Regression and the Bias-Variance Decomposition

William Cohen

10-601 April 2008

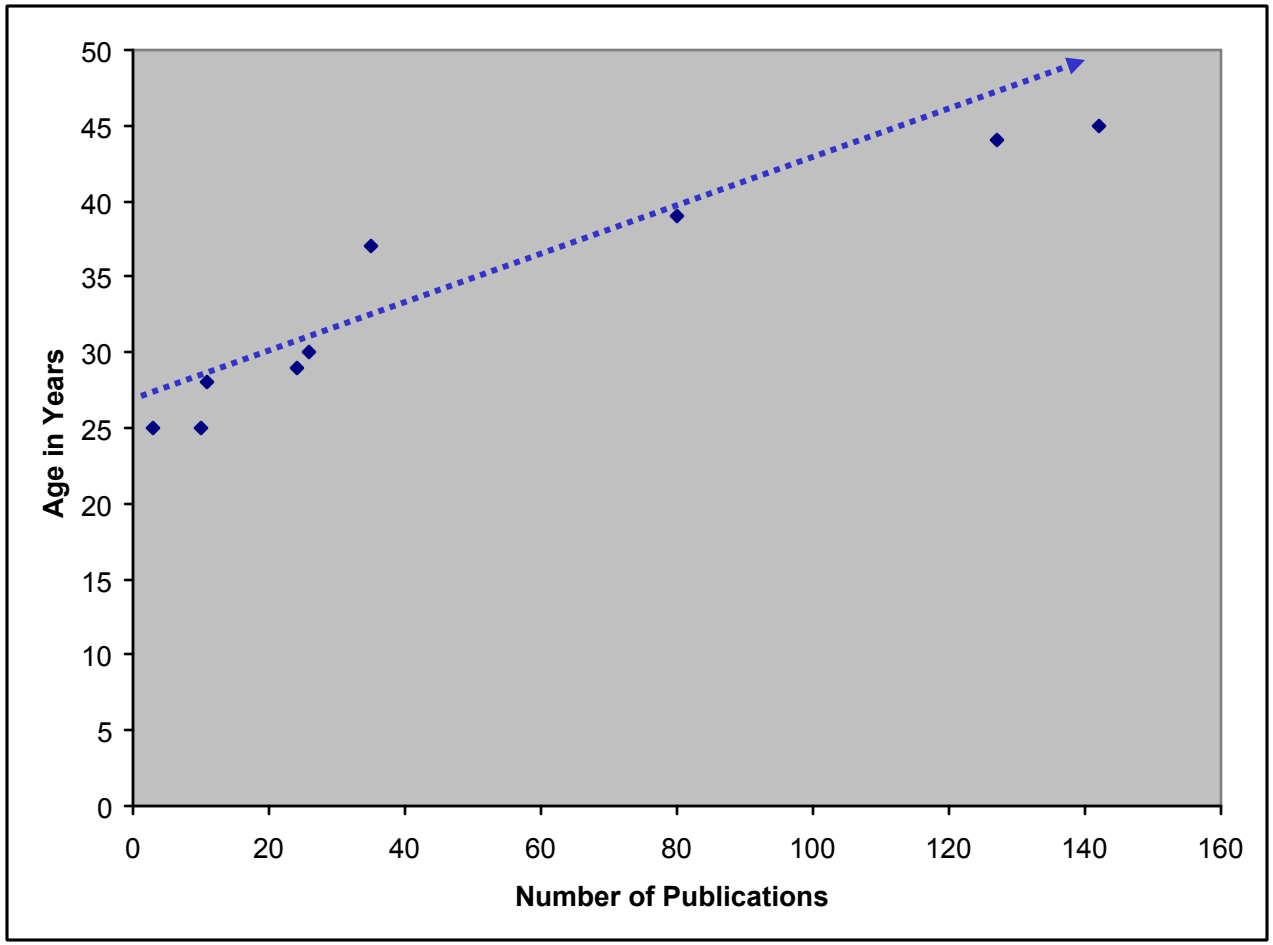
Readings: Bishop 3.1,3.2

Regression

- Technically: learning a function $f(\boldsymbol{x})=y$ where y is **real-valued**, rather than **discrete**.
  - Replace livesInSquirrelHill(x_1,x_2,…,x_n) with averageCommuteDistanceInMiles(x_1,x_2,…,x_n)
  - Replace userLikesMovie(**u**,**m**) with usersRatingForMovie(**u**,**m**)
  - …

