CS6510
Applied Machine Learning

Regression

7 Oct 2017

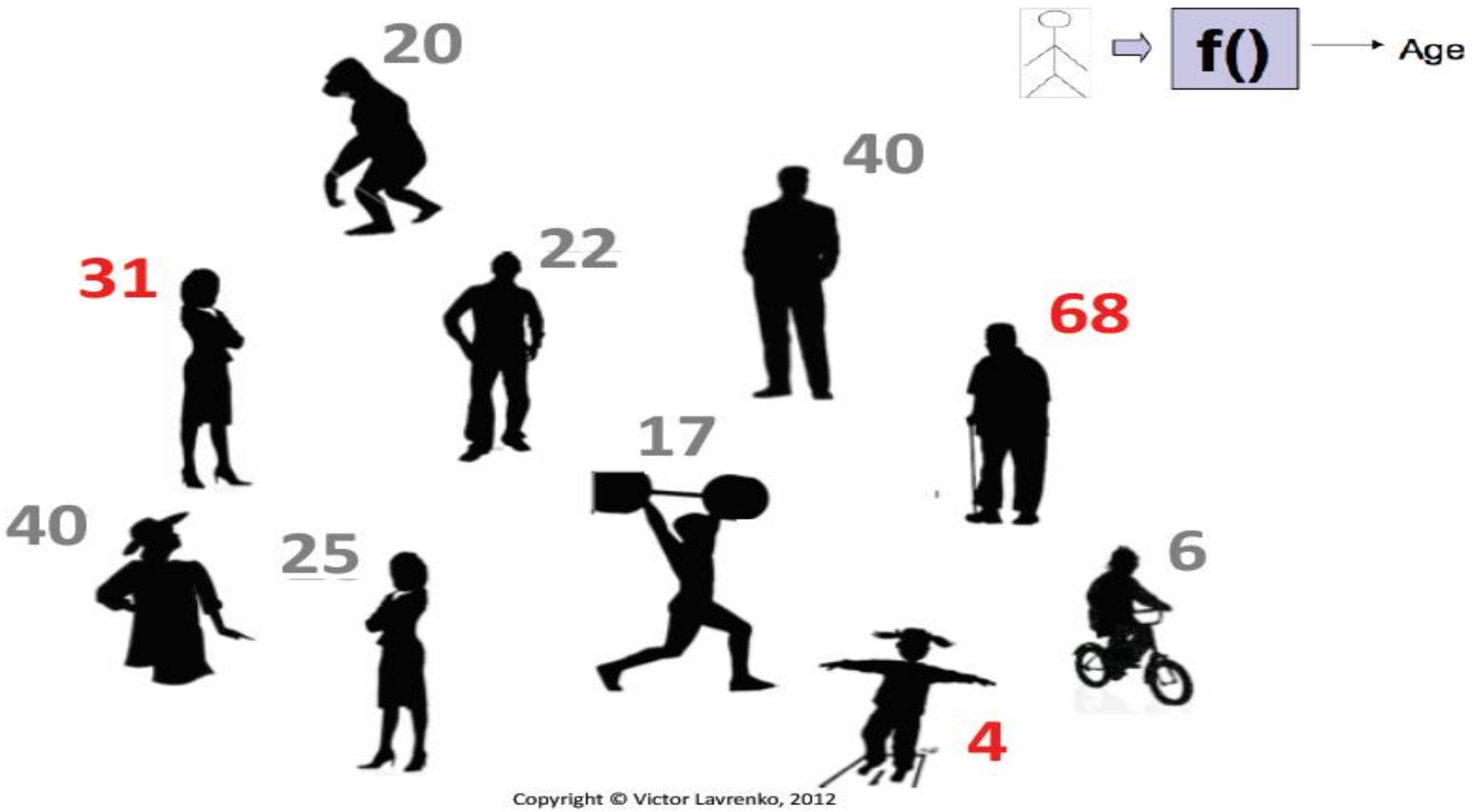
Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

ML Problems

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

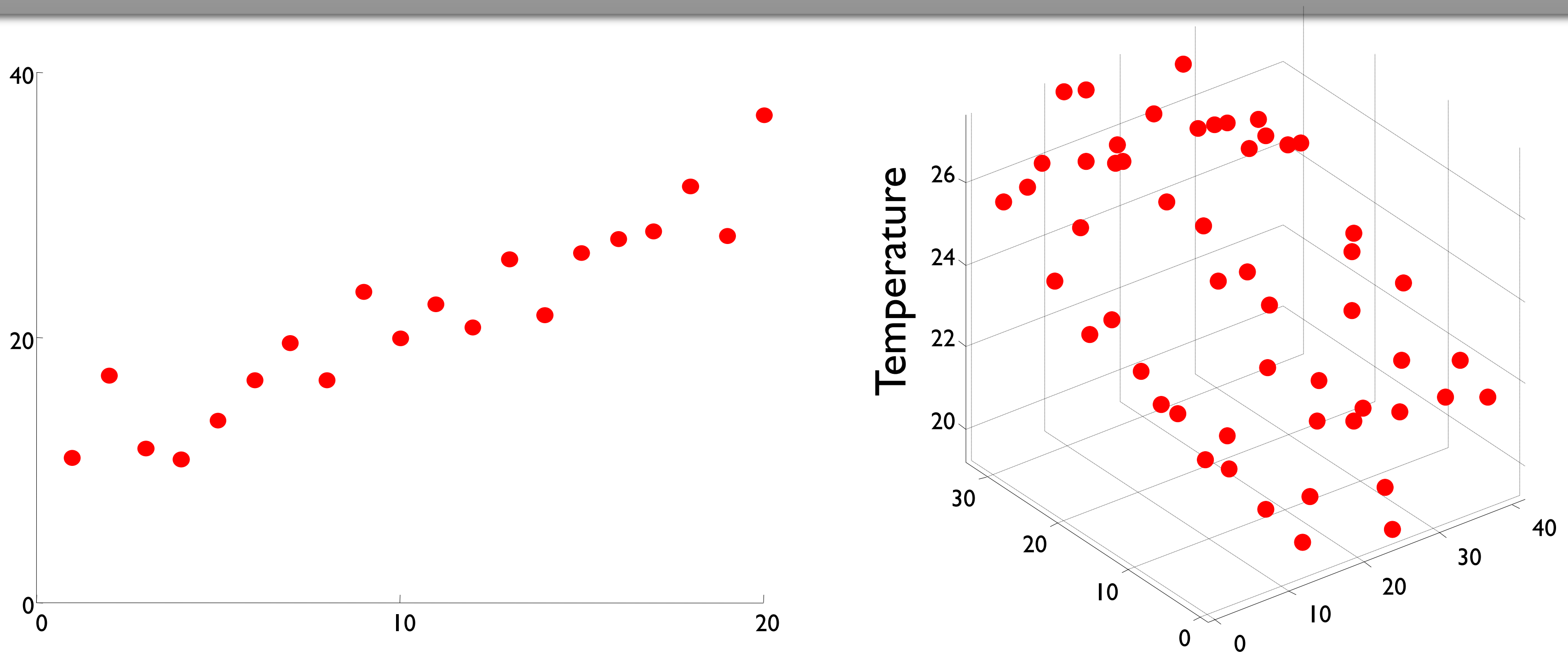
Regression (Supervised Learning)



Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

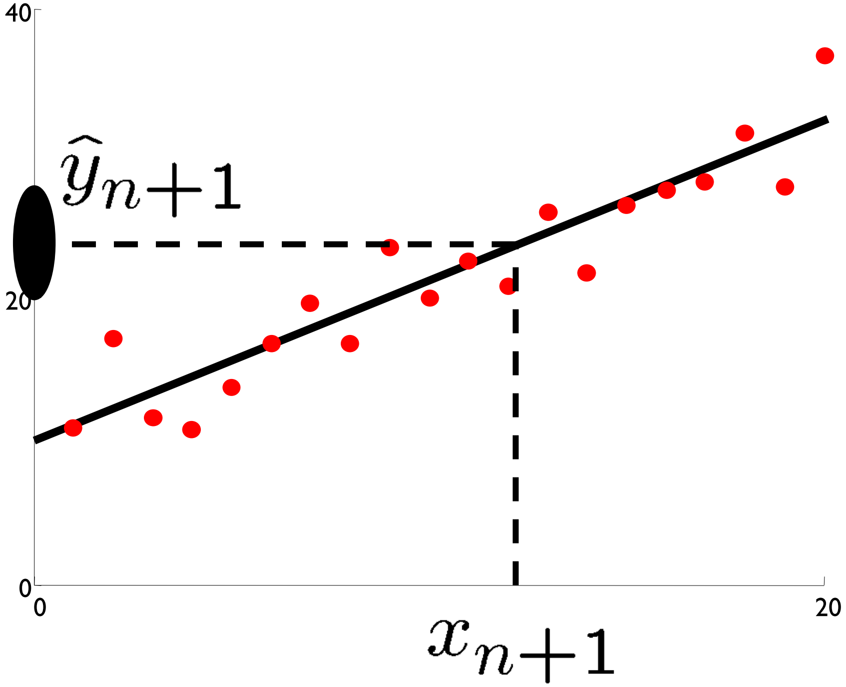
Linear Regression

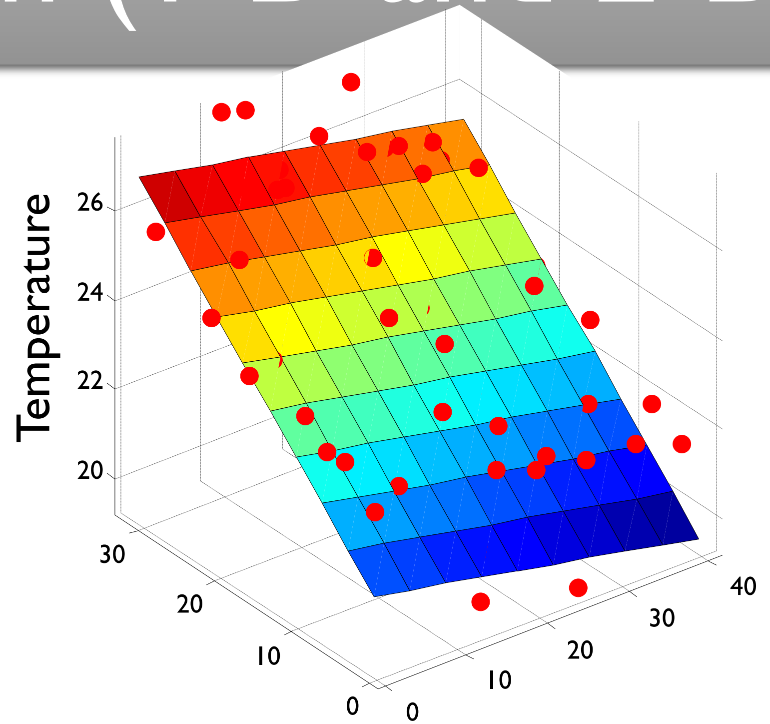


Given examples $(x_i, y_i)_{i=1\ldots n}$

Predict y_{n+1} given a new point x_{n+1}

Linear Regression (1-D and 2-D)





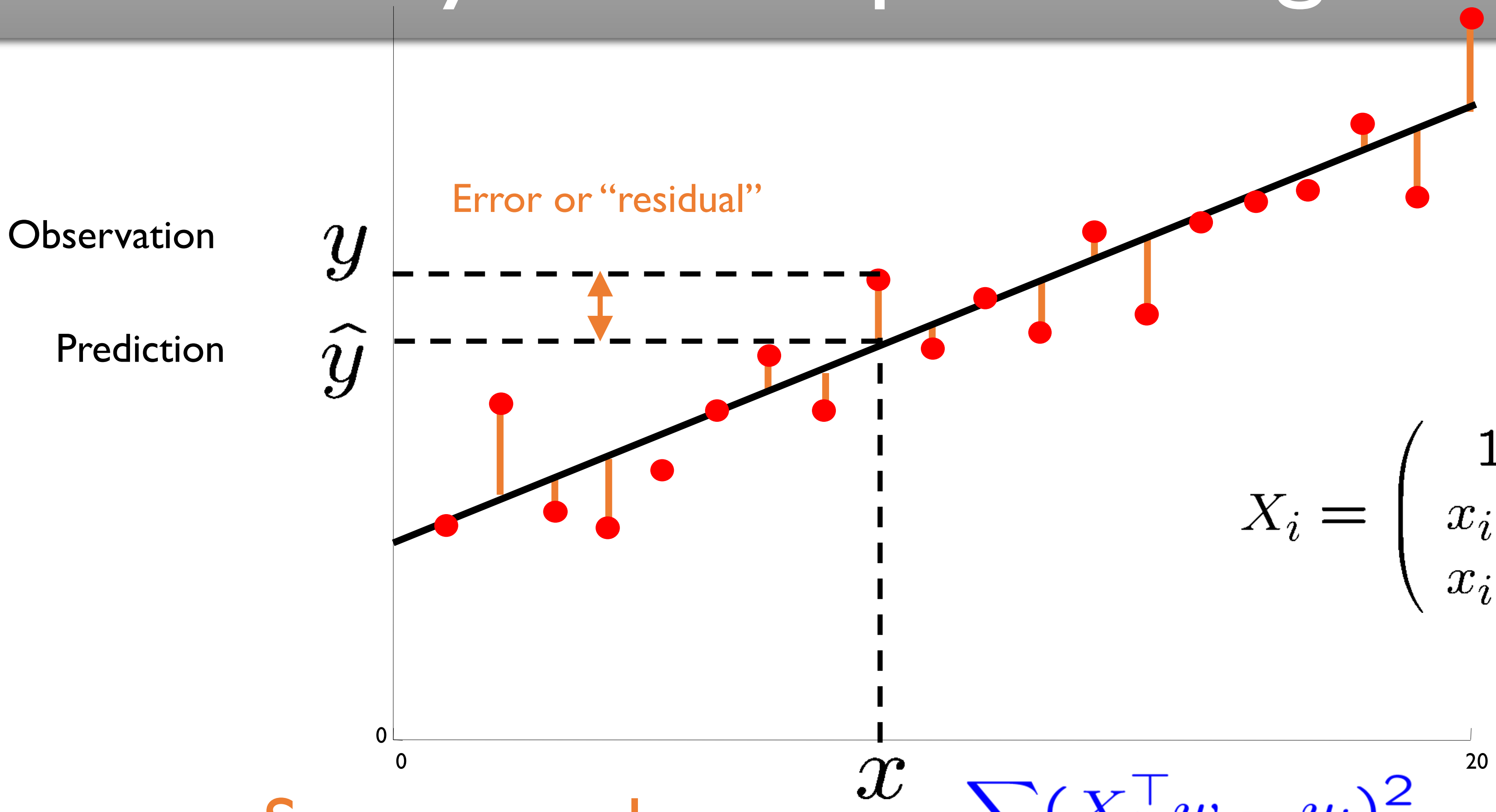
Prediction

$\widehat{y}_i = w_0 + w_1 x_i$

Pre $= \begin{pmatrix} 1 & x_{i,1} & x_{i,2} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$

