Data Analytics
EEE 4774 & 6777

Module 5 - Regression

Linear Regression

Spring 2022

Linear Regression

Training Data

	size x_1	# rooms x_2	location x_3	pool x_4	price ($100k) t
House 1	2500	4	3	0	320
	3000	5	2	1	517
House N					

$X =$

Goal: Infer $w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$ from training data

Test: Given $x = (x_1, x_2, x_3, x_4)$ Use $w^T x$ to predict the house price.

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_M x_M$$

(w_0: offset value)

$$y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$$

(y: output, e.g. house price; $\boldsymbol{w}$: parameter vector; $\boldsymbol{x}$: input vector, e.g. size, location, #rooms, etc.)

$w_{opt} = \arg\min_w E(w)$

- Input variables (regressors, independent variables, predictors, features): $\boldsymbol{x}$
- Output variables (dependent variables, targets): y
- Unknown parameters (regression coefficients): $\boldsymbol{w}$

Loss function / Objective function

$$E(t, w) = \frac{1}{N} \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

MSE: mean squared error; squared error; prediction error; sum of squared error

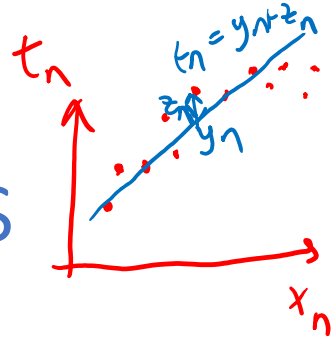
$$MAE = \frac{1}{N} \sum_{n=1}^{N} |t_n - w^T x_n|$$

MAE: mean absolute error

$$\min_w \sum_{n=1}^{N} (t_n - w^T x_n)^2 = \min_w \frac{1}{N} \sum (t_n - w^T x_n)^2$$

$\frac{1}{N} \sum (t_n - w^T x_n)^2$: MSE

Maximum Likelihood and Least Squares



max likeli. func. min SSE loss func.

- Assume observations from a deterministic function with added Gaussian noise: $y(w,x)$ random fluctuation

$$t_n = \boldsymbol{w}^T \boldsymbol{x}_n + z_n \quad \text{where} \quad z_n \sim \mathcal{N}(0, \beta^{-1})$$

β: inverse variance $= \frac{1}{\sigma^2}$ (precision)

which is the same as saying,

mean, variance

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}) \sim t_n$$

- Given observed inputs, $\boldsymbol{X}^{N \times M}$, and targets, $\boldsymbol{t} = [t_1, \ldots, t_N]^T$ we obtain the likelihood function

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T \boldsymbol{x}_n, \beta^{-1})$$

Maximum Likelihood and Least Squares

$$t_n = w^T x_n + z_n$$

Typically, standard Gaussian with $\beta = 1$

$z_n \sim \mathcal{N}(0, 1)$

$$\log p(t|w,\beta) = \sum_{n=1}^{N} \log \mathcal{N}(t_n | w^T x_n, \beta^{-1})$$

$$w_{ML} = \arg\min_w \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

Sum-of-squares error $E_D(\boldsymbol{w})$

Computing the gradient and setting it to zero yields

$$w_{ML} = \left(\sum_{n=1}^{N} x_n x_n^T \right)^{-1} \sum_{n=1}^{N} t_n x_n$$

$$w_{ML} = (X^T X)^{-1} X^T t$$

X: input data matrix; t: output variable vector

1-dim. $w_{ML} = \frac{\sum_{i=1}^{N} x_i t_i}{\sum_{i=1}^{N} x_i^2}$

$\frac{\beta}{2\pi} e^{-\frac{\beta}{2}(t_n - w^T x_n)^2}$

$-\frac{\beta}{2}(t_n - w^T x_n)^2 + \frac{1}{2}\log\frac{\beta}{2\pi}$

$\beta = 1$

$X = \begin{bmatrix} - & x_1 & - \\ & \vdots & \\ - & x_N & - \end{bmatrix}$

$\min \sum_{n=1}^{N} (t_n - w^T x_n)^2$

$w_{ML} = \arg\max_w \log p(t|w,\beta)$

ML

$\sum_n x_n (x_n^T w) = \sum_n t_n x_n$

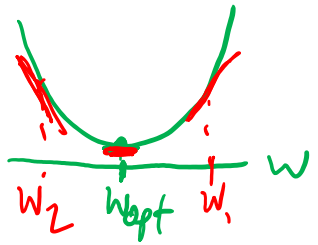
LS

$\left(\sum_n x_n x_n^T\right) w = \sum_n t_n x_n$

$(X^T X) w = X^T t$

$(X^T X)^{-1}(X^T X) w = (X^T X)^{-1} X^T t$

ML=LS for the Gaussian case



Gradient = 0

$\frac{\partial}{\partial w} \frac{1}{2} \sum_{n=1}^{N} (t_n - w_{ML}^T x_n)^2 = \frac{2}{2} \sum_{n=1}^{N} (t_n - w_{ML}^T x_n)(-x_n) = 0$

$\sum_{n=1}^{N} t_n x_n = \sum_{n=1}^{N} (w_{ML}^T x_n) x_n$

$w_{ML}^T x_n$: scalar; x_n: vector

Geometry of Least Squares

$y_n = w^T x_n$

- Consider

$$\boldsymbol{y} = X\boldsymbol{w}_{ML} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_M]\boldsymbol{w}_{ML}$$

