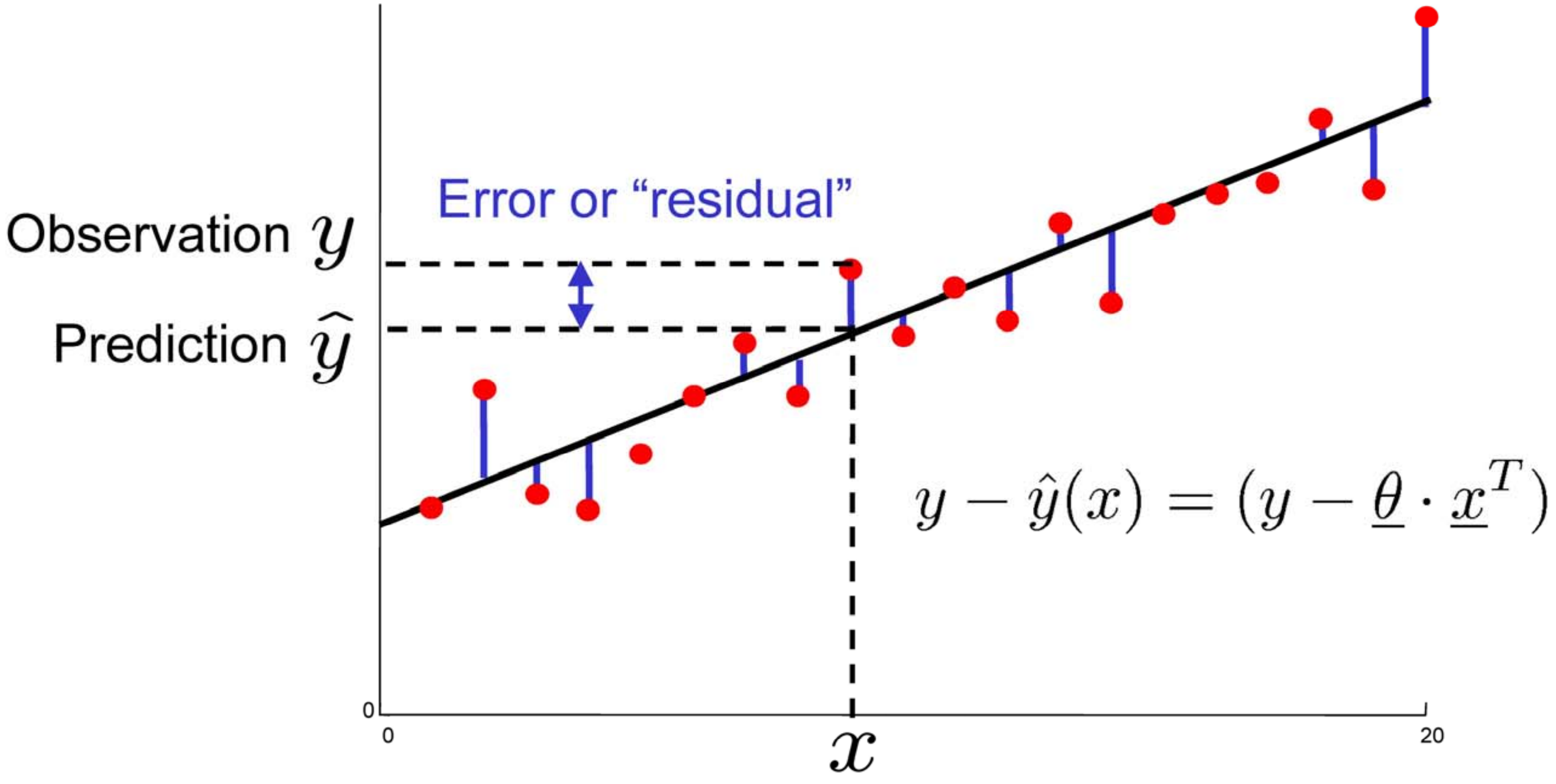
Measuring error



$$\hat{y}(x) = \theta_0 + \theta_1 x$$

- What makes a good predictor?

Measuring error


Error or "residual"
Observation y
Prediction $\widehat{y}$
0
0
20
x
$y - \hat{y}(x) = (y - \underline{\theta} \cdot \underline{x}^T)$

Notation

- Variables X, Y….. with values x, y (lower case)
  - Vectors indicated by $\underline{X}$
- Components of X indicated by X_j with values x_j
- "Matrix" data set D with n rows and p columns
  - jth column contains values for variable X_j
  - ith row contains a vector of measurements on object i, indicated by x(i)
  - The jth measurement value for the ith object is $x_j(i)$
- Unknown parameter for a model = θ
  - Can also use other Greek letters, like α, β, δ, γ
  - Vector of parameters = $\underline{\theta}$

Multivariate Linear Regression

- Task: predict real-valued Y, given real-valued vector $\underline{X}$

- Score function, e.g., least squares is often used

$$S(\underline{\theta}) = \Sigma_i \, [\underset{\text{target value}}{\underset{\uparrow}{y(i)}} - f(\underset{\text{predicted value}}{\underset{\nwarrow}{\underline{x}(i)}} \, ; \, \underline{\theta}) \,]^2$$

- Model structure: e.g., linear $f(\underline{x} \, ; \, \underline{\theta}) = \alpha_0 + \Sigma \, \alpha_j \, x_j$

- Model parameters = $\underline{\theta} = \{\alpha_0, \alpha_1, \ldots\ldots \alpha_p \}$

Note that we can write

$$S(\theta) = \Sigma_i \, [y(i) - \Sigma \, \alpha_j \, x_j]^2$$

$$= \Sigma_i \, e_i^2$$

$$= \underline{e}' \, \underline{e}$$

where $\underline{e} = y - X \, \theta$

$$= (y - X \, \theta)' \, (y - X \, \theta)$$

y = N x 1 vector
of target values

N x (p+1) vector
of input values

(p+1) x 1 vector
of parameter values

$S(\theta) = \Sigma e^2 = e' e = (y - X\theta)' (y - X\theta)$

$= y' y - \theta' X' y - y' X \theta + \theta' X' X \theta$

$= y' y - 2 \theta' X' y + \theta' X' X \theta$

Taking derivative of $S(\theta)$ with respect to the components of θ gives....

$dS/d\theta = -2 X' y + 2 X' X \theta$

Set this to 0 to find the extremum (minimum) of S as a function of θ ...

Set to 0 to find the extremum (minimum) of S as a function of θ ...