$\widehat{y}_i = X_i^\top w$

Ordinary Least-Squares Regression



$$X_i = \begin{pmatrix} 1 \\ x_{i,1} \\ x_{i,2} \end{pmatrix}$$

Sum squared error $\sum_i (X_i^\top w - y_i)^2$

Example

- Predict energy requirement

Wind Speed	People inside building	Energy requirement
100	2	5
50	42	25
45	31	22
60	35	18

Solving Least-Squares Regression

Sum squared error $E = \sum_i (X_i^\top w - y_i)^2$

$$\frac{\partial E}{\partial w_j} = \sum_i \frac{\partial}{\partial w_j}(X_i^\top w - y_i)^2$$
$$= \sum_i 2X_{i,j}(X_i^\top w - y_i)$$

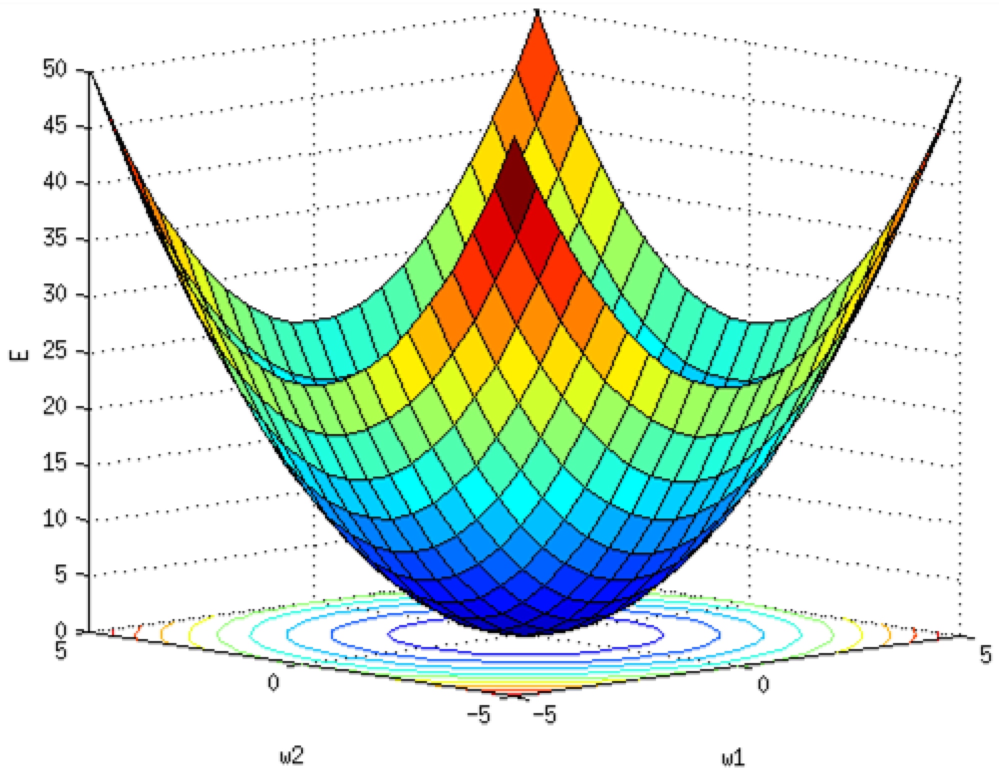
$$\frac{\partial E}{\partial w_j} = 0 \iff \left(\sum_i X_{i,j} X_i^\top\right) w = \sum_i y_i X_{i,j}$$ Linear equation

$$\frac{\partial E}{\partial w} = 0 \iff \underbrace{\left(\sum_i X_i X_i^\top\right)}_{A} w = \underbrace{\sum_i y_i X_i}_{b}$$ Linear system

$$Aw = b$$

Matrix View

- Sum of squared error:
  $E = (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w})$
  - Why is this the same as what we saw?
  - What are the dimensions?
- What's the solution for this?
- Can we solve it by hand?
- Handling multiple outputs
  - Can be trivially extended:
    $\mathbf{y} = X\mathbf{w}$ to $Y = XW$



Linear Least Squares Regression

- With slight notation changes, we have linear least-squares solution to be:

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{X}'\mathbf{y}$$

- Assume $(\mathbf{X}'\mathbf{X})^{-1}$ exists, then

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{X}'\mathbf{y} \;\Rightarrow \mathbf{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

- Alternative Representation:

$$\mathbf{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = \mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

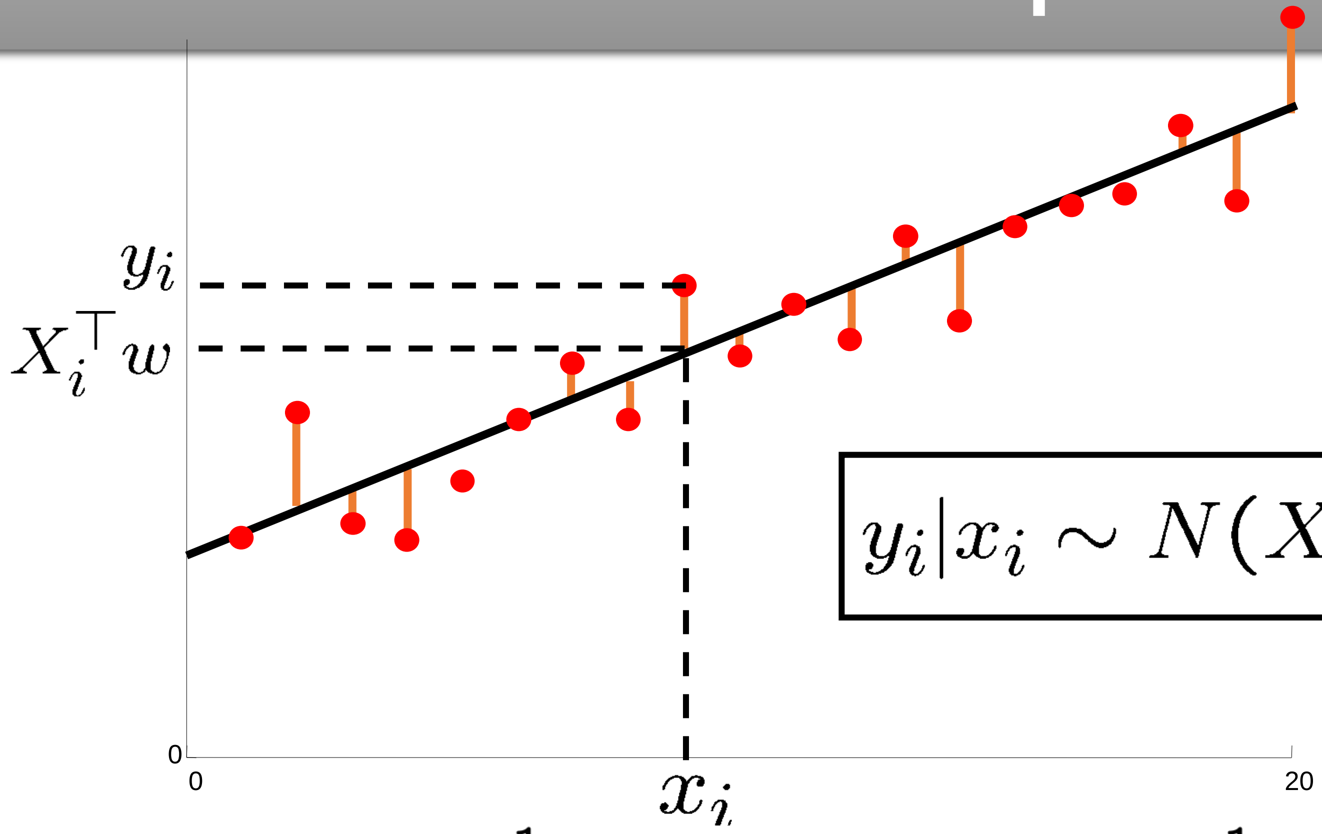
$$= \mathbf{X}'\left(\mathbf{X}(\mathbf{X}'\mathbf{X})^{-2}\mathbf{X}'\mathbf{y}\right) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\text{where } \boldsymbol{\alpha} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-2}\mathbf{X}'\mathbf{y},\ \mathbf{w} = \sum_{i=1}^{\ell}\alpha_i \mathbf{x}_i$$