Example: univariate linear regression

- Example: predict age from number of publications



Linear regression

- Model: $y_i = ax_i + b + \varepsilon_i$ where $\varepsilon_i \sim N(0,\sigma)$
- Training Data: $(x_1,y_1),\ldots.(x_n,y_n)$
- Goal: estimate a,b with $\mathbf{w}=(\hat{a},\hat{b})$

$$\begin{aligned}
\mathbf{w} &= \arg\max \Pr(\mathbf{w} \mid D) \\
&= \arg\max \Pr(D \mid \mathbf{w}) \Pr(\mathbf{w}) \\
&\approx \arg\max \log \Pr(D \mid \mathbf{w}) \\
&\approx \arg\max \sum_i \log \Pr(y_i \mid x_i, \mathbf{w}) \\
&= \arg\min \sum_i [\hat{\varepsilon}_i(\mathbf{w})]^2
\end{aligned}$$

assume MLE

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

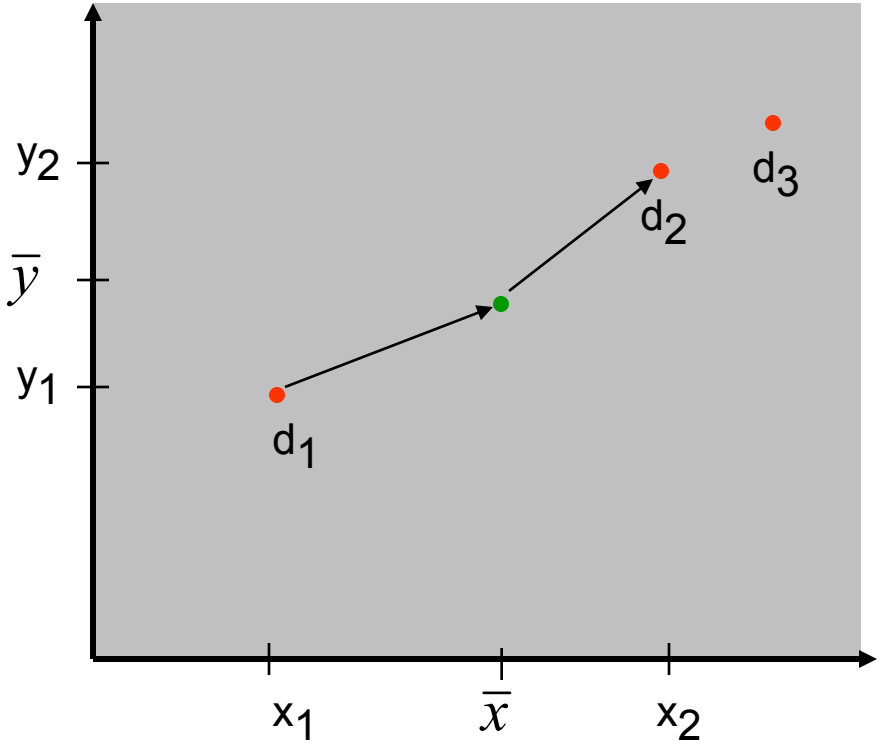
$$\hat{\varepsilon}_i(\mathbf{w}) \equiv y_i - (\hat{a}x_i + \hat{b})$$

Linear regression

- Model: $y_i = ax_i + b + \varepsilon_i$ where $\varepsilon_i \sim N(0,\sigma)$
- Training Data: $(x_1,y_1),\ldots.(x_n,y_n)$
- Goal: estimate a,b with $\mathbf{w}=(\hat{a},\hat{b})$ $\mathbf{w} = \arg\min \sum_i \hat{\varepsilon}_i^{\,2}$
- Ways to estimate parameters
  - Find derivative wrt parameters a,b
  - Set to zero and solve
    - Or use gradient ascent to solve
    - Or ….

Linear regression

How to estimate the slope?



$$slope = \frac{\Delta y}{\Delta x}$$

$$\approx \frac{(y_1 - \bar{y})}{(x_1 - \bar{x})} \approx \frac{(y_2 - \bar{y})}{(x_2 - \bar{x})} \approx \ldots$$

$$\approx \frac{\frac{1}{n}\sum_i (y_i - \bar{y})}{\frac{1}{n}\sum_i (x_i - \bar{x})} \approx \sum_i \frac{(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