$\Rightarrow$ $-2X'y + 2X'X\theta = 0$

$\Rightarrow$ $X'X\theta = X'y$ (known in statistics as the Normal Equations)

Letting $X'X = C$, and $X'y = b$,
we have $C\theta = b$, i.e., a set of linear equations

We could solve this directly, e.g., by matrix inversion

$$\theta = C^{-1}b = (X'X)^{-1}X'y$$

Solving for the θ's

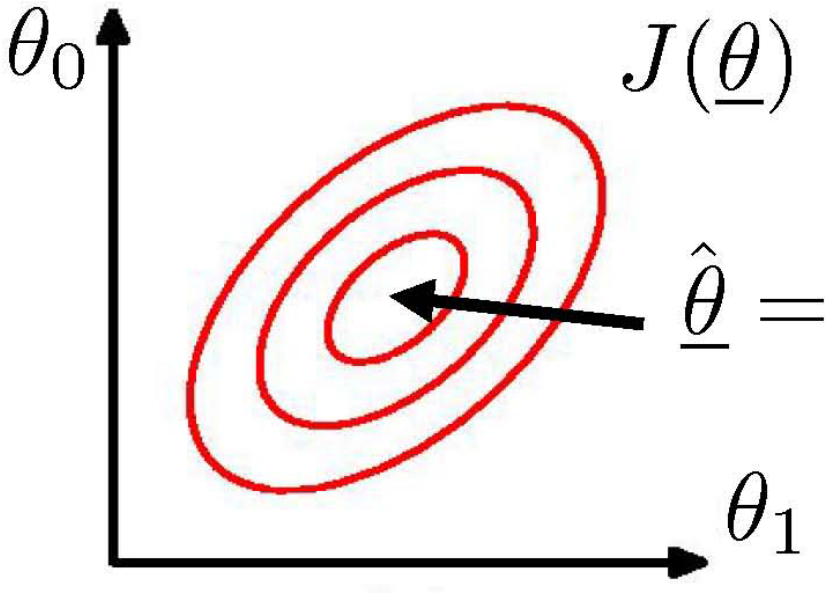
- Problem is equivalent to inverting X' X matrix
  - Inverse does not exist if matrix is not of full rank
    - E.g., if 1 column is a linear combination of another (collinearity)
    - Note that X'X is closely related to the covariance of the X data
      - So we are in trouble if 2 or more variables are perfectly correlated
    - Numerical problems can also occur if variables are almost collinear

- Equivalent to solving a system of p linear equations
  - Many good numerical methods for doing this, e.g.,
    - Gaussian elimination, LU decomposition, etc
  - These are numerically more stable than direct inversion

- Alternative: gradient descent
  - Compute gradient and move downhill

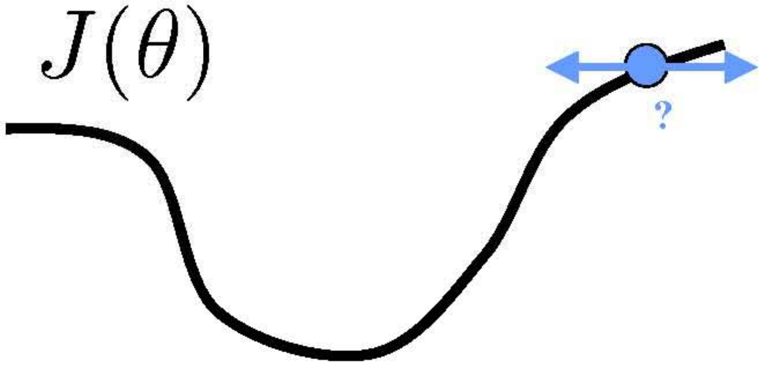
Finding good parameters

- Want to find parameters which minimize our error…
- Think of a cost “surface”: error residual for that θ…



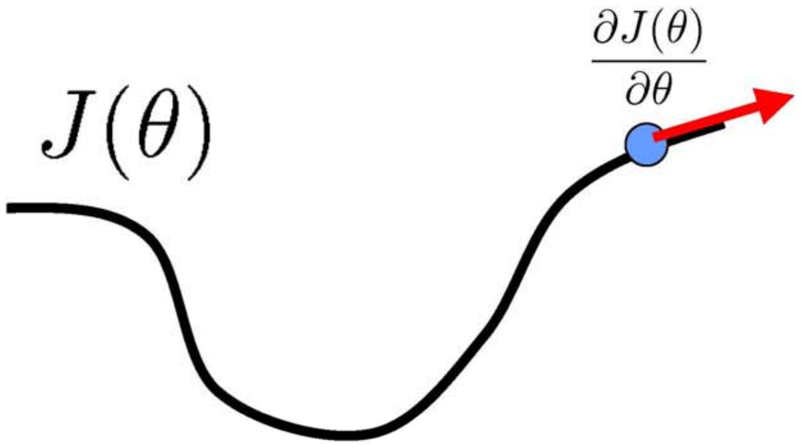
$$\hat{\underline{\theta}} = \arg\min_{\underline{\theta}} J(\underline{\theta})$$

Gradient descent



- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing

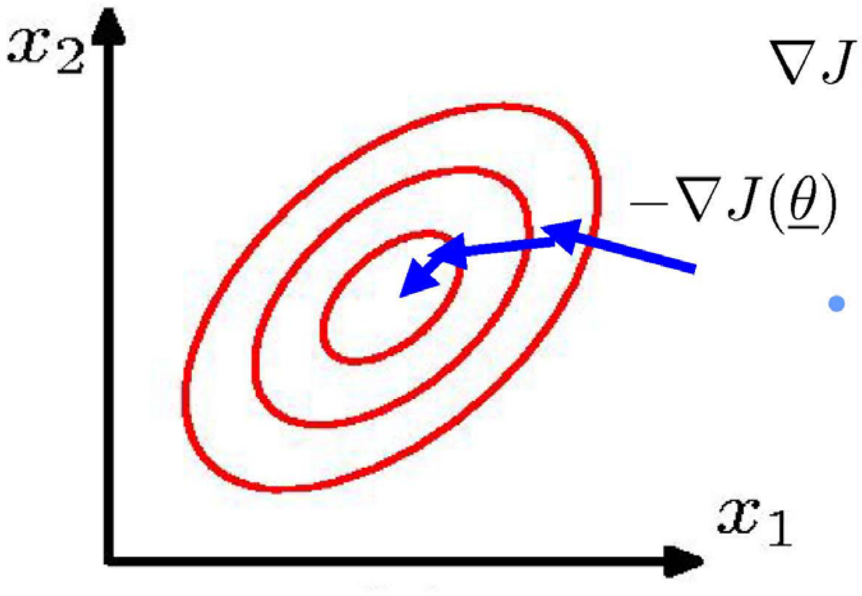
Gradient descent



- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing
- Gradient $\frac{\partial J(\theta)}{\partial \theta}$

- Positive => increasing
- Negative => decreasing

Gradient descent in more dimensions



- Gradient vector

$$\nabla J(\underline{\theta}) = \left[\frac{\partial J(\underline{\theta})}{\partial \theta_0} \quad \frac{\partial J(\underline{\theta})}{\partial \theta_1} \quad \ldots \right]$$

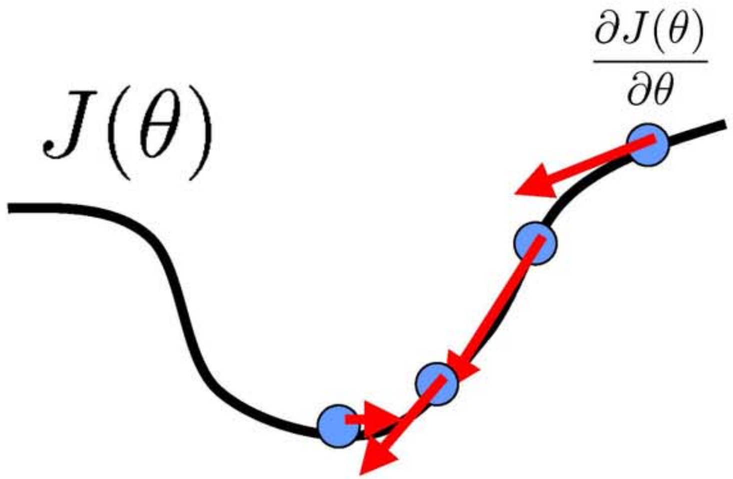
- Indicates direction of steepest ascent