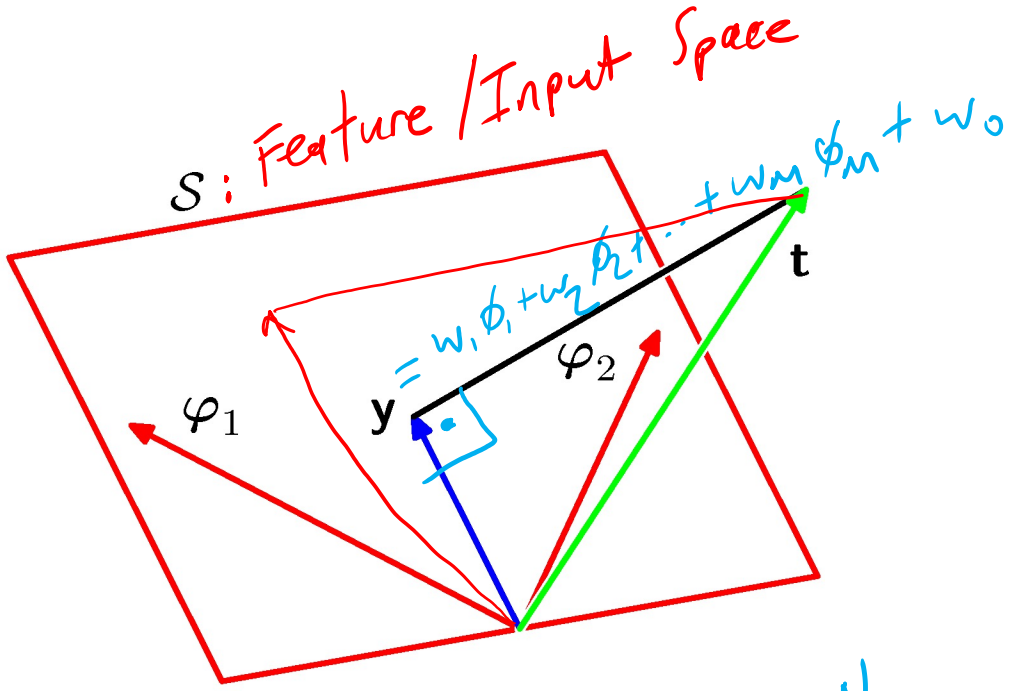
feature vectors

$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$

$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \qquad \mathbf{t} \in \mathcal{T}$

N-dimensional

M-dimensional



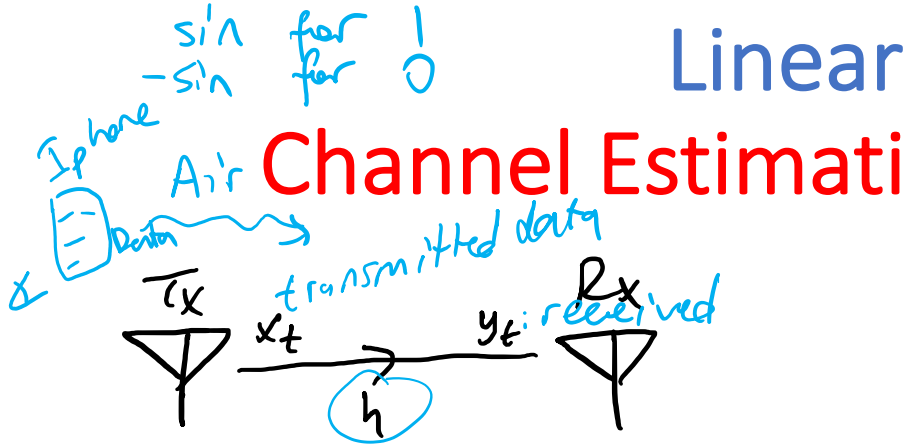
- S is spanned by $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_M$.
- $\boldsymbol{w}_{ML}$ minimizes the distance between t and its orthogonal projection on S, i.e. y.

$$\|\vec{t} - \vec{y}\|^2 = \|\vec{t} - X\vec{w}\|^2 = \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

SSE

Linear Regression Example:

Channel Estimation in Wireless Communications



$$y_t = h x_t + z_t$$

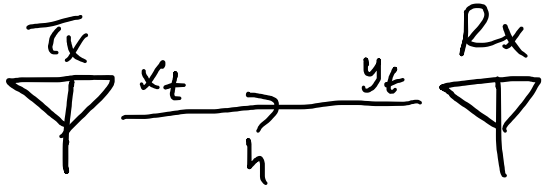
y_t: received
h: channel coefficient
x_t: transmitted
z_t: noise

$$z_t \sim \mathcal{N}(0, \sigma^2)$$

Gaussian noise

Given $\{x_t, y_t\}_t$ (pilot signal) estimate h

Linear Regression Example:
Channel Estimation in Wireless Communications



$$y_t = h x_t + z_t$$

y_t: received; h: channel coefficient; x_t: transmitted; z_t: noise

$$z_t \sim \mathcal{N}(0, \sigma^2)$$

Given $\{x_t, y_t\}_t$ estimate h

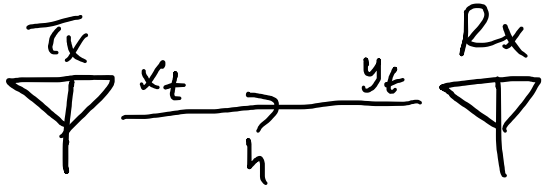
Least Squares Estimation

$$\hat{h}_{LS} = \arg\min_h \sum_{t=1}^{T} (y_t - h x_t)^2$$

$$\frac{\partial}{\partial h} \sum_{t=1}^{T} (y_t - h x_t)^2 \Big|_{h=\hat{h}_{LS}} = \sum_{t=1}^{T} 2(y_t - \hat{h}_{LS} x_t)(-x_t) = 0 \Rightarrow$$