*n*cov(X,Y)*

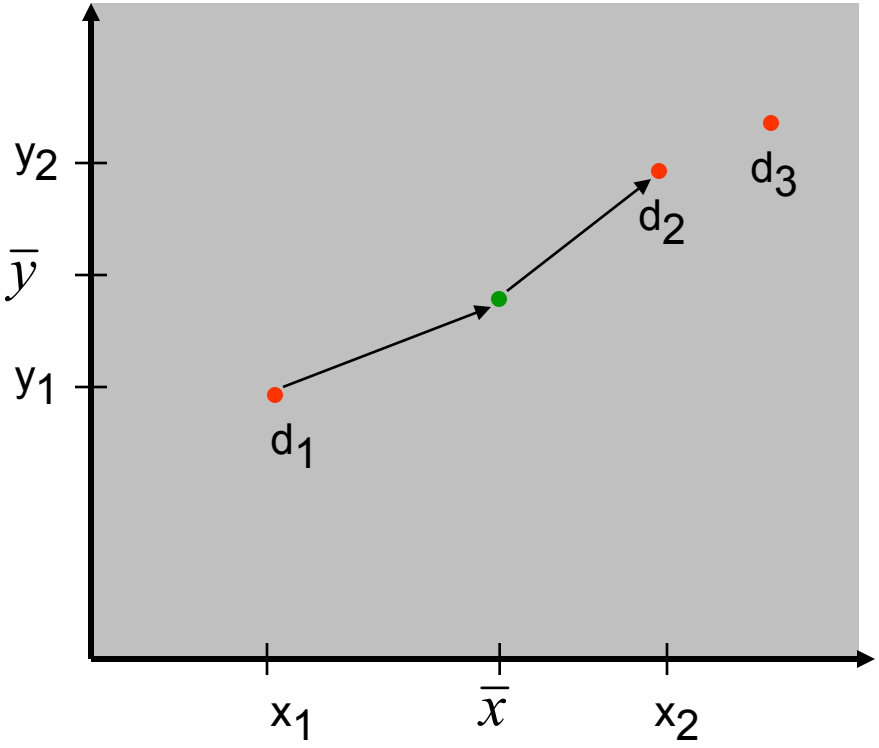
*n*var(X)*

Linear regression

How to estimate the intercept?

$$y = \hat{a}x + \hat{b}$$

$$\hat{b} = \bar{y} - \hat{a}\bar{x}$$



Bias/Variance Decomposition of Error

Bias – Variance decomposition of error

- Return to the simple regression problem f:X→Y

$y = f(x) + \varepsilon$ ← noise $N(0,\sigma)$

deterministic

What is the expected error for a learned *h*?

Bias – Variance decomposition of error

$$E_D[\int\int \left(f(x)+\varepsilon-h_D(x)\right)^2 \Pr(\varepsilon)\Pr(x)d\varepsilon dx]$$

dataset

true fct

learned from D

Experiment (the error of which I'd like to predict):

1. Draw size *n* sample D=$(x_1,y_1),\ldots.(x_n,y_n)$
2. Train linear function h_D *using* D
3. Draw a test example $(x,f(x)+\varepsilon)$
4. Measure squared error of h_D on that example

Bias – Variance decomposition of error (2)

$$E_{D,\varepsilon}\left\{\left(f(x)+\varepsilon-h_D(x)\right)^2\right\}$$

dataset

true fct

learned from D

Fix *x*, then do this experiment:

1. Draw size *n* sample D=$(x_1,y_1),\ldots.(x_n,y_n)$
2. Train linear function h_D *using* D
3. Draw the test example $(x,f(x)+\varepsilon)$
4. Measure squared error of h_D on that example

Bias – Variance decomposition of error

$$E_{D,\varepsilon}\left\{\left(\overbrace{f(x)}^{f}+\varepsilon-\overbrace{h_D(x)}^{\hat{y}}\right)^2\right\}$$

t

f

ŷ ← really $\hat{y}_D$

$$E\left\{(t-\hat{y})^2\right\}$$

$$= E\left\{\left([t-f]+[f-\hat{y}]\right)^2\right\}$$ why not?

$$= E\left\{[t-f]^2+[f-\hat{y}]^2+2[t-f][f-\hat{y}]\right\}$$

$$= E\left\{[t-f]^2+[f-\hat{y}]^2+2[tf-t\hat{y}-f^2+f\hat{y}]\right\}$$

Bias – Variance decomposition of error

$$E_{D,\varepsilon}\left\{(t-\hat{y})^2\right\}$$

$$= E\left\{\left([t-f]+[f-\hat{y}]\right)^2\right\}$$

$$= E\left\{[t-f]^2+[f-\hat{y}]^2+2[t-f][f-\hat{y}]\right\}$$

$$= E\left\{[t-f]^2+[f-\hat{y}]^2+2[tf-t\hat{y}-f^2+f\hat{y}]\right\}$$

$$= E[\varepsilon^2]+E[(f-\hat{y})^2]+2\Big(\underbrace{E[tf]}_{(f+\varepsilon)f}-\underbrace{E[t\hat{y}]}_{(f+\varepsilon)\hat{y}}-E[f^2]+E[f\hat{y}]\Big)$$

Intrinsic noise

Depends on how well learner approximates *f*

Bias – Variance decomposition of error

$$E\left\{(f-\hat{y})^2\right\}$$
$$= E\left\{\left([f-h]+[h-\hat{y}]\right)^2\right\}$$
$$= E\left\{[f-h]^2+[h-\hat{y}]^2+2[f-h][h-\hat{y}]\right\}$$
$$= E\left\{[f-h]^2+[h-\hat{y}]^2+2[fh-f\hat{y}-h^2+h\hat{y}]\right\}$$
$$= E[(f-h)^2]+E[(h-\hat{y})^2]+2\left(E[fh]-E[f\hat{y}]-E[h^2]+E[h\hat{y}]\right)$$

$$h \equiv E_D\{h_D(x)\}$$
$$\hat{y} = \hat{y}_D \equiv h_D(x)$$

Squared difference between best possible prediction for x, f(x), and our "long-term" expectation for what the learner will do if we averaged over many datasets D, $E_D[h_D(x)]$