  (negative = steepest descent)

Gradient descent

- Initialization
- Step size
  - Can change as a function of iteration
- Gradient direction
- Stopping condition

```
Initialize $\theta$
Do {
  $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$
} while ( $\alpha \|\nabla J\| > \epsilon$ )
```



Gradient for the SSE

- SSE $$J(\underline{\theta}) = \frac{1}{2}\sum_j (y^{(j)} - \underline{\theta}\cdot \underline{x}^{(j)\,T})^2$$

- ∇ J = ?

$$J(\underline{\theta}) = \frac{1}{2}\sum_j (y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \ldots)^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}\frac{1}{2}\sum_j (\ g(\theta)\ )^2$$

$$= \frac{1}{2}\sum_j \ \frac{\partial}{\partial \theta_0}(\ g(\theta)\ )^2$$

$$= \frac{1}{2}\sum_j \ 2g(\theta)\ \frac{\partial}{\partial \theta_0} g(\theta)$$

$$\frac{\partial}{\partial \theta_0} g(\theta) = \underbrace{\frac{\partial}{\partial \theta_0} y^{(j)}}_{\mathbf{0}} - \frac{\partial}{\partial \theta_0}\theta_0 x_0^{(j)} - \underbrace{\frac{\partial}{\partial \theta_0}\theta_1 x_1^{(j)}}_{\mathbf{0}} - \ldots$$

$$= -x_0^{(j)}$$

Gradient descent

- Initialization
- Step size
  - Can change as a function of iteration
- Gradient direction
- Stopping condition

```
Initialize $\theta$
Do {
  $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$
} while ( $\alpha||\nabla J|| > \epsilon$ )
```

$$J(\underline{\theta}) = \frac{1}{2} \sum_j (y^{(j)} - \underline{\theta} \cdot {\underline{x}^{(j)}}^T)^2$$

$$\nabla J(\underline{\theta}) = - \sum_j (y^{(j)} \underbrace{- \underline{\theta} \cdot}_{\text{Error magnitude \& direction for datum j}} {\underline{x}^{(j)}}^T) \cdot [\underbrace{x_0^{(j)} x_1^{(j)}}_{\text{Sensitivity to each } \theta_i} \ldots]$$

Comments on Multivariate Linear Regression

- Prediction model is a linear function of the parameters

- Score function: quadratic in predictions and parameters
  - $\Rightarrow$ Derivative of score is linear in the parameters
  - $\Rightarrow$ Leads to a linear algebra optimization problem, i.e., $C\,\theta = b$

- Model structure is simple….
  - – p-1 dimensional hyperplane in p-dimensions
  - – Linear weights => interpretability

- Often useful as a baseline model
  - – e.g., to compare more complex models to

- Note: even if it's the wrong model for the data (e.g., a poor fit) it can still be useful for prediction

Limitations of Linear Regression

- True relationship of X and Y might be non-linear
  - Suggests generalizations to non-linear models
- Complexity:
  - $O(N p^2 + p^3)$ - problematic for large p
- Correlation/Collinearity among the X variables
  - Can cause numerical instability (C may be ill-conditioned)
  - Problems in interpretability (identifiability)
- Includes all variables in the model…
  - But what if p=1000 and only 3 variables are actually related to Y?

Non-linear models, but linear in parameters

- We can add additional polynomial terms in our equations, e.g., all "2nd order" terms

$$f(\underline{x} ; \underline{\theta}) = \alpha_0 + \sum \alpha_j x_j + \sum \beta_{ij} x_i x_j$$

- Note that it is a non-linear functional form, but it is linear in the parameters (so still referred to as "linear regression")
  – We can just treat the $x_i x_j$ terms as additional fixed inputs
  – In fact we can add in any non-linear input functions!, e.g.

$$f(\underline{x} ; \underline{\theta}) = \alpha_0 + \sum \alpha_j f_j(\underline{x})$$

Comments:

- Exact same linear algebra for optimization (same math)
- Number of parameters has now exploded -> greater chance of overfitting
  - Ideally would like to select only the useful quadratic terms
  - Can generalize this idea to higher-order interactions

Non-linear (both model and parameters)

- We can generalize further to models that are nonlinear in all aspects

$$f(\underline{x} ; \underline{\theta}) = \alpha_0 + \sum \alpha_k \, g_k(\beta_{k0} + \sum \beta_{kj} \, x_j)$$

where the g's are non-linear functions with fixed functional forms.

In machine learning this is called a neural network

In statistics this might be referred to as a generalized linear model or projection-pursuit regression

For almost any score function of interest, e.g., squared error, the score function is a non-linear function of the parameters.

Closed form (analytical) solutions are rare.

Thus, we have a multivariate non-linear optimization problem
(which may be quite difficult!)

Optimization in the Non-Linear Case

- We seek the minimum of a function in d dimensions, where d is the number of parameters (d could be large!)

- There are a multitude of heuristic search techniques
  - Steepest descent (follow the gradient)
  - Newton methods (use 2nd derivative information)
  - Conjugate gradient
  - Line search
  - Stochastic search
  - Genetic algorithms

- Two cases:
  - Convex (nice -> means a single global optimum)
  - Non-convex (multiple local optima => need multiple starts)

Other non-linear models

- Splines
  - "patch" together different low-order polynomials over different parts of the x-space
  - Works well in 1 dimension, less well in higher dimensions

- Memory-based models

  $y' = \sum w_{(x',x)}\, y,$ where y's are from the training data

  $w_{(x',x)}$ = function of distance of x from x'

- Local linear regression

  $y' = \alpha_0 + \sum \alpha_j x_j$, where the alpha's are fit at prediction time just to the (y,x) pairs that are close to x'

Local linear regression will be in HW2

Selecting the k best predictor variables

- Linear regression: find the best subset of k variables to put in model
  - This is a generic problem when p is large
    (arises with all types of models, not just linear regression)

- Now we have models with different complexity..
  - E.g., p models with a single variable
  - p(p-1)/2 models with 2 variables, etc…
  - 2^p possible models in total
    - Can think of space of models as a lattice
  - Note that when we add or delete a variable, the optimal weights on the other variables will change in general
    - k best is not the same as the best k individual variables

- Aside: what does “best” mean here? (will return to this shortly…)

Search Problem

- How can we search over all 2^p possible models?
  - exhaustive search is clearly infeasible

  - Heuristic search is used to search over model space:
    - Forward search (greedy)
    - Backward search (greedy)
    - Generalizations (add or delete)
      - Think of operators in search space
    - Branch and bound techniques

  - This type of variable selection problem is common to many data mining algorithms
    - Outer loop that searches over variable combinations
    - Inner loop that evaluates each combination

Empirical Learning

- Squared Error score (as an example: we could use other scores)

  $$S(\underline{\theta}) = \sum_i [y(i) - f(\underline{x}(i) ; \underline{\theta})]^2$$