Is this a good solution?

Computing inverse is not trivial; may not exist at all

Probabilistic Interpretation



$$\boxed{y_i | x_i \sim N(X_i^\top w, \sigma^2)}$$

Likelihood $L = \prod_i \text{exp} -\frac{1}{2\sigma^2}(X_i^\top w - y_i)^2 = \text{exp} -\frac{1}{2\sigma^2}\sum_i (X_i^\top w - y_i)^2$

$$\underset{w}{\text{argmax}}\, L = \underset{w}{\text{argmin}}\, E$$

Regularized Least-Squared Regression

- Complex models (lots of parameters) often prone to overfitting.
- Overfitting can be reduced by imposing a constraint on the overall magnitude of the parameters.
- Two common types of regularization in linear regression:
  - **L_2 regularization (a.k.a. ridge regression):** Find **w** which minimizes:

$$\sum_{j=1}^{N}(y_j - \sum_{i=0}^{d} w_i \cdot x_i)^2 + \lambda \sum_{i=1}^{d} w_i^2$$

    - λ is the regularization parameter: bigger λ imposes more constraint
  - **L_1 regularization (a.k.a. lasso):** Find **w** which minimizes:

$$\sum_{j=1}^{N}(y_j - \sum_{i=0}^{d} w_i \cdot x_i)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

Regularized Least-Squared Regression

- Understanding norms and regularization
  - L_p-norms:

Solving Ridge Regression

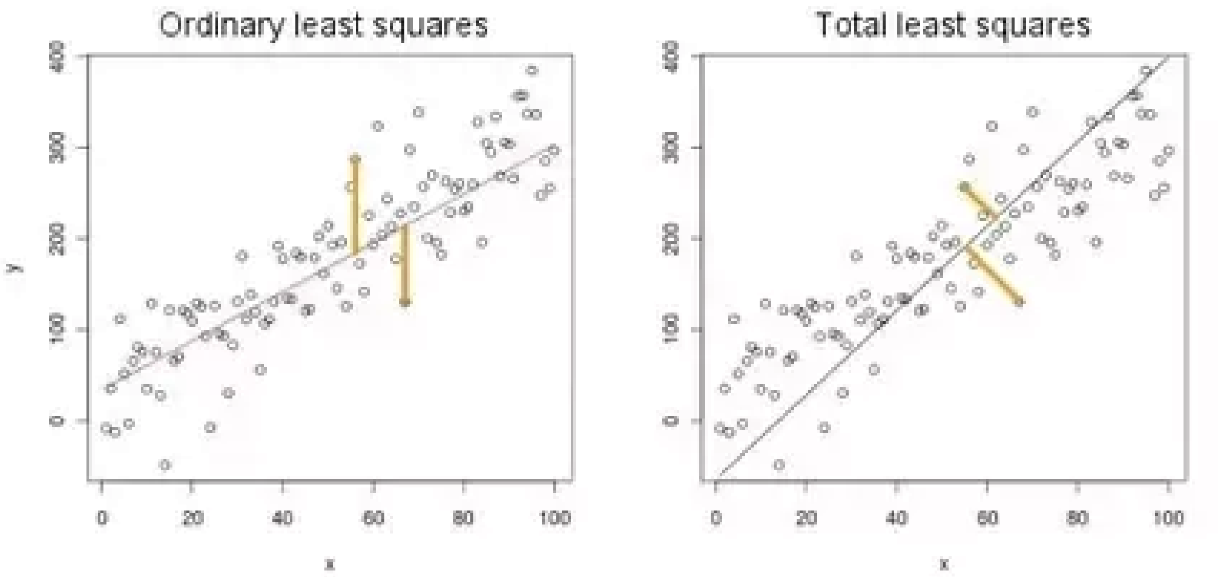
- Minimizing $\lambda\|\mathbf{w}\|^2 + \|\mathbf{y} - \mathbf{Xw}\|^2$ over **w**, we get

$$\left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_n\right)\mathbf{w} = \mathbf{X}'\mathbf{y}$$

- Solution: $\mathbf{w} = \left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}'\mathbf{y}$
- How is this different from the solution for OLS (Ordinary Least-Squares Regression)?
    - Inverse always exists for any $\lambda > 0$.

Total Least Squares and Partial Least Squares

- **Total Least Squares:** Model error in output and input
- **Partial Least Squares:**
  - Seeks to address the correlation between predictor variables in its model
  - Also called "Projection to Latent Structures" (PLS)

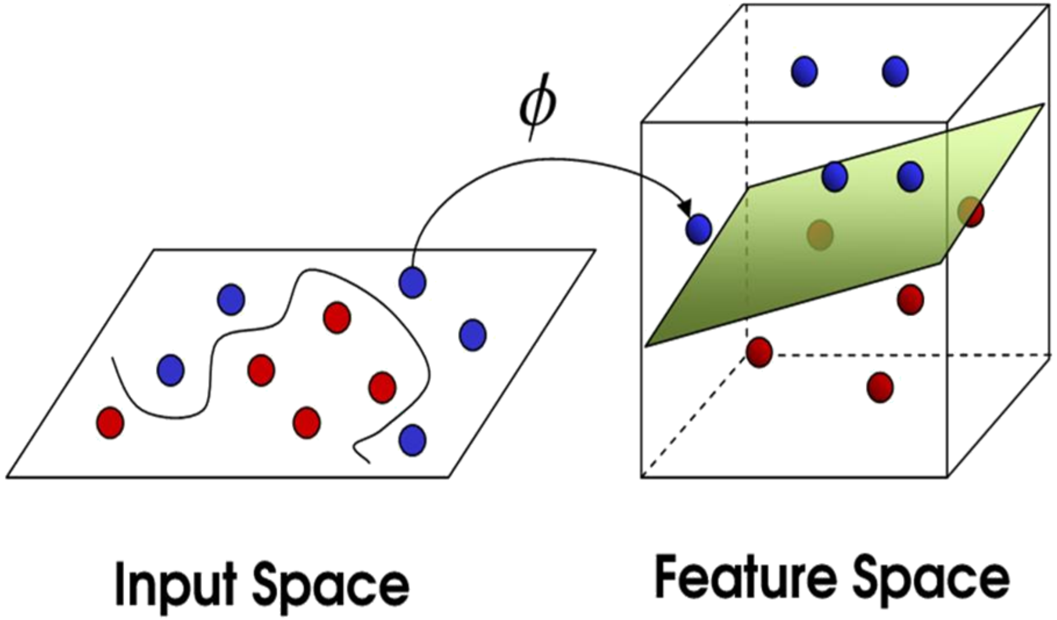


Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

Non-Linear Regression

- Recall: "kernel trick"
- **Key Idea:** Map data to higher dimensional space (feature space) and perform linear regression in embedded space