BIAS^2

Squared difference btwn our long-term expectation for the learners performance, $E_D[h_D(x)]$, and what we expect in a representative run on a dataset D (hat y)

VARIANCE

Bias-variance decomposition

$$E_D\left[\int_y\int_x (h(x)-f(x))^2 p(y|x)p(x)dydx\right]$$

Make the long-term average better approximate the true function f(x)

$$bias^2 = \int (E_D[h(x)]-f(x))^2 p(x)dx$$

$$variance = \int E_D[(h(x)-E_D[h(x)])^2]p(x)dx$$

Make the learner less sensitive to variations in the data

How can you reduce **bias** of a learner?

How can you reduce **variance** of a learner?

A generalization of bias-variance decomposition to other loss functions

- "Arbitrary" real-valued loss *L(t,y)*
  But L(y,y')=L(y',y), L(y,y)=0,
  and L(y,y')!=0 if y!=y'
- Define "optimal prediction":
  $y^* = argmin_{y'} L(t,y')$
- Define "main prediction of learner"
  $y_m=y_{m,D} = argmin_{y'} E_D\{L(t,y'$ $m=n=|D|$
- Define "bias of learner":
  $B(x)=L(y^*,y_m)$
- Define "variance of learner"
  $V(x)=E_D[L(y_m,y)]$
- Define "noise for x":
  $N(x) = E_t[L(t,y^*)]$

Claim:

$E_{D,t}[L(t,y)] = c_1 N(x) + B(x) + c_2 V(x)$

where

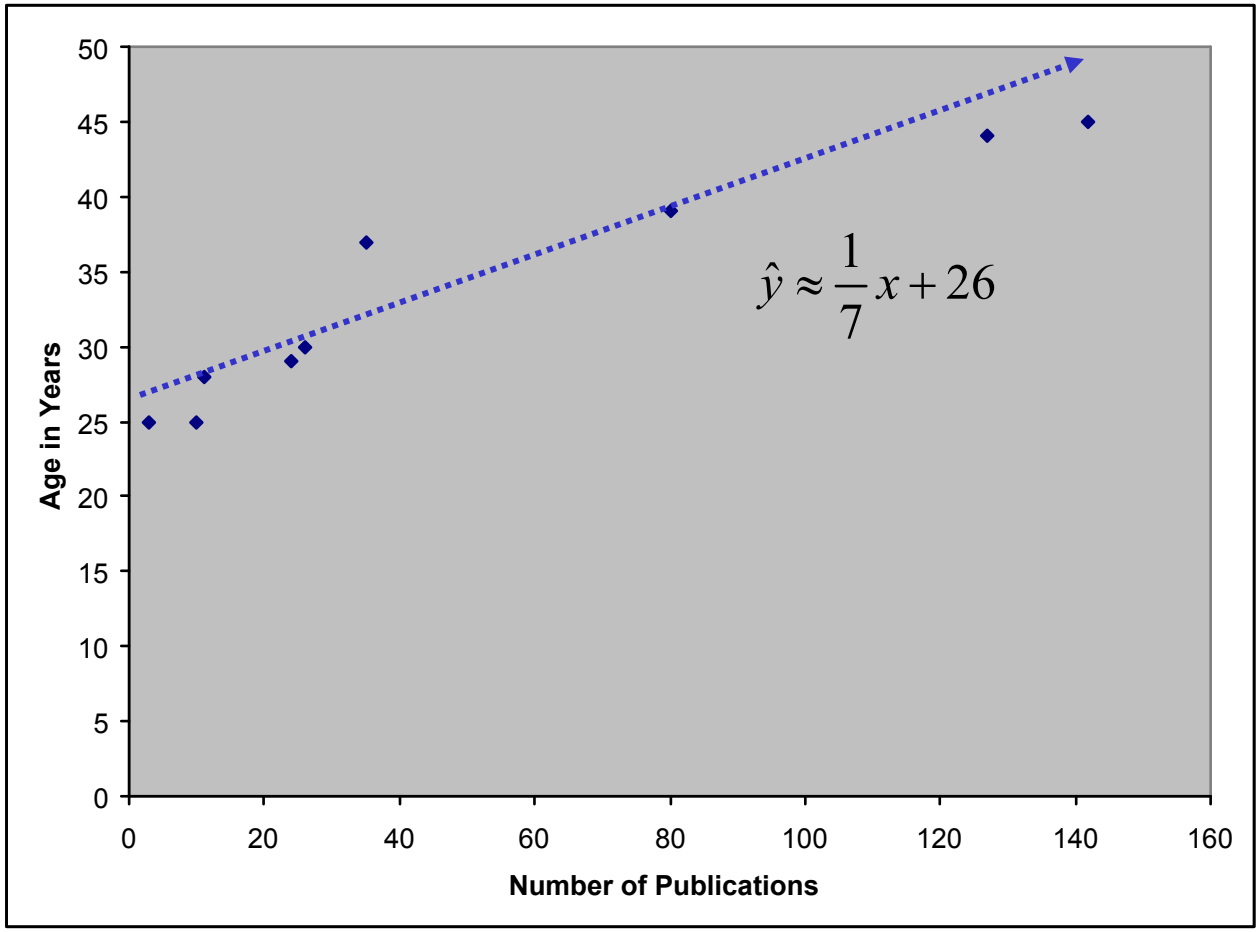
$c_1 = Pr_D[y=y^*] - 1$

$c_2 = 1$ *if* $y_m=y^*$, *-1 else*

Other regression methods

Example: univariate linear regression

- Example: predict age from number of publications



Paul Erdős

Hungarian mathematician, 1913-1996

x ~ 1500

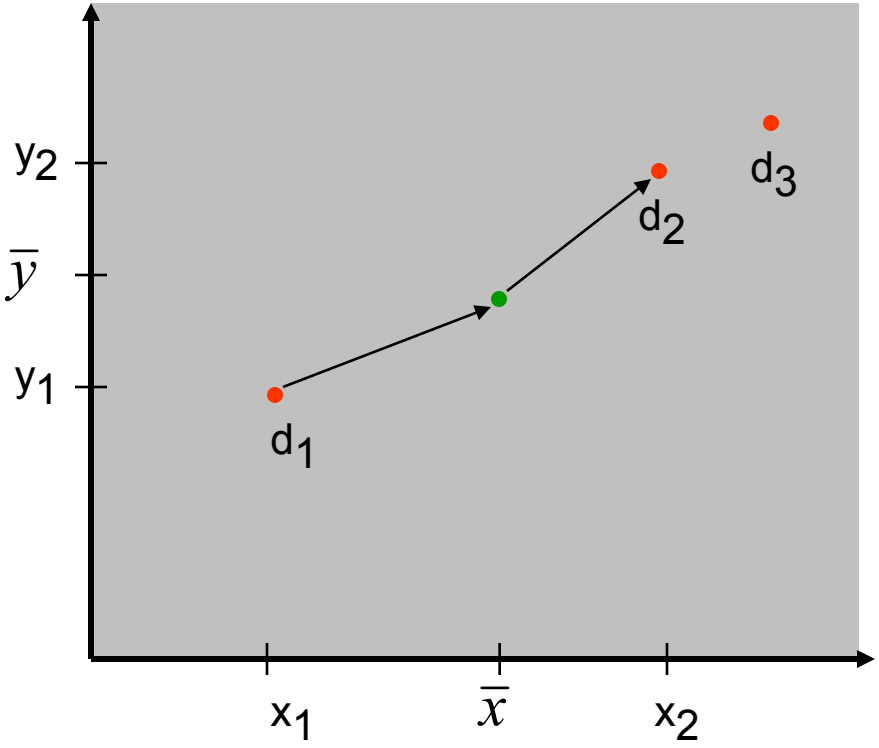
age about 240

Linear regression

Summary:

$$\hat{a} = \sum_i \frac{(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

$$\hat{b} = \bar{y} - \hat{a}\bar{x}$$



To simplify:

- assume *zero-centered data,* as we did for PCA
- let **x=**(x_1,…,x_n) and **y =**(y_1,…,y_n)
- then…

$$\hat{a} = \mathbf{x}^T\mathbf{y}(\mathbf{x}^T\mathbf{x})^{-1}$$

$$\hat{b} = 0$$

Onward: multivariate linear regression

Univariate

$$\mathbf{x} = \langle x_1, \ldots, x_n \rangle$$

$$\mathbf{y} = \langle y_1, \ldots, y_n \rangle$$

$$\hat{w} = \mathbf{x}^T \mathbf{y} (\mathbf{x}^T \mathbf{x})^{-1}$$

col is feature

Multivariate

$$X = \begin{bmatrix} x_1^1, \ldots, x_1^k \\ \ldots \\ x_n^1, \ldots, x_n^k \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix}$$

row is example

$$\hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k$$

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

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2$$

$$\hat{\varepsilon}_i(\mathbf{w}) \equiv y_i - \mathbf{w}^T \mathbf{x}_i$$

Onward: ^regularized multivariate linear regression

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2$$

$$\hat{\varepsilon}_i(\mathbf{w}) \equiv y_i - \mathbf{w}^T \mathbf{x}_i$$

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

$$X = \begin{bmatrix} x_1^1, \ldots, x_1^k \\ \ldots \\ x_n^1, \ldots, x_n^k \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix}$$