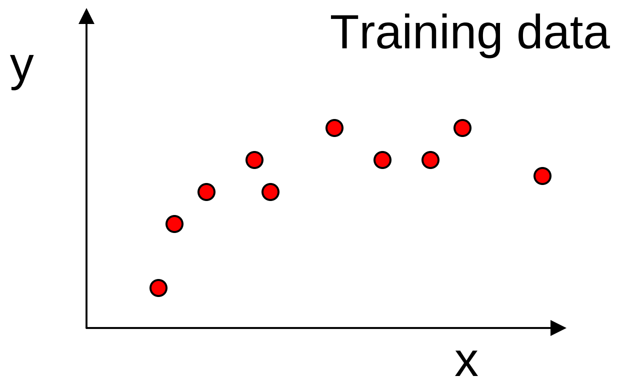
  where $S(\underline{\theta})$ is defined on the training data D

- We are really interested in finding the $f(x; \underline{\theta})$ that best predicts y on **future** data, i.e., minimizing

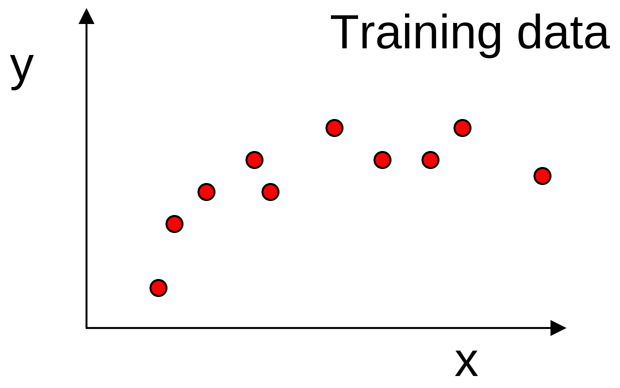
  $$E [S] = E [y - f(\underline{x} ; \underline{\theta})]^2$$ (where the expectation is over future data)

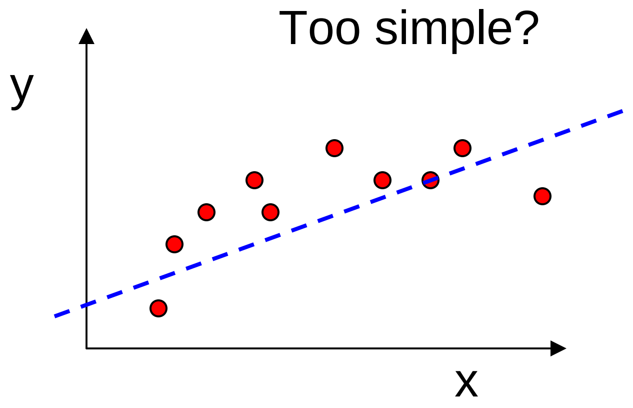
- Empirical learning
  - Minimize $S(\underline{\theta})$ on the training data D_{train}
  - If D_{train} is large and model is simple we are assuming that the best f on training data is also the best predictor f on future test data D_{test}