$$\boxed{\hat{h}_{LS} = \frac{\sum_{t=1}^{T} y_t x_t}{\sum_{t=1}^{T} x_t^2}}$$

Linear Regression Example:
Channel Estimation in Wireless Communications



$$y_t = h x_t + z_t \sim \mathcal{N}(h x_t, \sigma^2)$$

y_t: received; h: channel coefficient; x_t: transmitted; z_t: noise

$$z_t \sim \mathcal{N}(0, \sigma^2)$$

Given $\{x_t, y_t\}_t$ estimate h

Least Squares Estimation

$$\hat{h}_{LS} = \arg\min_h \sum_{t=1}^{T} (y_t - h x_t)^2$$

$$\frac{\partial}{\partial h} \sum_{t=1}^{T} (y_t - h x_t)^2 \Big|_{h=\hat{h}_{LS}} = \sum_{t=1}^{T} 2(y_t - \hat{h}_{LS} x_t)(-x_t) = 0 \Rightarrow$$

$$\boxed{\hat{h}_{LS} = \frac{\sum_{t=1}^{T} y_t x_t}{\sum_{t=1}^{T} x_t^2}}$$

Likelihood function: $p(\{x_t, y_t\} | h) = \prod_{t=1}^{T} p(x_t, y_t | h) = \prod_{t=1}^{T} \mathcal{N}(h x_t, \sigma^2)$

$$\prod_{t=1}^{T} \frac{e^{-\frac{(y_t - h x_t)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} = \frac{e^{-\sum_{t=1}^{T} \frac{(y_t - h x_t)^2}{2\sigma^2}}}{(2\pi\sigma^2)^{T/2}}$$

ML Estimation

$$\hat{h}_{ML} = \arg\max_h \log \frac{e^{-\sum_{t=1}^{T} \frac{(y_t - h x_t)^2}{2\sigma^2}}}{(2\pi\sigma^2)^{T/2}}$$

$$\hat{h}_{ML} = \arg\min_h \sum_{t=1}^{T} (y_t - h x_t)^2$$

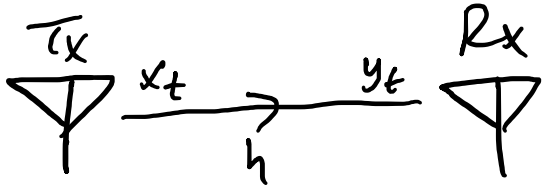
$$\boxed{\hat{h}_{ML} = \hat{h}_{LS}}$$

Linear Regression Example:

Channel Estimation in Wireless Communications

$y_t = h_1 x_{1t} + h_2 x_{2t} + \dots + h_N x_{Nt}$

$\vec{x}_t = \begin{bmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{Nt} \end{bmatrix}$

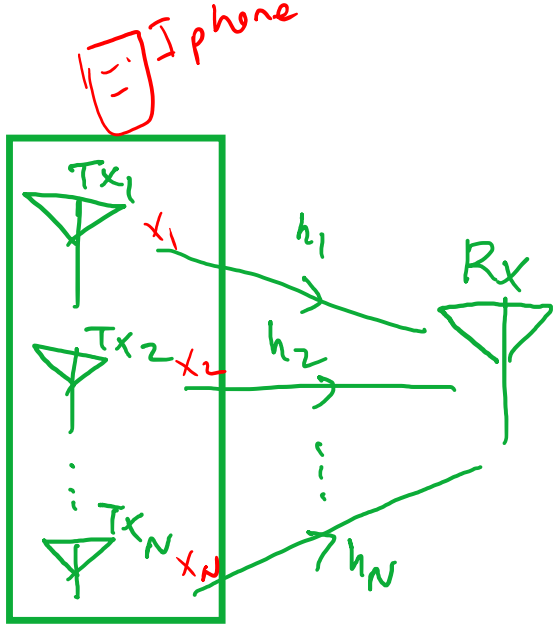


$$y_t = h x_t + z_t$$

y_t: received; h: channel coefficient; x_t: transmitted; z_t: noise

$$z_t \sim \mathcal{N}(0, \sigma^2)$$

Given $\{x_t, y_t\}_t$ estimate h



Least Squares Estimate

MISO: $y_t = X_t' H + z_t$ $\quad H = \begin{bmatrix} h_1 \\ \vdots \\ h_N \end{bmatrix}$

$$\hat{H}_{LS} = \arg\min_H \sum_{t=1}^{T} (y_t - X_t' H)^2$$

$\frac{\partial}{\partial H}$

$$\sum_{t=1}^{T} 2(y_t - X_t' \hat{H}_{LS})(-X_t) = 0$$

$$\sum_{t=1}^{T} (X_t X_t') \hat{H}_{LS} = \sum_{t=1}^{T} X_t y_t$$

$X^{T \times N} = \begin{bmatrix} X_1' \\ \vdots \\ X_T' \end{bmatrix}$

$$\hat{H}_{LS} = (X^T X)^{-1} X^T Y$$

$Y = X H + Z$

Least Squares Estimation

$$\hat{h}_{LS} = \arg\min_h \sum_{t=1}^{T} (y_t - h x_t)^2$$

$$\frac{\partial}{\partial h} \sum_{t=1}^{T} (y_t - h x_t)^2 \Big|_{h=\hat{h}_{LS}} = \sum_{t=1}^{T} 2(y_t - \hat{h}_{LS} x_t)(-x_t) = 0 \Rightarrow$$

$$\hat{h}_{LS} = \frac{\sum_{t=1}^{T} y_t x_t}{\sum_{t=1}^{T} x_t^2}$$

ML Estimation

$$\hat{h}_{ML} = \arg\max_h \log \frac{e^{-\sum_{t=1}^{T} \frac{(y_t - h x_t)^2}{2\sigma^2}}}{(2\pi\sigma^2)^{T/2}}$$