Non-Linear/Kernel Regression

- Alternative view to ridge regression solution:

$$\mathbf{w} = \left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}'\mathbf{y}$$

Solving is $0(d^3)$

$$\left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{w} = \mathbf{X}'\mathbf{y} \Rightarrow \mathbf{w} = \lambda^{-1}\left(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}\right)$$

$$\Rightarrow \mathbf{w} = \lambda^{-1}\mathbf{X}'\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \lambda^{-1}\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)$$

$$\Rightarrow \lambda\boldsymbol{\alpha} = \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right) = \left(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha}\right)$$

$$\Rightarrow \mathbf{X}\mathbf{X}'\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha} = \mathbf{y}$$

$$\Rightarrow \boldsymbol{\alpha} = \left(\mathbf{G} + \lambda\mathbf{I}_\ell\right)^{-1}\mathbf{y} \text{ where } \mathbf{G} = \mathbf{X}\mathbf{X}'$$

Solving is $0(n^3)$

Do you spot anything interesting?

Non-Linear/Kernel Regression

- To predict new point:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{i=1}^{\mathrm{d}} \alpha_i \mathbf{x}_i , \mathbf{x} \right\rangle = \mathbf{y}'(\mathbf{G} + \lambda \mathbf{I})^{-1} \mathbf{z}$$

$$\text{where} \quad \mathbf{z} = \langle \mathbf{x}_i , \mathbf{x} \rangle$$

- Need to only compute G, the Gram Matrix (or the inner products between data points)

$$\mathbf{G} = \mathbf{X}\mathbf{X}' \qquad G_{ij} = \langle \mathbf{x}_i , \mathbf{x}_j \rangle$$

Kernel Ridge Regression

- To predict new point:

$$g(\phi(\mathbf{x})) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \left\langle \sum_{i=1}^{\ell} \alpha_i \phi(\mathbf{x}_i), \phi(\mathbf{x}) \right\rangle = \mathbf{y}'(\mathbf{G} + \lambda \mathbf{I})^{-1} \mathbf{z}$$

$$\text{where} \quad \mathbf{z} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$$

- Use kernel to compute inner products

$$\mathbf{G} = \phi(\mathbf{X})\phi(\mathbf{X})' \qquad G_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = K(\mathbf{x}_i, \mathbf{x}_j)$$

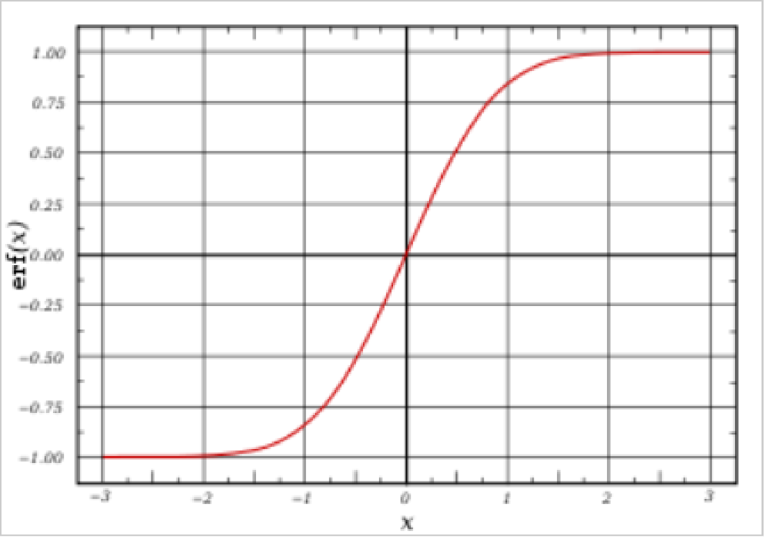
Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

Logistic Regression

- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.
- Used because having a categorical outcome variable violates the assumption of linearity in normal regression.

- Let X be the data instance, and Y be the class label: Learn P(Y|X) directly
  - Let $W = (W_1, W_2, \ldots W_n)$, $X=(X_1, X_2, \ldots, X_n)$, **W.X** is the dot product
  - Sigmoid function: $$P(Y=1|\mathbf{X}) = \frac{1}{1+e^{-\mathbf{wx}}}$$



Logistic Regression

- Generative classifier, e.g., Naïve Bayes:
  - Assume some functional form for **P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X=x)
  - This is 'generative' model
    - Indirect computation of P(Y|X) through Bayes rule
    - But, can generate a sample of the data, $P(X) = \sum_y P(y)P(X \mid y)$
- Discriminative classifier, e.g., Logistic Regression:
  - Assume some functional form for **P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the 'discriminative' model
    - Directly learn P(Y|X)
    - But cannot sample data, because P(X) is not available

Logistic Regression

- In logistic regression, we learn the conditional distribution P(y|x)
- Let $p_y(x;w)$ be our estimate of P(y|x), where w is a vector of adjustable parameters.
- Assume there are two classes, y = 0 and y = 1 and

$$p_1(\mathbf{x};\mathbf{w}) = \frac{1}{1+e^{-\mathbf{wx}}} \qquad p_0(\mathbf{x};\mathbf{w}) = 1 - \frac{1}{1+e^{-\mathbf{wx}}}$$

- This is equivalent to $\log \frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{wx}$

- That is, the log odds of class 1 is a linear function of x
- Q: How to find **W**?

Logistic Regression

- Conditional data likelihood - Probability of observed Y values in the training data, conditioned on corresponding X values.
- We choose parameters w that satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \prod_{l} P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- where
  - $\mathbf{w} = \langle w_0, w_1, \ldots, w_n \rangle$ is the vector of parameters to be estimated,
  - y^l denotes the observed value of Y in the *l* th training example, and
  - $\mathbf{x}^l$ denotes the observed value of **X** in the *l* th training example

Logistic Regression

- Equivalently, we can work with log of conditional likelihood:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_l \ln P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- Conditional data log likelihood, l(W), can be written as

$$l(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 \mid \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 \mid \mathbf{x}^l, \mathbf{w})$$

- Note here that Y can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given y^l

Logistic Regression: Training

- We need to estimate:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

$$= arg \max_{w} - \sum_{l} \ln(1 + \exp(-y_i W^T x_i))$$

- Equivalently, we can minimize negative log likelihood:

$$= arg \min_{w} \sum_{l} \ln(1 + \exp(-y_i W^T x_i))$$

- This is convex – so, unique global minimum
- No closed-form solution though. Iterative method required.

Logistic Regression: Training

- Use gradient ascent for the maximization problem
- The i th component of the vector gradient has the form

$$\frac{\partial}{\partial w_i} l(\mathbf{w}) = \sum_l x_i^l \left(y^l - \boxed{\hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w})}\right)$$

Logistic Regression prediction

- Beginning with initial weights of zero, we repeatedly update the weights in the direction of the gradient, changing the i th weight according to

$$w_i \leftarrow w_i + \eta \sum_l x_i^l (y^l - \hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w}))$$

Regularization in Logistic Regression

- Overfitting can arise especially when data has very high dimensions and is sparse.
- One approach -> modified "penalized log likelihood function," which penalizes large values of **w**, as before.