Complexity versus Goodness of Fit

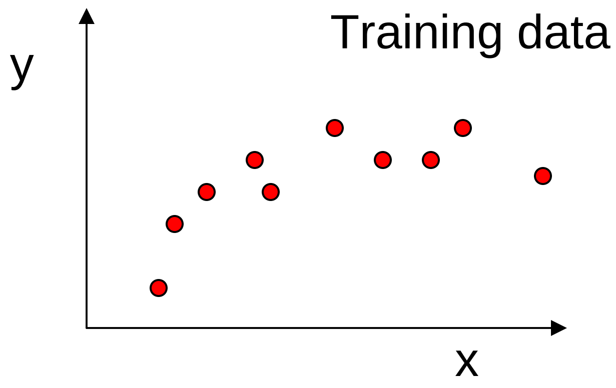


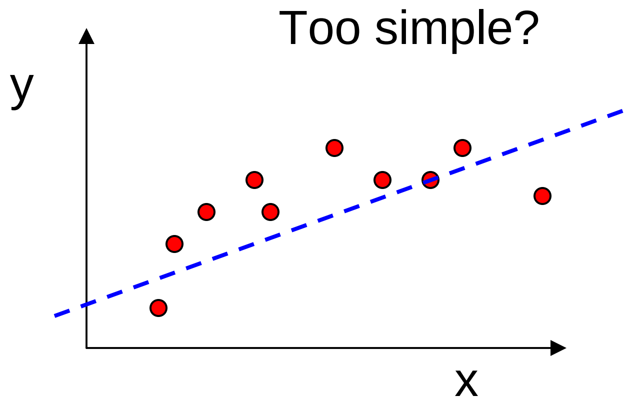
Complexity versus Goodness of Fit

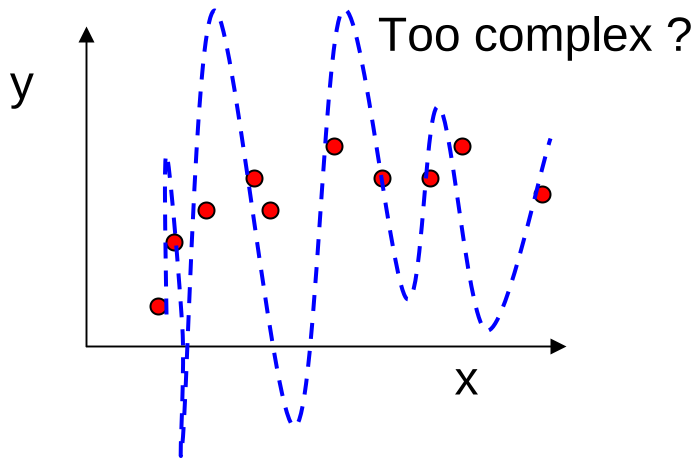




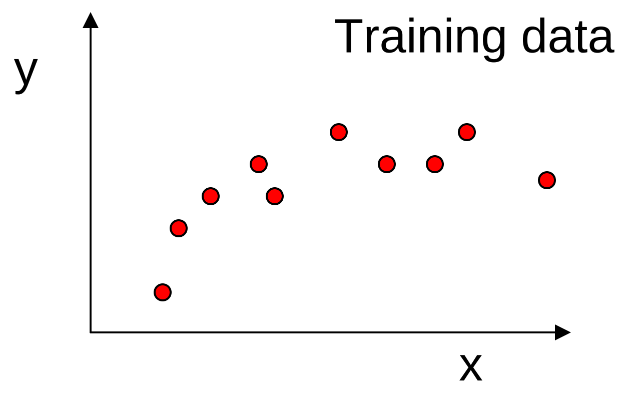
Complexity versus Goodness of Fit

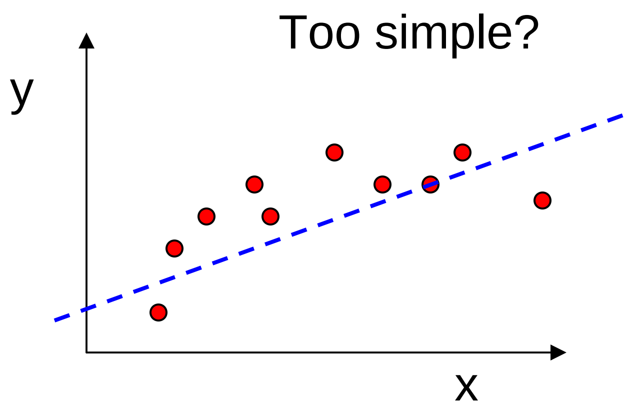


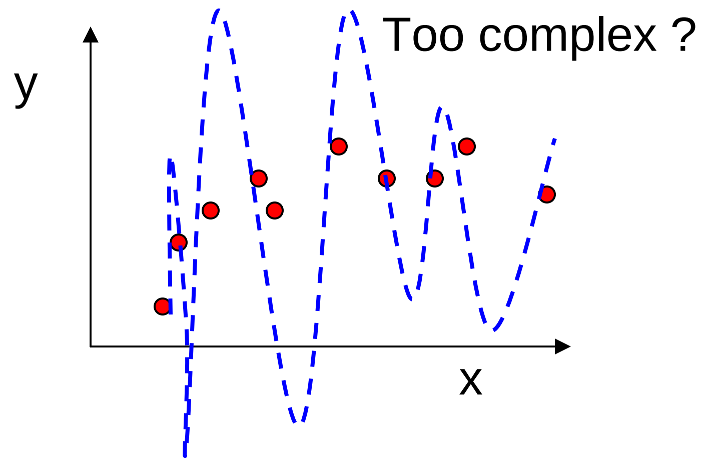


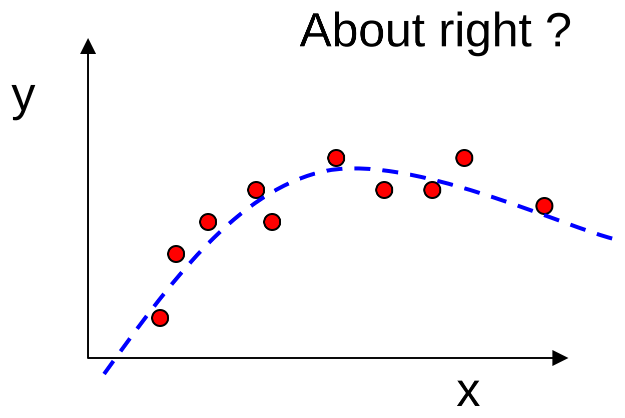


Complexity versus Goodness of Fit

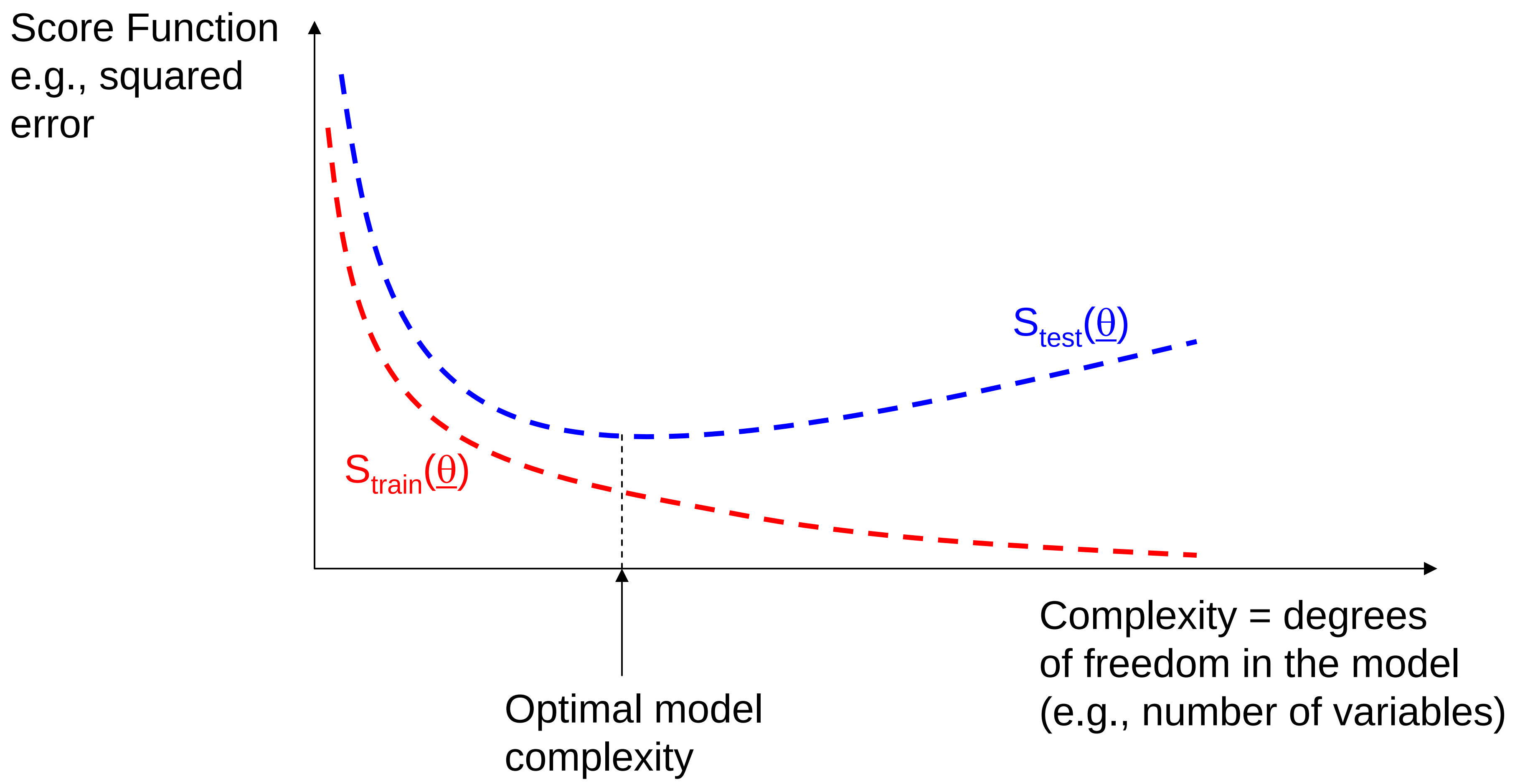




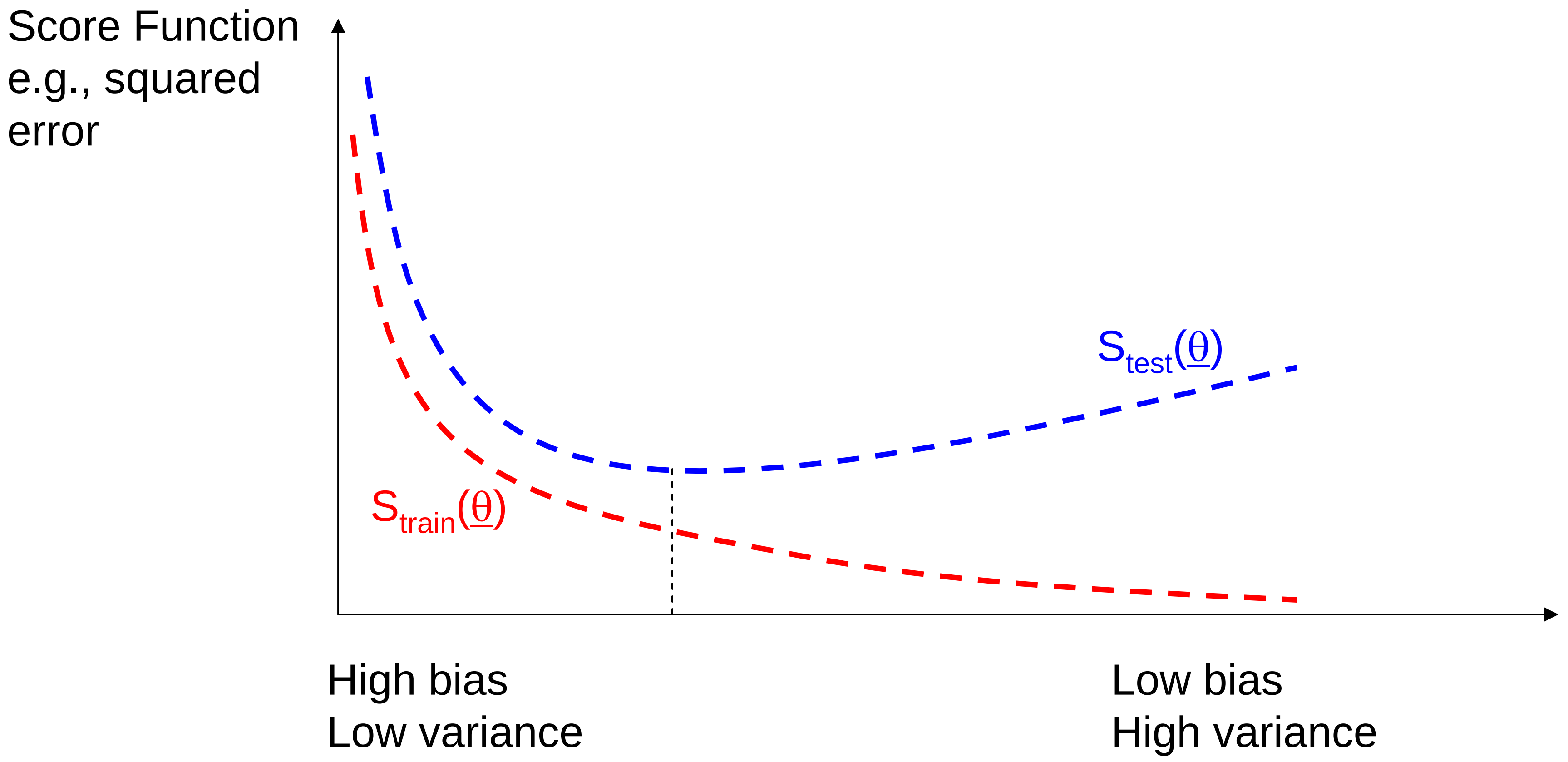




Complexity and Generalization



Complexity and Generalization



Defining what "best" means

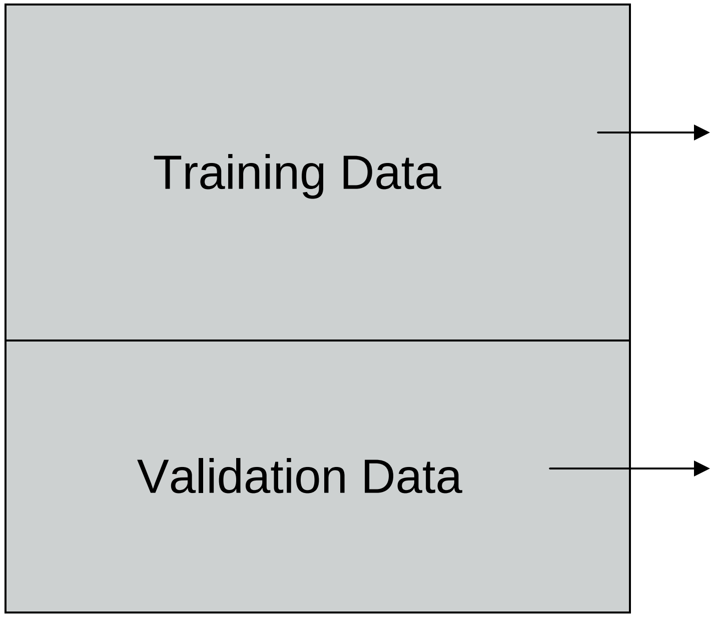
- How do we measure "best"?
  - Best performance on the training data?
    - K = p will be best (i.e., use all variables), e.g., p=10,000
    - So this is not useful in general
  - Performance on the training data will in general be optimistic

- Practical Alternatives:
  - Measure performance on a single validation set
  - Measure performance using multiple validation sets
    - Cross-validation
  - Add a penalty term to the score function that "corrects" for optimism
    - E.g., "regularized" regression: SSE + λ sum of weights squared

Training Data

Use this data to find the best $\underline{\theta}$ for each model $f_k(\underline{x} ; \underline{\theta})$

Validation Data

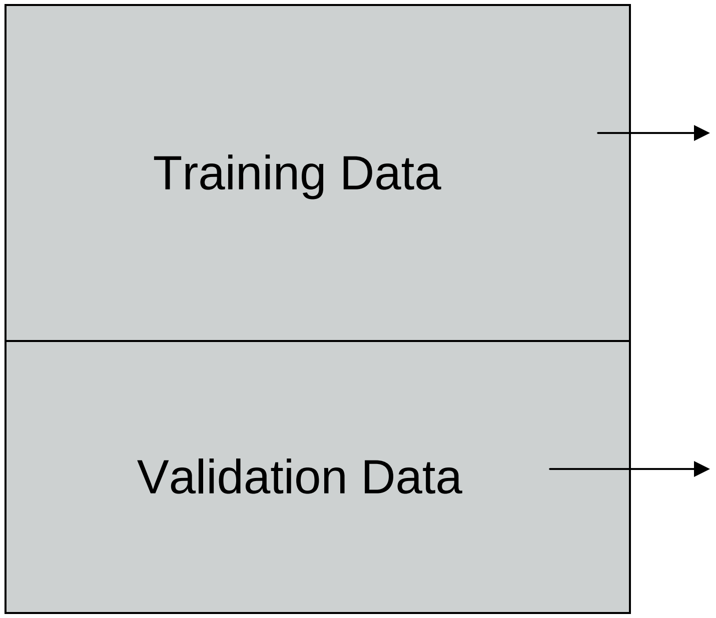


Use this data to find the best $\underline{\theta}$ for each model $f_k(\underline{x} ; \underline{\theta})$

Use this data to
(1) calculate an estimate of $S_k(\underline{\theta})$ for each $f_k(\underline{x} ; \underline{\theta})$ and
(2) select $k^* = \arg\min_k S_k(\underline{\theta})$

Validation Data

can generalize to **cross-validation….**



Use this data to find the best $\underline{\theta}$ for each model $f_k(\underline{x} ; \underline{\theta})$