$$\hat{h}_{ML} = \arg\min_h \sum_{t=1}^{T} (y_t - h x_t)^2$$

$$\hat{h}_{ML} = \hat{h}_{LS}$$

Linear Regression Example:
Autoregressive (AR) Model

- AR(p): the next value depends on the previous p values

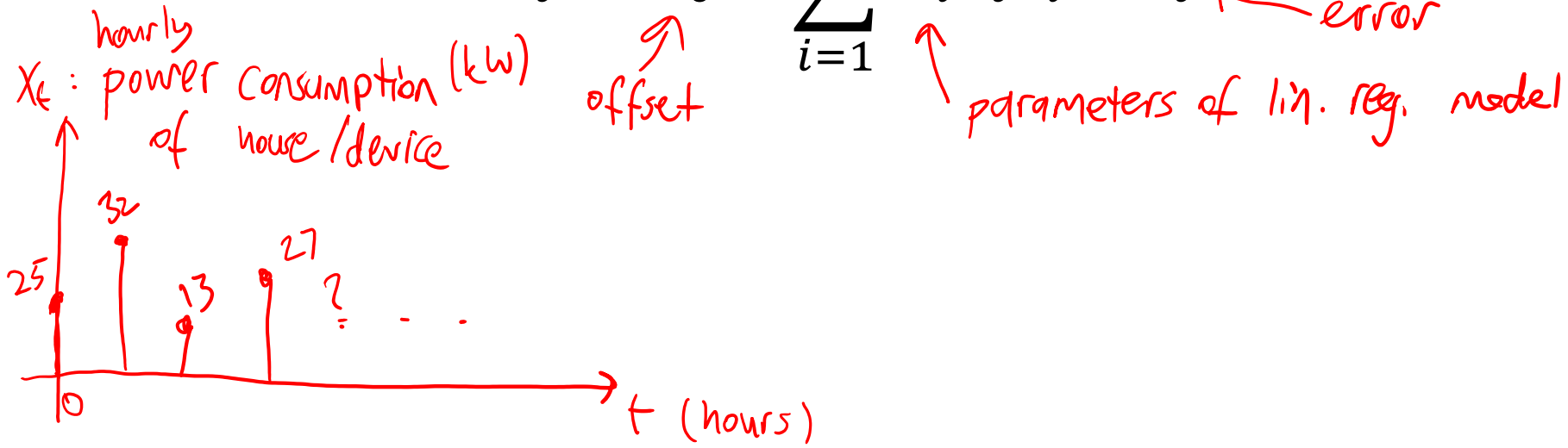
$$x_t = w_0 + w_1 x_{t-1} + w_2 x_{t-2} + \cdots + w_p x_{t-p} + z_t$$

$$x_t = w_0 + \sum_{i=1}^{p} w_i x_{t-i} + z_t$$

z_t ← error

w_0: offset

w_i: parameters of lin. reg. model



Regularized Least Squares

- Consider the error function:

$$\min \left\{ \beta E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \right\}$$

SSE (MSE), Likelihood

Data term + Regularization term

- With the sum-of-squares error function and a quadratic regularizer, we get

$$\text{Min} \left\{ \frac{\beta}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^{\mathrm{T}} \mathbf{w} \right\}$$

inverse variance in ML approach

SSE

$\mathbf{w}^{\mathrm{T}}\mathbf{w} = \sum_{i=1}^{M} w_i^2$

Quadratic regularization penalty

λ is called the regularization coefficient.

- which is minimized by

$$\mathbf{w} = \left(\frac{\lambda \mathbf{I}}{\beta} + \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}.$$

data matrix

$\beta = 1$

$\mathbf{w} = (\lambda \mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$

w/o regularization

$\mathbf{w}_{LS} = (\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$

$\min_{\mathbf{w}} E_D(\mathbf{w})$

Choose w values to minimize loss function

Complex models with many parameters have a very high-dimensional search space for optimizing parameters. In turn, training loss can be very well minimized (i.e. over fitting to train. data happens) in this high-dim. unrestricted search space.

by choosing extreme values for parameters.

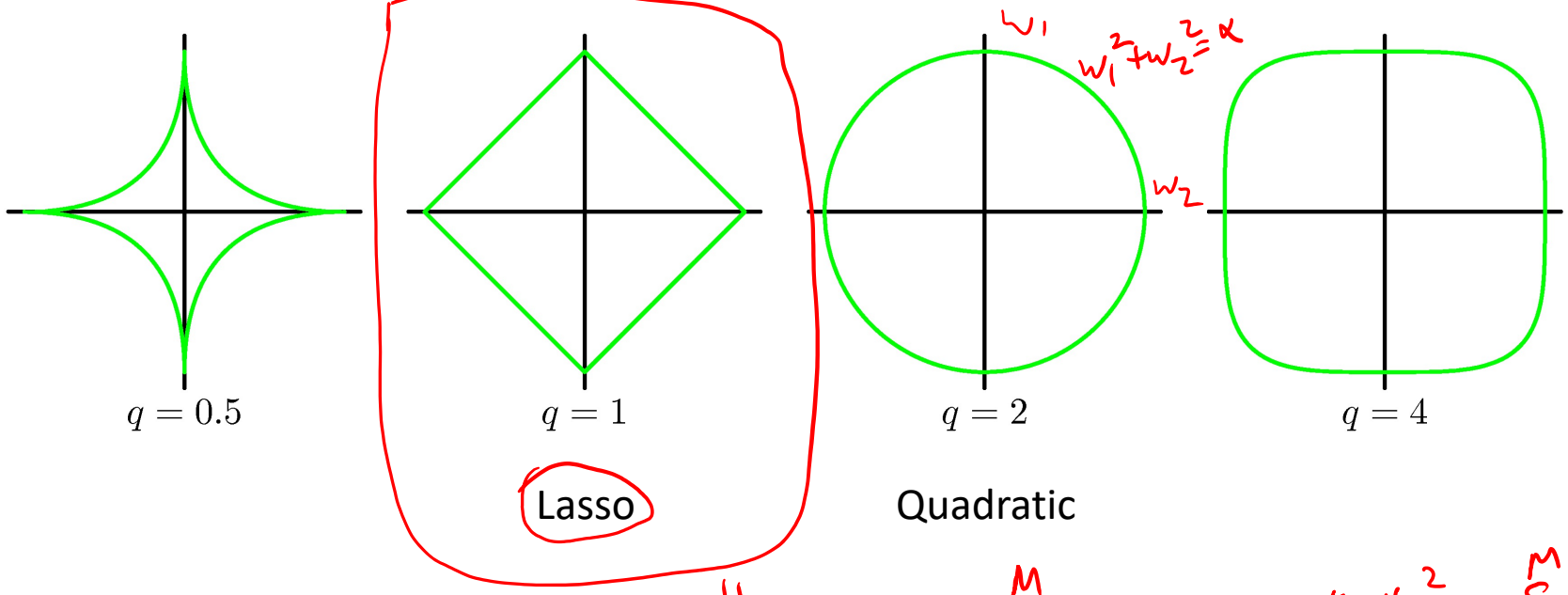
e.g. $\mathbf{w} = \begin{bmatrix} 10^{12} \\ -10^{15} \\ \vdots \end{bmatrix}$