$$\hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k$$

$$\hat{\mathbf{w}} = X^T \mathbf{y} (\lambda I + X^T X)^{-1}$$

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

Onward: multivariate linear regression

Multivariate, multiple outputs

$$X = \begin{bmatrix} x_1^1, \ldots, x_1^m \\ \ldots \\ x_n^1, \ldots, x_n^m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix} \quad Y = \begin{bmatrix} y_1^1, \ldots, y_1^1 \\ \ldots \\ y_n^1, \ldots, y_n^k \end{bmatrix}$$

$$\hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k \qquad \hat{\mathbf{y}} = W\mathbf{x}$$

$$\hat{\mathbf{w}} = X^T \mathbf{y} (X^T X)^{-1} \qquad W = X^T Y (X^T X)^{-1}$$

regularized

Onward: multivariate linear regression

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2$$

$$\hat{\varepsilon}_i(\mathbf{w}) \equiv y_i - \mathbf{w}^T \mathbf{x}_i$$

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

$$X = \begin{bmatrix} x_1^1, \ldots, x_1^k \\ \ldots \\ x_n^1, \ldots, x_n^k \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix}$$

What does increasing λ do?

$$\hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k$$

$$\hat{\mathbf{w}} = X^T \mathbf{y}(\lambda I + X^T X)^{-1} \qquad \hat{\mathbf{w}} = X^T \mathbf{y}(X^T X)^{-1}$$

$$bias^2 = \int (E_D[h(x)] - f(x))^2 p(x) dx$$

$$variance = \int E_D[(h(x) - E_D[h(x)])^2] p(x) dx$$

Onward: regularized multivariate linear regression

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2$$

$$\hat{\varepsilon}_i(\mathbf{w}) \equiv y_i - \mathbf{w}^T \mathbf{x}_i$$

$$\mathbf{w} = \arg\min \sum [\hat{\varepsilon}_i(\mathbf{w})]^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

$$X = \begin{bmatrix} 1, x_1 \\ \dots \\ 1, x_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}$$

w=(w^1,$w^{2)}$ What does fixing w^2=0 do (if λ=0)?

$$\hat{y} = \hat{w}^1 x^1 + \dots + \hat{w}^k x^k$$

$$\hat{\mathbf{w}} = X^T \mathbf{y} (\lambda I + X^T X)^{-1} \qquad \hat{\mathbf{w}} = X^T \mathbf{y} (X^T X)^{-1}$$

$$bias^2 = \int (E_D[h(x)] - f(x))^2 p(x) dx$$

$$variance = \int E_D[(h(x) - E_D[h(x)])^2] p(x) dx$$

Regression trees - summary

[Quinlan's M5]

- Growing tree:
  - Split to optimize $\Delta error = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i).$
- At each ~~leaf~~ node
  - ~~Predict the majority~~ class

build a linear model, then greedily remove features

- Pruning tree:
  - Prune to reduce estimated error on training data

estimates are adjusted by *(n+k)/(n-k):* *n*=#cases, *k*=#features

- Prediction:
  - Trace path to a leaf and predict

using to a linear interpolation of every prediction made by every node on the path

If the case follows branch S_i of subtree S, let n_i be the number of training cases at S_i, $PV(S_i)$ the predicted value at S_i, and $M(S)$ the value given by the model at S. The predicted value backed up to S is

$$PV(S) = \frac{n_i \times PV(S_i) + k \times M(S)}{n_i + k}$$

Regression trees: example - 1

CHMIN $\leq$ 7 : RM0
CHMIN $>$ 7 :
| MMAX $>$ 24000 : RM1
| MMAX $\leq$ 24000 :
| | CACH $\leq$ 48 : RM2
| | CACH $>$ 48 : average 217.5

Model:	RM0	RM1	RM2
	11.5	-101.4	11.9
Cycle time			
Min mem		0.030	
Max mem	0.003		0.008
Cache size	0.902		
Min chans			
Max chans	0.518	4.686	