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_l \ln P(y^l \mid \mathbf{x}^l, \mathbf{w}) - \frac{\lambda}{2} \| \mathbf{w} \|^2$$

- Derivative then becomes:

$$\frac{\partial}{\partial w_i} l(\mathbf{w}) = \sum_l x_i^l (y^l - \hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w})) - \lambda w_i$$

Logistic Regression

- In general, NB and LR make different assumptions
  - NB: Features independent given class -> assumption on P(X|Y)
  - LR: Functional form of P(Y|X), no assumption on P(X|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - Concave (convex) -> global optimum with gradient ascent (descent)
- Extending logistic regression to multiple classes
  - Use softmax for each class k!

$$p(y = k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^K \exp(\theta_i^\top x)}$$

Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

k-NN Regression

- Calculate the mean value of the *k* nearest training examples rather than calculate their most common value

$$f : \Re^d \rightarrow \Re \qquad \hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Regression Methods

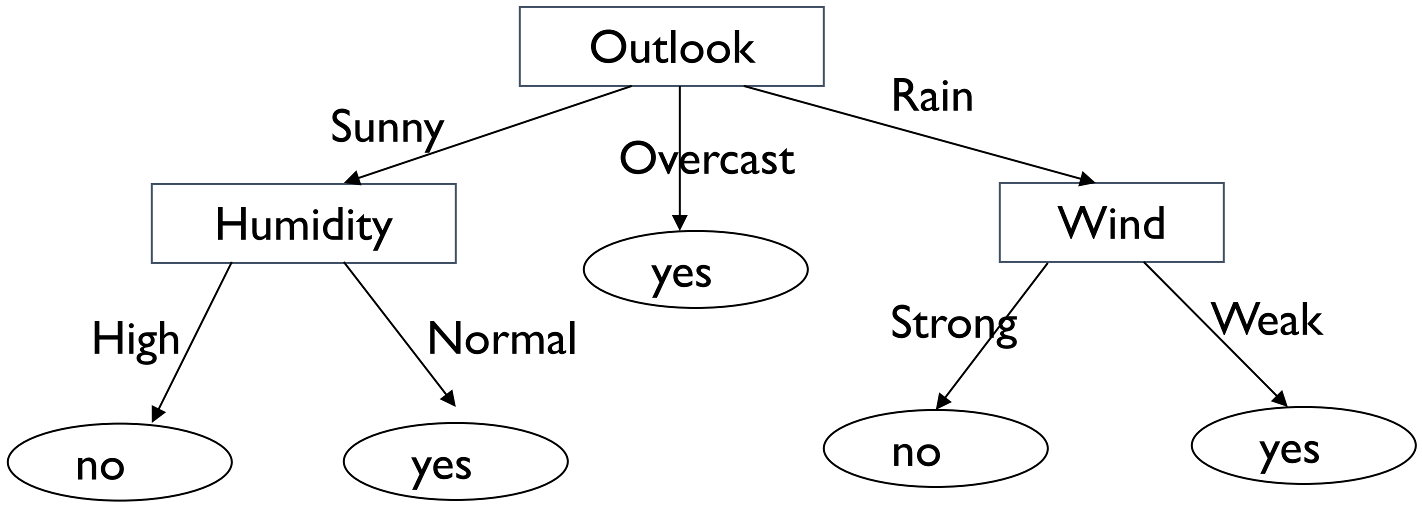
- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

Recall: Example

PlayTennis: training examples

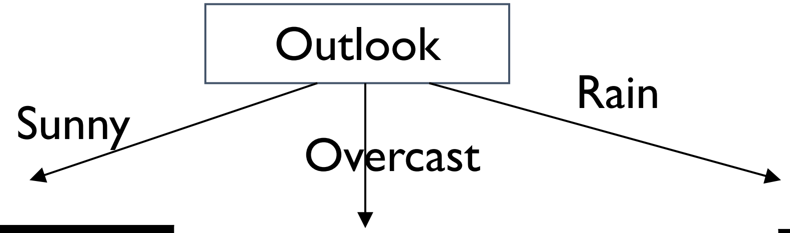
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Recall: Example



Recall: Decision Trees

- Choose « best » attribute
- Split the learning sample
- Proceed recursively until each object is correctly classified



Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Hot	Normal	Weak	Yes
D11	Sunny	Cool	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Play
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Play
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Strong	Yes
D14	Rain	Mild	High	Strong	No

Recall: Decision Trees

- The "best" split is the split that maximizes the expected reduction of impurity

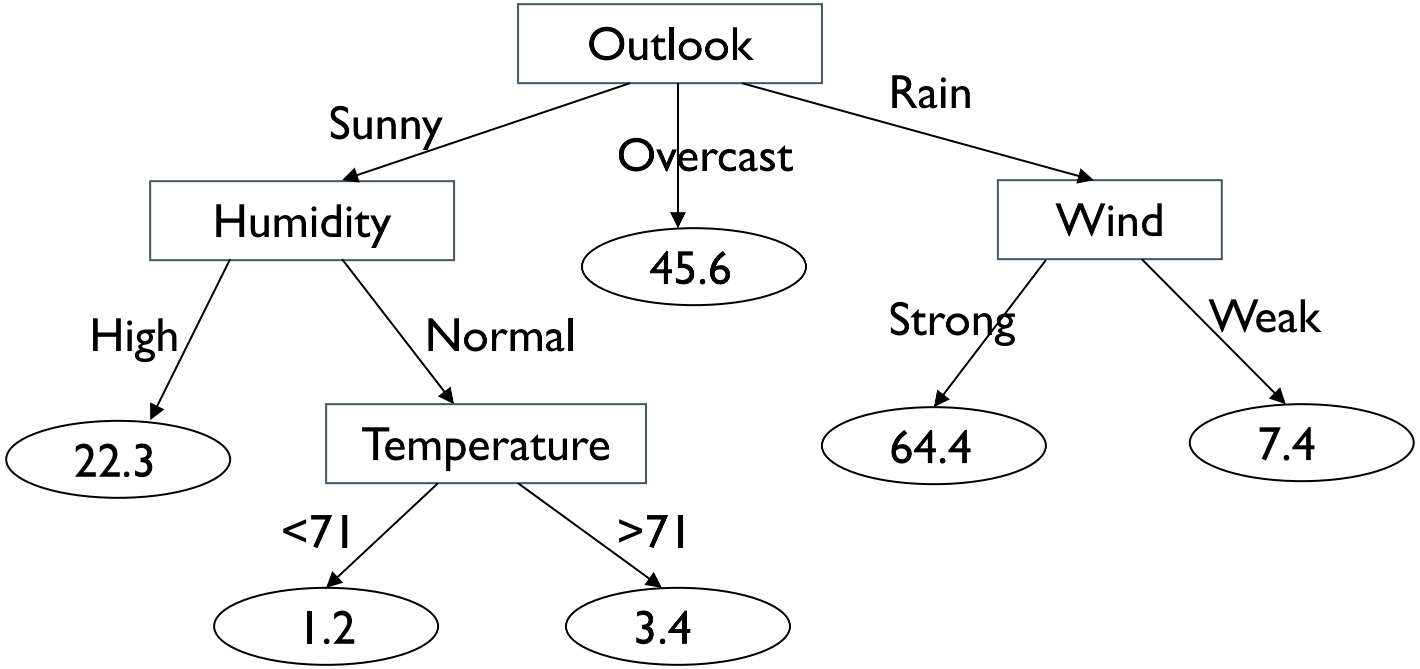
$$\Delta I(LS, A) = I(LS) - \sum_a \frac{|LS_a|}{|LS|} I(LS_a)$$

where LS_a is the subset of records from LS (dataset) such that $A=a$

- Example of impurity measure:
  - *Shannon entropy:* $I(LS)=-\sum_j p_j \log p_j$
  - If two classes, $p_1=1-p_2$
  - The reduction of entropy is called the information gain