Regularized Least Squares

- With a more general regularizer, we have

$$\min \left\{ \frac{\beta}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^{M} |w_j|^q \right\}$$

$$\min \sum_{n=1}^{N} (t_n - w^T \phi)^2 \quad \text{s.t.} \quad \lambda w^T w \le \alpha$$

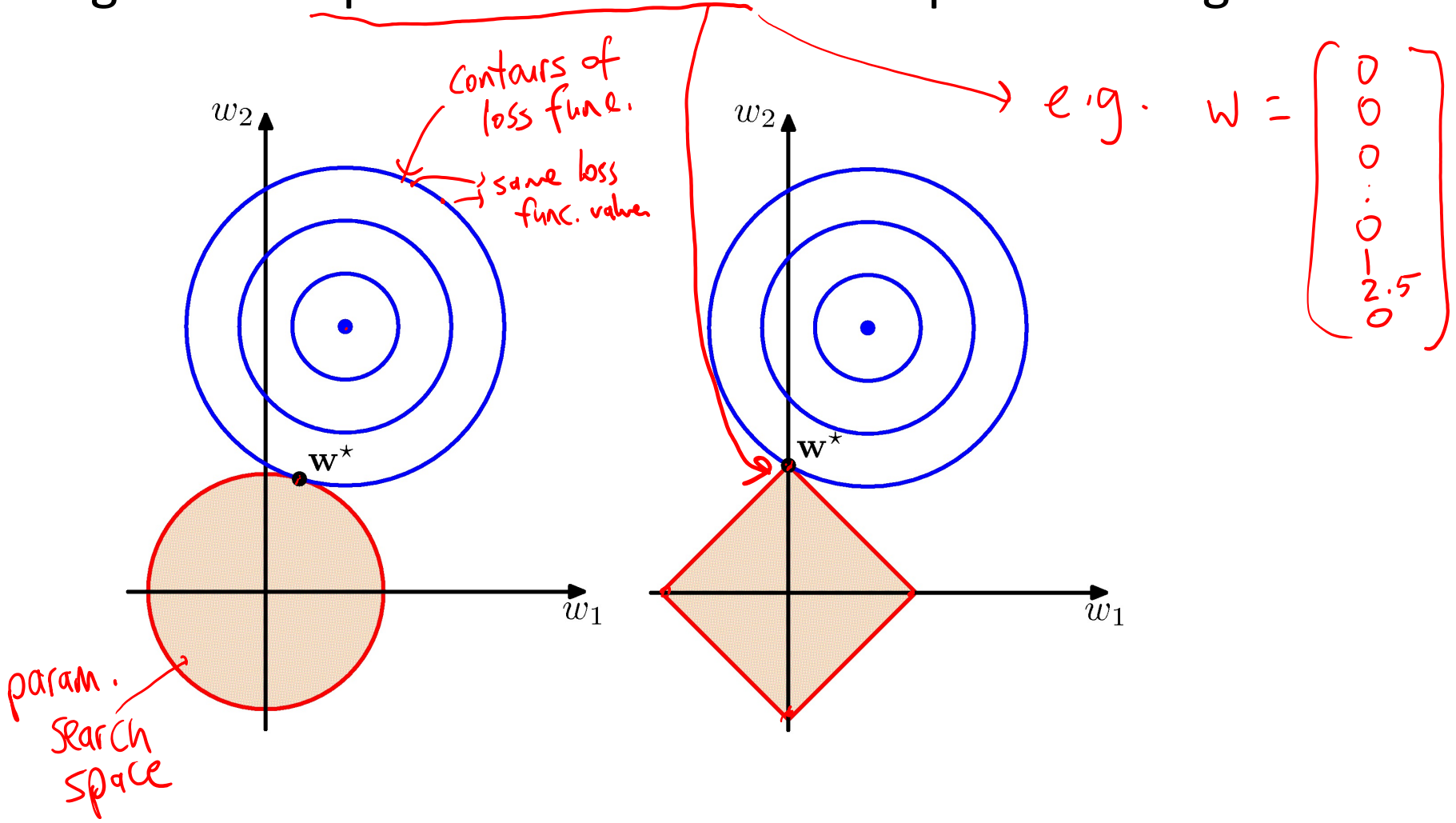


L1 penalty

$$\|w\|_1 = \sum_{j=1}^{M} |w_j| \qquad \|w\|_2^2 = \sum_{j=1}^{M} w_j^2$$

Regularized Least Squares

- Lasso tends to generate sparser solutions than a quadratic regularizer.