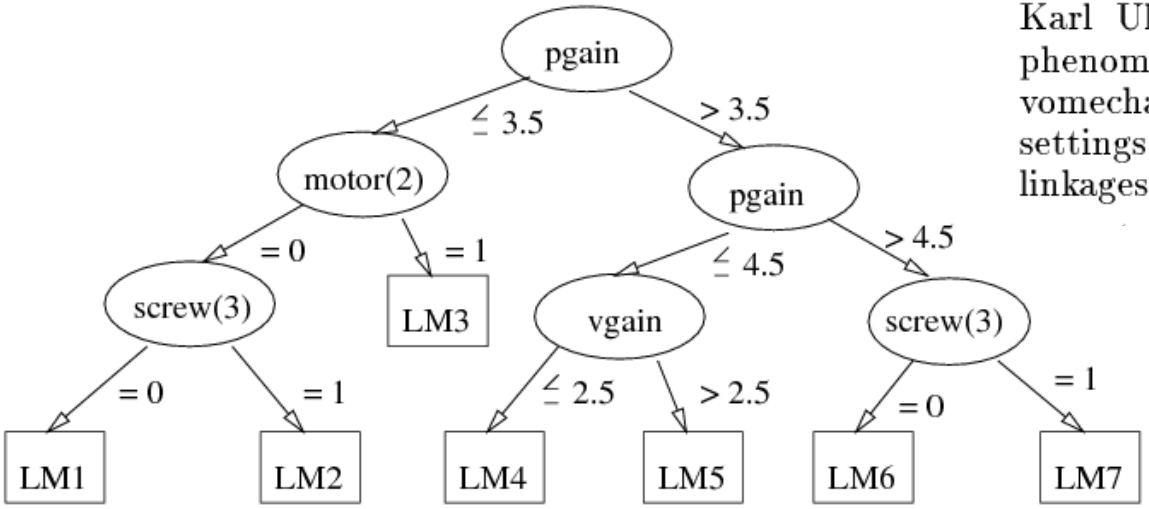
Figure 1: Model tree for CPU performance

	Original Attributes		Transformed Attributes	
	Correlation	Percentage Deviation	Correlation	Percentage Deviation
Ein–Dor (retrial)	–	–	.966	33.9%
IBP	–	35.0%	–	33.0%
M5	.921	34.9%	.956	34.0%
M5 (no smoothing)	.908	37.2%	.957	33.9%
M5 (no models)	.803	49.9%	.853	48.6%

Table 1: CPU performance data

Regression trees – example 2

- *servo:* (167 cases)
  This interesting collection of data, provided by Karl Ulrich, refers to an extremely non-linear phenomenon – predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages.



Model		LM1	LM2	LM3	LM4	LM5	LM6	LM7
constant term		−0.44	2.6	3.5	0.18	0.52	0.36	0.23
pgain								
vgain		0.82		0.42				0.06
motor = D	*vs* E, C, B, A		3.3		0.24	0.42		
motor = D, E	*vs* C, B, A	1.8			−0.16		0.15	0.22
motor = D, E, C	*vs* B, A				0.1	0.09		0.07
motor = D, E, C, B	*vs* A			0.18				
screw = D	*vs* E, C, B, A							
screw = D, E	*vs* C, B, A	0.47						
screw = D, E, C	*vs* B, A	0.63		0.28	0.34			
screw = D, E, C, B	*vs* A			0.9	0.16	0.14		

Fig. 2. Model tree and linear models for data set *servo*

What does pruning do to bias and variance?