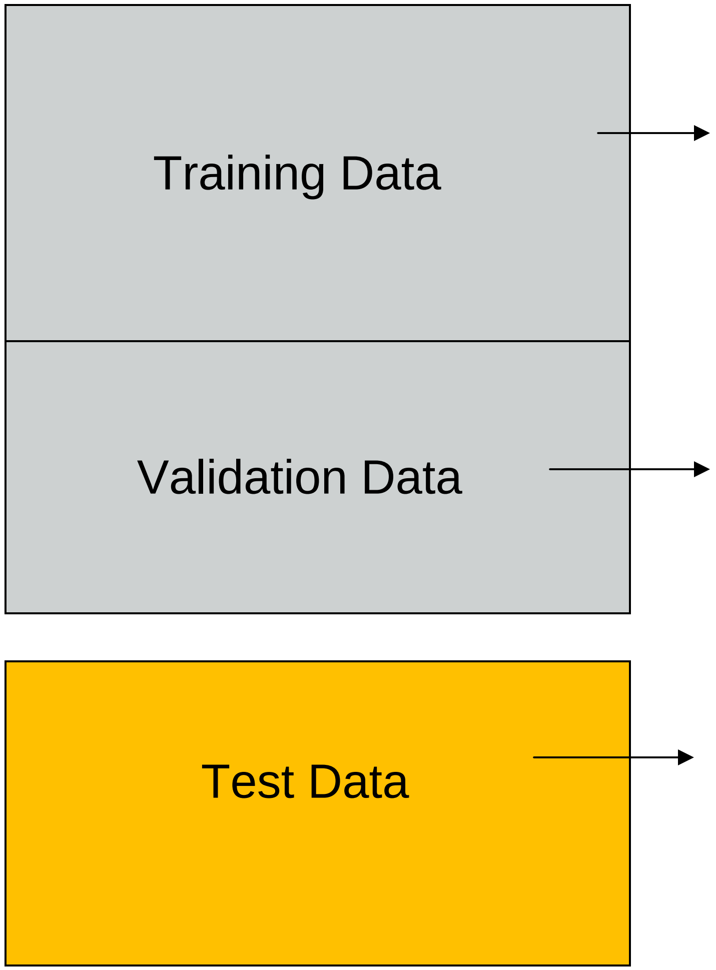
Use this data to
(1) calculate an estimate of $S_k(\underline{\theta})$ for each $f_k(\underline{x} ; \underline{\theta})$ and
(2) select $k^* = \arg\min_k S_k(\underline{\theta})$

2 different (but related) issues here

1. Finding the function f that minimizes $S(\underline{\theta})$ for future data

2. Getting a good estimate of $S(\underline{\theta})$, using the chosen function, on future data,
   - e.g., we might have selected the best function f, but our estimate of its performance will be optimistically biased if our estimate of the score uses any of the same data used to fit and select the model.

Test Data



Use this data to find the best $\underline{\theta}$ for each model $f_k(\underline{x}\ ;\ \underline{\theta})$

Use this data to
(1) calculate an estimate of $S_k(\underline{\theta})$ for each $f_k(\underline{x}\ ;\ \underline{\theta})$ and
(2) select $k^* = \arg\min_k S_k(\underline{\theta})$

Use this data to calculate an unbiased estimate of $S_{k^*}(\underline{\theta})$ for the selected model

Another Approach with Many Predictors: Regularization

- Modified score function:

$$S_\lambda(\underline{\theta}) = \Sigma_i \, [y(i) - f(\underline{x}(i) \, ; \, \underline{\theta}) \,]^2 + \lambda \, \Sigma \, \theta_j^{\,2}$$

- The second term is for “regularization”
  - When we minimize -> encourages keeping the θ_j‘s near 0
  - Bayesian interpretation: minimizing - log P(data|q) - log P(q)

- L1 regularization

$$S_\lambda(\underline{\theta}) = \Sigma_i \, [y(i) - f(\underline{x}(i) \, ; \, \underline{\theta}) \,]^2 + \lambda \, \Sigma \, | \, \theta_j \, |$$

(basis of popular “Lasso” method, e.g., see Rob Tibshirani’s page on lasso methods: http://www-stat.stanford.edu/~tibs/lasso.html)

Time-series prediction as regression

- Measurements over time $x_1, \ldots\ldots x_t$
- We want to predict x_{t+1} given $x_1, \ldots\ldots x_t$
- Autoregressive model

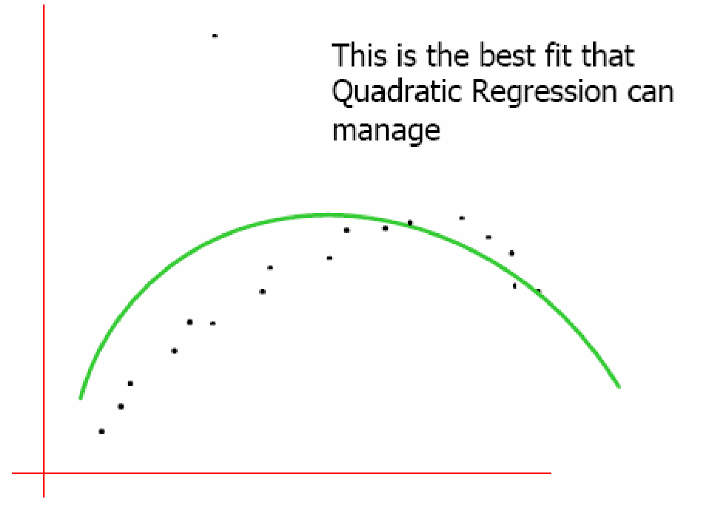
$$x_{t+1} = f(\, x_1, \ldots\ldots x_t \,;\, \underline{\theta}\,) \;=\; \Sigma\, \alpha_k\, x_{t-k}$$

  - Number of coefficients K = memory of the model
  - Can take advantage of regression techniques in general to solve this problem (e.g., linear in parameters, score function = squared error, etc)

- Generalizations
  - Vector x
  - Non-linear function instead of linear
  - Add in terms for time-trend (linear, seasonal), for “jumps”, etc

Other aspects of regression

- Diagnostics
  - Useful in low dimensions
- Weighted regression
  - Useful when rows have different weights
- Different score functions
  - E.g. absolute error, or additive noise varies as a function of x
- Predicting y values constrained to a certain range, e.g., $y > 0$, or $0 < y < 1$
- Predicting binary y values
  - Regression as a generalization of classification



Generalized Linear Models (GLMs)

(McCullagh and Nelder, 1989)

- $g(y) = \quad u(x) = \alpha_0 + \sum \alpha_j x_j$
  - Where g [ ] is a "link" function
  - u(x) is a linear function of the vector x

- Examples:
  - g = identity function -> linear regression
  - Logistic regression: $g(y) = \log(y / 1\text{-}y) \; = \alpha_0 + \sum \alpha_j x_j$
  - Logarithmic link: $g(y) = \log(y) = \alpha_0 + \sum \alpha_j x_j$
  - GLMs are widely used in statistics
  - Details of learning/fitting algorithm depend on the specifics of the link function