$MSE = E[\{y(x)-t\}^2] = E[\{y(x)-h(x)+h(x)-t\}^2] = E[\{y(x)-h(x)\}^2] + E[\{h(x)-t\}^2] + 2E[\{y(x)-h(x)\}\{h(x)-t\}]$

How to select regularization coefficient λ? The Bias-Variance Decomposition

Cross-term = 0

$E[y(x)-h(x)]\,E[h(x)-t]$

- Recall the *expected squared loss, i.e., Mean Squared Loss (MSE),*

$$MSE = \mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

loss

where

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t\,p(t|\mathbf{x})\,\mathrm{d}t.$$

Best Bayesian predictor that minimizes MSE

given information

posterior mean

posterior prob. dist. of output var. given input var.

$E[h(x)] = E_x[E[t|x]]$

Tower property $= E[t]$

$E[h(x)-t] = E[t] - E[t] = 0$

- The second term of E[L] corresponds to the noise inherent in the random variable t.
- What about the first term?

$MSE = E[\{y(x)-h(x)\}^2] + E[\{h(x)-t\}^2]$

Focus on this

The Bias-Variance Decomposition

- Suppose we were given multiple data sets, each of size N. Any particular data set, D, will give a particular function y(x;D). We then have

$$\mathbb{E}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right]$$

$$= \mathbb{E}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2\right]$$

$$= \mathbb{E}\Big[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2$$

$$+2\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}.\Big]$$

$\mathbb{E}[y - \mathbb{E}[y]] = 0$

Variance of our predictor y

Bias of our predictor y

$$= \mathbb{E}\left[\{y - \mathbb{E}[y]\}^2\right] + \mathbb{E}\left[\{\mathbb{E}[y] - h(x)\}^2\right]$$

mean

The Bias-Variance Decomposition

- Taking the expectation over D yields

Controllable part of MSE →

$$\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right]$$
$$= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.$$

The Bias-Variance Decomposition

- Thus we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

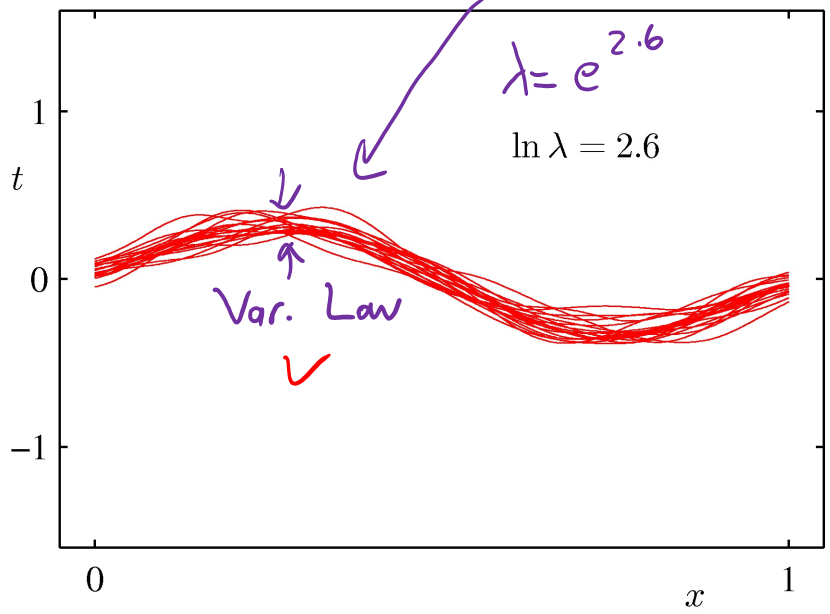
$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

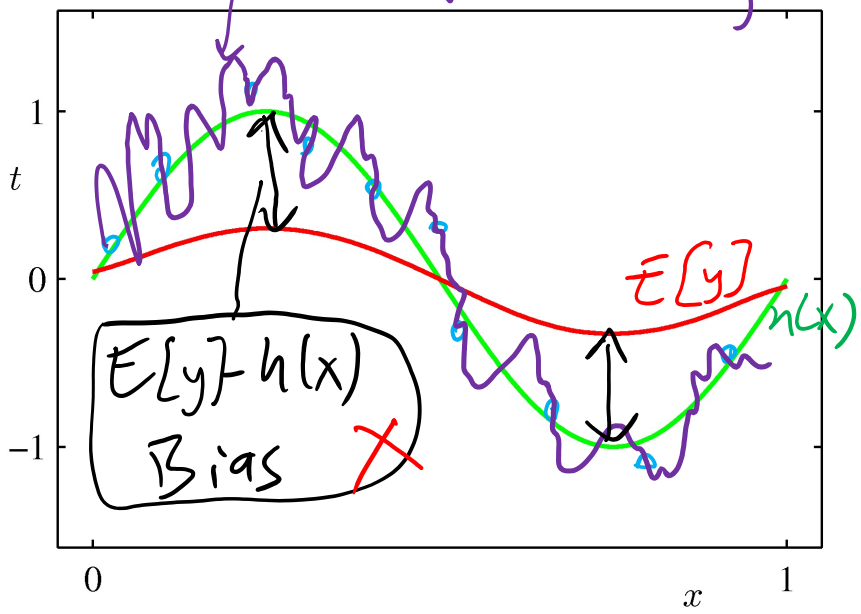
$y(\tilde{x}) = w^T \tilde{x}$ $\quad \tilde{x} = \begin{bmatrix} x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$ $\quad y(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_M x^M$

The Bias-Variance Decomposition

$$\min \sum_{i=1}^{N} (y(x_i) - t_i)^2 + \frac{\lambda}{2} \|w\|^2$$

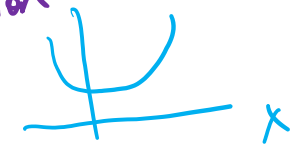
- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.