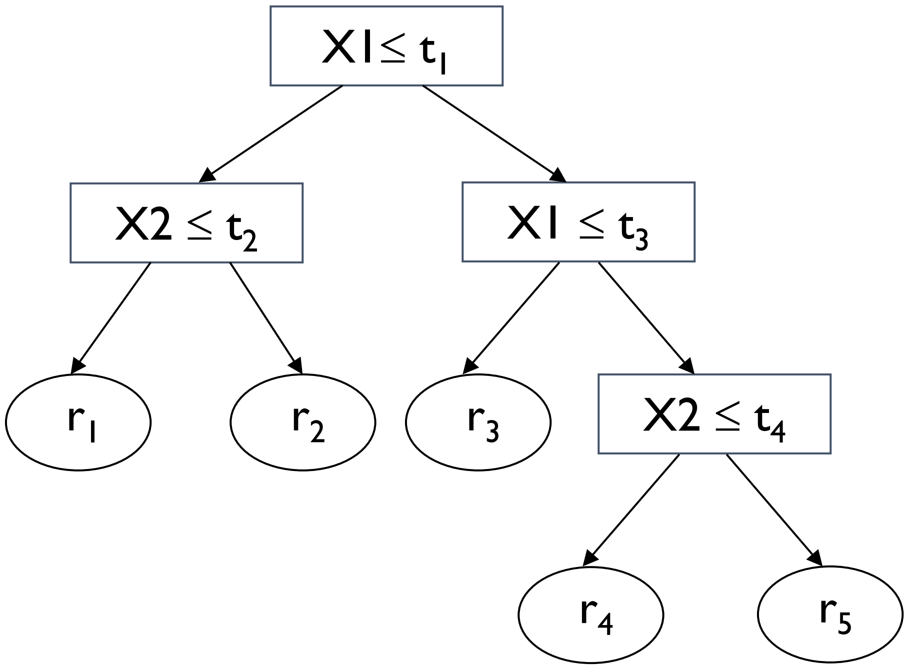
Regression Trees

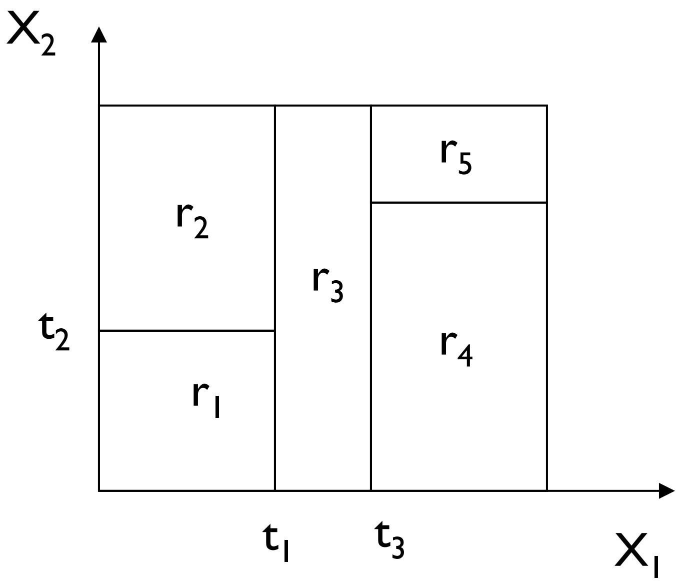
- Tree for regression: exactly the same model but with a number in each leaf instead of a class



Regression Trees

- A regression tree is a piecewise constant function of the input attributes





Growing Regression Trees

- To minimize the square error on the learning sample, the prediction at a leaf is the average output of the learning cases reaching that leaf
- Impurity of a sample is defined by the variance of the output in that sample:

$$I(LS)=var_{y|LS}\{y\}=E_{y|LS}\{(y-E_{y|LS}\{y\})^2\}$$

- The best split is the one that reduces the most variance:

$$\Delta I(LS,A)=\mathrm{var}_{y|LS}\{y\}-\sum_a \frac{|LS_a|}{|LS|}\mathrm{var}_{y|LS_a}\{y\}$$

Regression Tree Pruning

- Exactly the same algorithms apply: pre-pruning and post-pruning.
- In post-pruning, the tree that minimizes the squared error on *VS* is selected.

- In practice, pruning is more important in regression because full trees are much more complex
  - Each data instance can have a different output value and hence the full tree has as many leaves as there are training instances

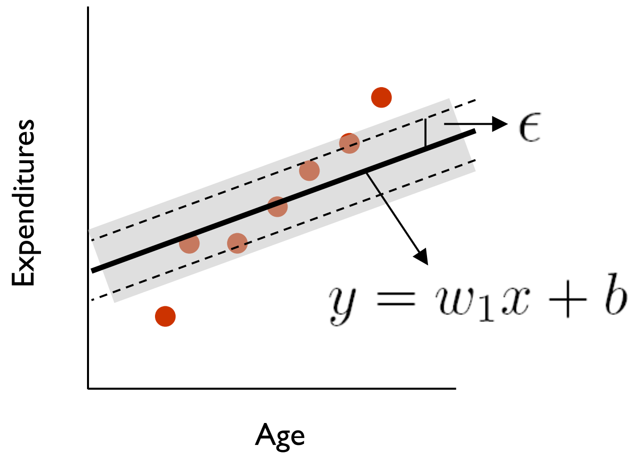
Regression Methods

- Linear Least-Squares Regression
  - Partial Least-Squares
  - Total Least-Squares
  - Ridge Regression, LASSO
- Kernel Regression
- Logistic Regression
- k-NN Regression
- Regression Trees
- Support Vector Regression

Support Vector Regression

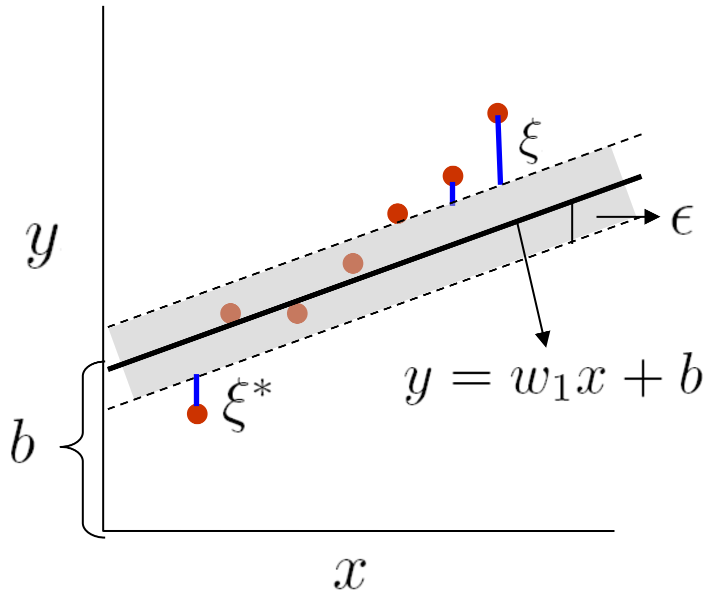
- Given training data $\{x_i, y_i\}_{i=1}^{n}$
- Find: w_1 and b ,

  such that $y = w_1 x + b$ optimally describes the data:



Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

Support Vector Regression



$|w_1|$ vs. $\sum_i (\xi_i + \xi_i^*)$

Complexity | Sum of errors

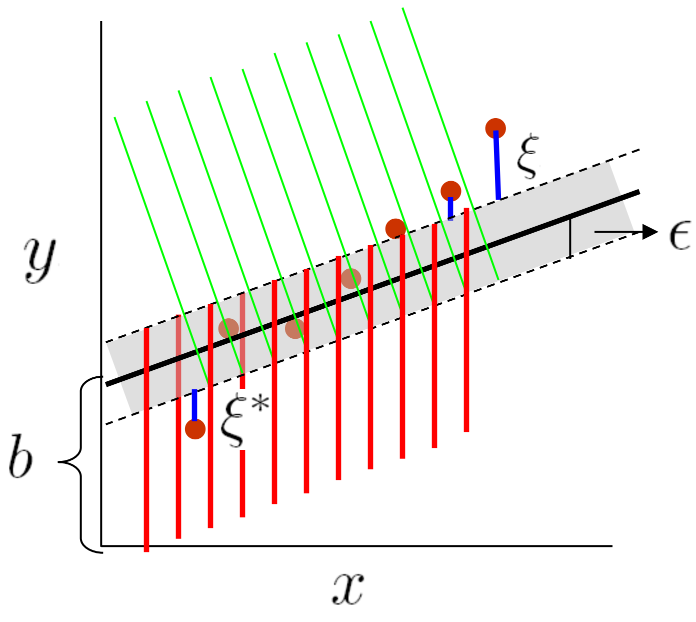
$$\min_{w_1, b, \xi_i, \xi_i^*} \frac{1}{2} w_1^2 + C \sum_i (\xi_i + \xi_i^*)$$

Case I: $w_1 \downarrow \longrightarrow$ "tube"$\uparrow \longrightarrow$ complexity $\downarrow \longrightarrow \sum_i (\xi_i + \xi_i^*) \uparrow$

Case II: $w_1 \uparrow \longrightarrow$ "tube"$\downarrow \longrightarrow$ complexity $\uparrow \longrightarrow \sum_i (\xi_i + \xi_i^*) \downarrow$

Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

Support Vector Regression



$$y = w_1 x + b$$

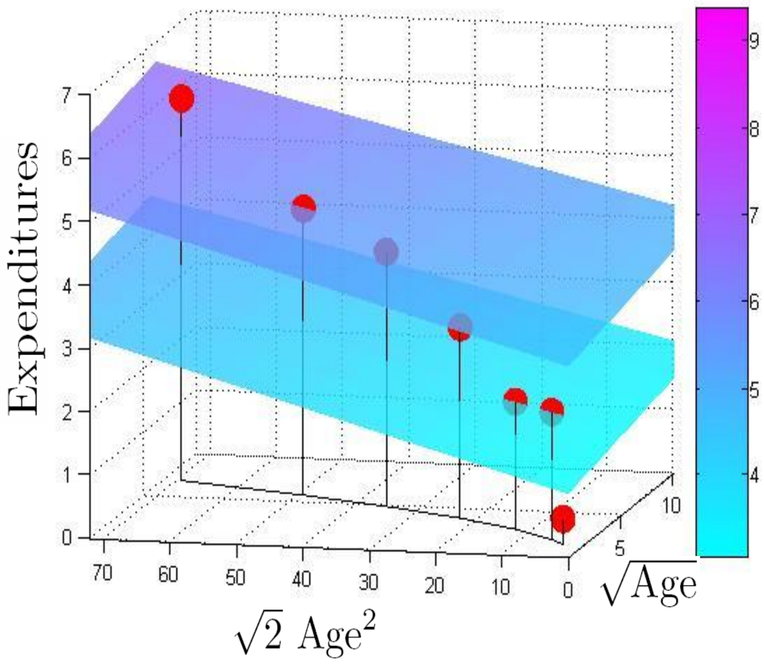
$$\min_{w_1, b, \xi_i, \xi_i^*} \quad \frac{1}{2} w_1^2 + C \sum_i (\xi_i + \xi_i^*)$$

Subject to:

$$y_i - (w_1 x_{i1}) - b \leq \epsilon + \xi_i$$
$$(w_1 x_{i1}) + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0 \qquad i = 1, 2, \ldots, n$$

Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

Non-linear (Kernel) SVR



$$y = \mathbf{w}'\Phi(x) + b$$

$$\min_{w_1, b, \xi_i, \xi_i^*} \quad \frac{w_1^2 + w_2^2}{2} + C\sum_i(\xi_i + \xi_i^*)$$

Subject to:

$$y_i - (\mathbf{w}'\phi(x_{i1})) - b \leq \epsilon + \xi_i$$
$$(\mathbf{w}'\phi(x_{i1})) + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0 \qquad i = 1, 2, \ldots, n$$

Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

SVR: Derivation

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i (\xi_i + \xi_i^*)$$

Subject to:

$$y_i - (\mathbf{w}'\phi(\mathbf{x}_i)) - b \leq \epsilon + \xi_i$$
$$(\mathbf{w}'\phi(\mathbf{x}_i))) + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0 \qquad i = 1, 2, \ldots, n$$

$$L := \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i + \eta_i^* \xi_i^*)$$
$$- \sum_i \alpha_i (\epsilon + \xi_i - y_i + \mathbf{w}'\phi(\mathbf{x}_i) + b) - \sum_i \alpha_i^* (\epsilon + \xi_i^* + y_i - \mathbf{w}'\phi(\mathbf{x}_i)) - b)$$

min with respect to $\mathbf{w},b,\xi_i,\xi_i^*$

max with respect to $\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$

Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

SVR: Derivation

$$L := \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C\sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i + \eta_i^* \xi_i^*)$$
$$- \sum_i \alpha_i (\epsilon + \xi_i - y_i + \mathbf{w}'\phi(\mathbf{x}_i) + b) - \sum_i \alpha_i^* (\epsilon + \xi_i^* + y_i - \mathbf{w}'\phi(\mathbf{x}_i)) - b)$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i (\alpha_i - \alpha_i^*)\phi(\mathbf{x}_i) = 0$$

...

$$f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}) + b$$

$$f(\mathbf{x}) = \sum_i (\alpha_i - \alpha_i^*)(\phi(\mathbf{x}_i)'\phi(\mathbf{x})) + b$$

$$\boxed{f(\mathbf{x}) = \sum_i (\alpha_i - \alpha_i^*)k(\mathbf{x}_i, \mathbf{x}) + b}$$

Slide Credit: Georgi Nalbantov, Erasmus Research Institute of Management

SVR: Summary

- Strengths of SVR:
  - No local minima
  - Scales relatively well to high dimensional data
  - Trade-off between classifier complexity and error can be controlled explicitly via *C* and *epsilon*
  - Overfitting is avoided (for any fixed *C* and *epsilon*)
  - The “curse of dimensionality” is avoided through kernel functions
- Weaknesses of SVR:
  - What is the best trade-off parameter *C* and best *epsilon*?
  - What is a *good* transformation of the original space?

Other Regression Methods

- Bayesian Regression
- Generalized Regression Neural Network

Readings

- “Introduction to Machine Learning” by Ethem Alpaydin
  - Sections 4.6, 5.8
  - Chapter 10
  - Chapter 13