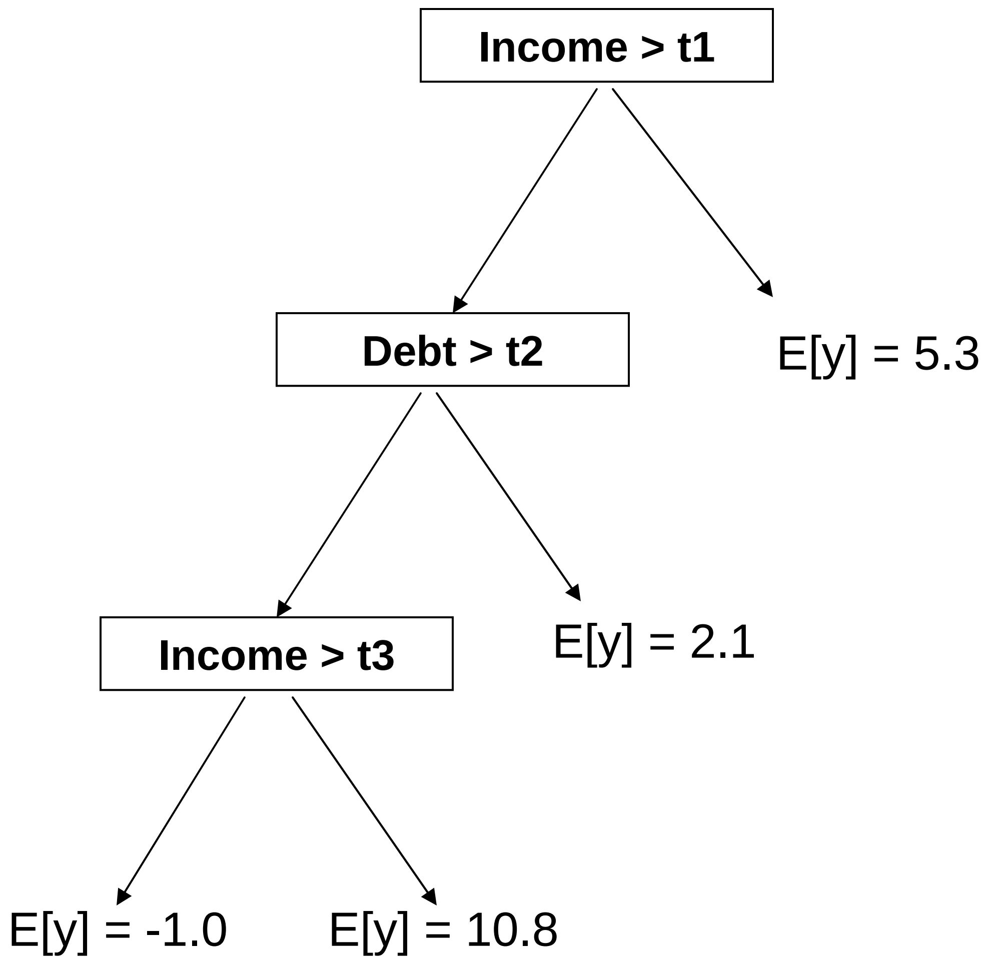
Tree-Structured Regression

- Functional form of model is a “regression tree”
  - Univariate thresholds at internal nodes
  - Constant or linear surfaces at the leaf nodes
  - Yields piecewise constant (or linear) surface
  - (like classification tree, but for regression)

- Very crude functional form…. but
  - Can be very useful in high-dimensional problems
  - Can useful for interpretation
  - Can handle combinations of real and categorical variables

- Search problem
  - Finding the optimal tree is intractable
  - Practice: greedy algorithms

Simple example of Tree Model



Greedy Search for Learning Regression Trees

- Binary_node_splitting, real-valued variables
  - For each variable x_j
    - For each possible threshold t_{jk} , compute

$$MSE(y; t_{jk}) = P(x \leq t_{jk}) \underbrace{MSE(y|x \leq t_{jk})}_{\text{MSE in left branch}} + P(x > t_{jk}) \underbrace{MSE(y|x > t_{jk})}_{\text{MSE in right branch}}$$

    - Select t_{jk} with the lowest MSE for that variable
  - Select variable x_j and t_{jk} with the lowest MSE
  - Split the training data into the 2 branches
  - For each branch
    - If leaf-node: prediction at this leaf node = mean value of y data points
    - If not: call binary_node_splitting recursively
- Time complexity?

Time-Complexity

p variables

O(N) thresholds

- Binary_node_splitting
  - For each variable x_j
    - For each possible threshold t_{jk} , compute

$$MSE(y;t_{jk}) = P(x \le t_{jk}) \underbrace{MSE(y|x \le t_{jk})}_{\text{MSE in left branch}} + P(x > t_{jk}) \underbrace{MSE(y|x > t_{jk})}_{\text{MSE in right branch}}$$

    - Select t_{jk} with the lowest MSE for that variable
  - Select variable x_j and t_{jk} with the lowest MSE
- O(p N logN) for the root node split
- For tree of depth L ??
  - Need to resort at each internal node, O(n' log n') …..

More on Regression Trees

- Greedy search algorithm
  - For each variable, find the best split point such the mean of Y either side of the split minimizes the mean-squared error
  - Select the variable with the minimum average error
    - Partition the data using the threshold
  - Recursively apply this selection procedure to each “branch”

- What size tree?
  - A full tree will likely overfit the data
  - Common methods for tree selection
    - Grow a large tree and select an appropriate subtree by Xvalidation
    - Grow a number of small fixed-sized trees and average their predictions

- Will discuss trees further in lectures on classification: very similar ideas used for building classification trees and regression trees

Model Averaging

- Can average over parameters and models
  - E.g., weighted linear combination of predictions from multiple models

$$y = \Sigma \; w_k \; y_k$$

  - Why? Any predictions from a point estimate of parameters or a single model has only a small chance of the being the best

  - Averaging makes our predictions more stable and less sensitive to random variations in a particular data set (good for less stable models like trees)

Model Averaging

- Model averaging flavors
  - Fully Bayesian: average over uncertainty in parameters and models
  - "empirical Bayesian": learn weights over multiple models
    - E.g., stacking and bagging (widely used in Netflix competition)
  - Build multiple simple models in a systematic way and combine them, e.g.,
    - Boosting: will say more about this in later lectures
    - Random forests (for trees): stochastically perturb the data, learn multiple trees, and then combine for prediction

Components of ML Algorithms

- Model Representation:
  - Determining the nature and structure of the representation to be used
- Score function
  - Measuring how well different representations fit the data
- Search/Optimization method
  - An algorithm to optimize the score function
- Data Management
  - Deciding what principles of data management are required to implement the algorithms efficiently.

Software

- MATLAB
  - Many free “toolboxes” on the Web for regression and prediction
  - e.g., see http://lib.stat.cmu.edu/matlab/
    and in particular the CompStats toolbox
- R
  - General purpose statistical computing environment (successor to S)
  - Free (!)
  - Widely used by statisticians, has a huge library of functions and visualization tools
- Commercial tools
  - SAS, Salford Systems, other statistical packages
  - Various data mining packages
  - Often are not programmable: offer a fixed menu of items

Useful References

N. R. Draper and H. Smith,
Applied Regression Analysis, 2nd edition,
Wiley, 1981
(the “bible” for classical regression methods in statistics)

T. Hastie, R. Tibshirani, and J. Friedman,
Elements of Statistical Learning, 2nd edition,
Springer Verlag, 2009
(statistically-oriented overview of modern ideas in regression and classification, mixes machine learning and statistics)