M=1 : $y(x) = w_0 + w_1 x$

M=2 : $y(x) = w_0 + w_1 x + w_2 x^2$



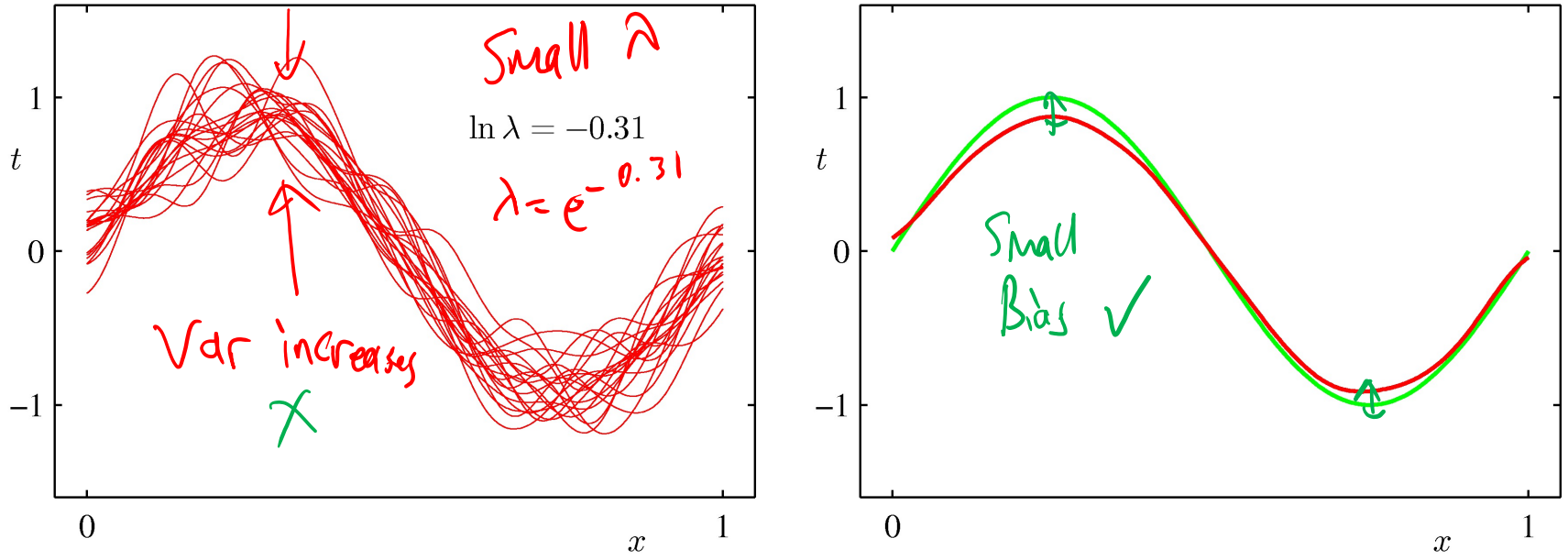
M=3 $y(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

M ↑ ↑ modeling power (model complexity) flexibility

Note: Complex models w/ many parameters should only be used for complex problems w/ abundance of data to estimate in training