Kernel regression

- *aka* locally weighted regression, locally linear regression, LOESS, …

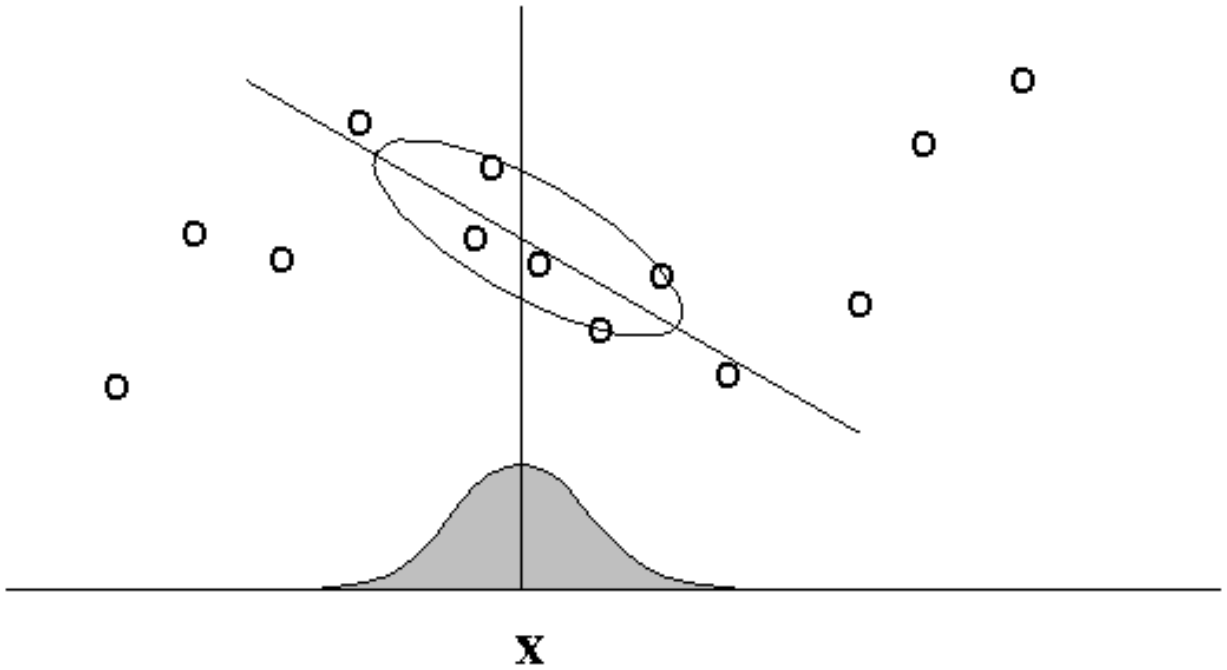


Figure 2: In locally weighted regression, points are weighted by proximity to the current **x** in question using a kernel. A regression is then computed using the weighted points.

Kernel regression

- *aka* locally weighted regression, locally linear regression, …
- Close approximation to kernel regression:
  - Pick a few values $z^1,\ldots,z^k$ up front
  - Preprocess: for each example (x,y), replace x with $\mathbf{x}=\langle K(x,z^1),\ldots,K(x,z^k)\rangle$

    where $K(x,z) = \exp(-(x-z)^2 / 2\sigma^2)$
  - Use multivariate regression on $\mathbf{x}$,y pairs

Kernel regression

- *aka* locally weighted regression, locally linear regression, LOESS, …

What does making the kernel wider do to bias and variance?

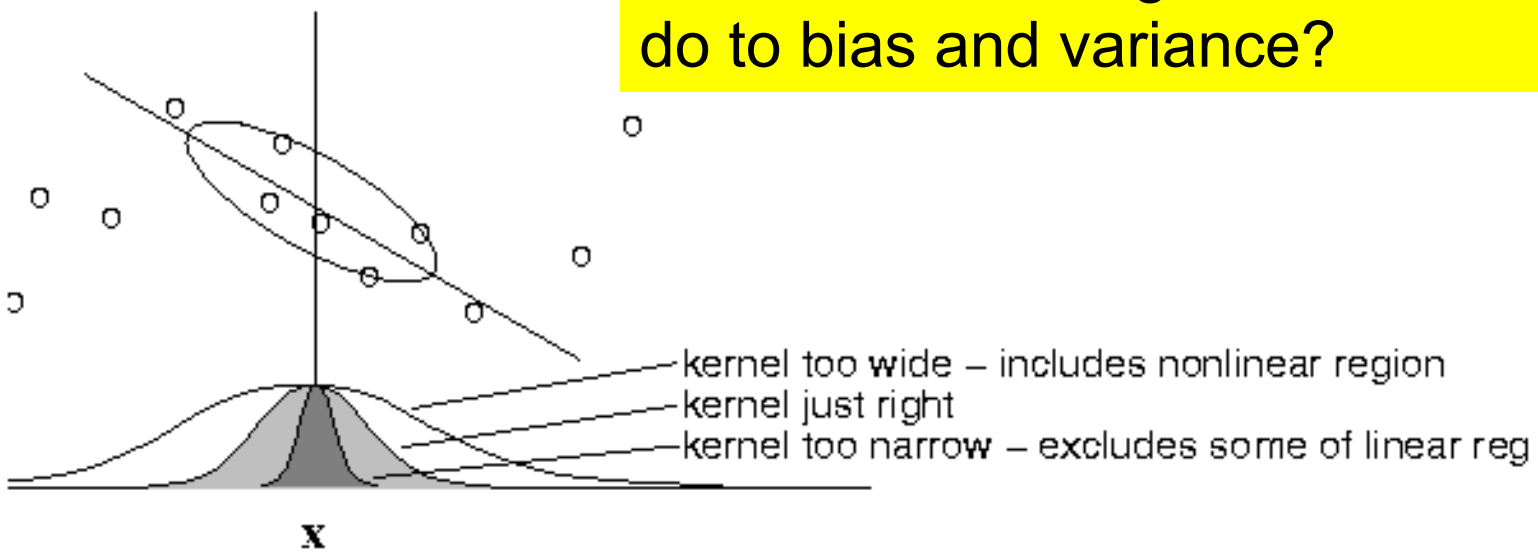


Figure 3: The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

Additional readings

- P. Domingos, A Unified Bias-Variance Decomposition and its Applications. Proceedings of the Seventeenth International Conference on Machine Learning (pp. 231-238), 2000. Stanford, CA: Morgan Kaufmann.
- J. R. Quinlan, Learning with Continuous Classes, 5th Australian Joint Conference on Artificial Intelligence, 1992.
- Y. Wang & I. Witten, Inducing model trees for continuous classes, 9th European Conference on Machine Learning, 1997
- D. A. Cohn, Z. Ghahramani, & M. Jordan, Active Learning with Statistical Models, JAIR, 1996.