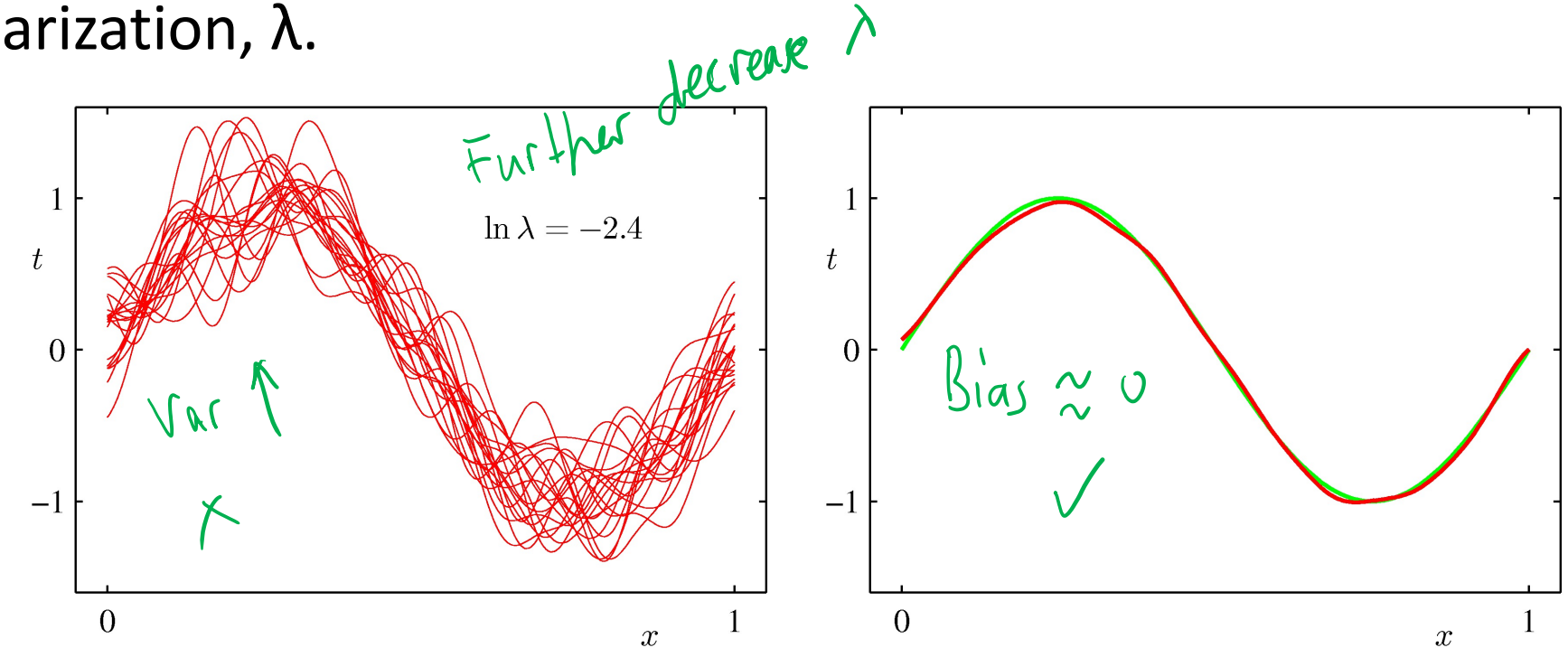
The Bias-Variance Decomposition

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



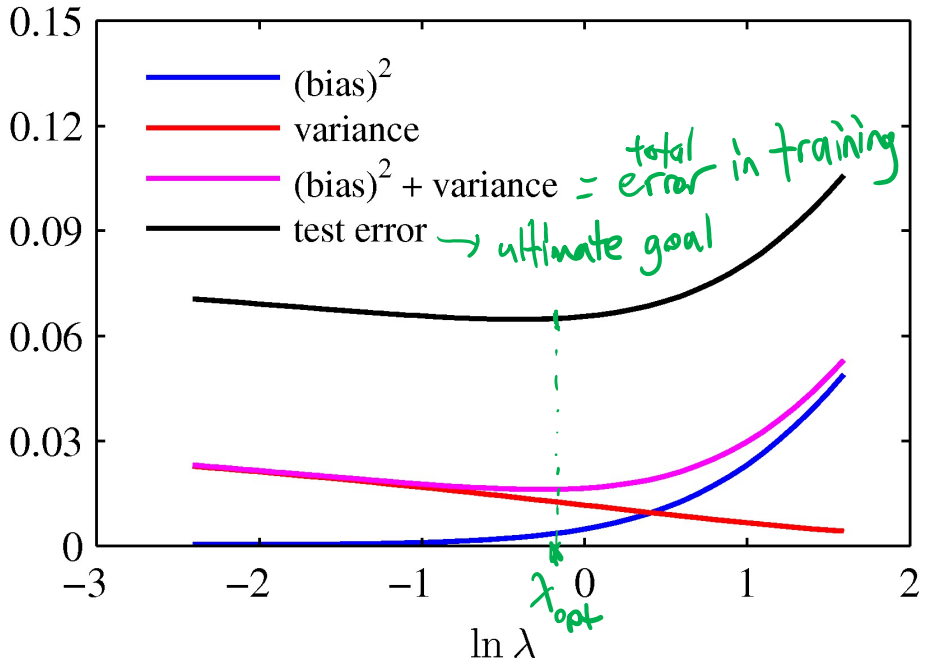
The Bias-Variance Decomposition

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



The Bias-Variance Trade-off

- From these plots, we note that an over-regularized model (large λ) will have a high bias, while an under-regularized model (small λ) will have a high variance.



$t = w^T x + z$

Bayesian Linear Regression

- Define a conjugate prior over w

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0).$$

prior covar.

prior mean

- Combining this with the likelihood function and using results for marginal and conditional Gaussian distributions, gives the posterior

- where

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

posterior

data matrix

$$\begin{aligned}
\mathbf{m}_N &= \mathbf{S}_N \left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta \mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \right) \\
\mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}.
\end{aligned}$$

Bayesian Linear Regression

- A common choice for the prior is

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

$S_0 = \begin{bmatrix} \alpha^{-1} & & 0 \\ & \alpha^{-1} & \\ & & \ddots \\ 0 & & \alpha^{-1} \end{bmatrix}$

- for which

$$\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}. \end{aligned}$$

$= \beta \left(\alpha I + \beta \Phi^T \Phi\right)^{-1} \Phi^T t$

$= \left(\frac{\alpha}{\beta} I + \Phi^T \Phi\right)^{-1} \Phi^T t$

$= w_{LS+Quad\ Reg}$

$w_{Bay} = m_N$

$\alpha \equiv \lambda$ ← reg coef.

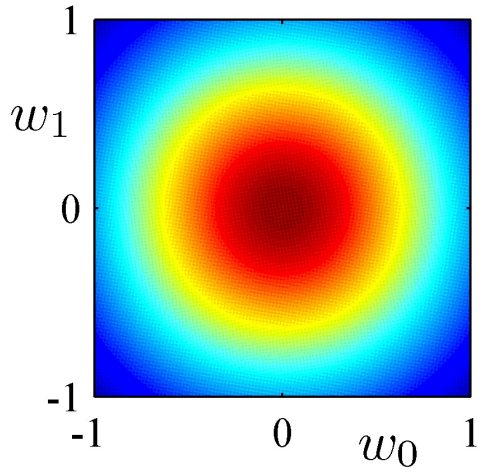
- Next we consider an example …

Bayesian Linear Regression

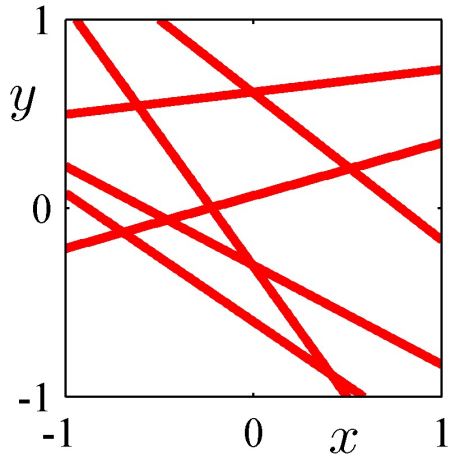
0 data points observed

$y = w_0 + w_1 x$

Prior



Data Space: samples from w_0 and w_1



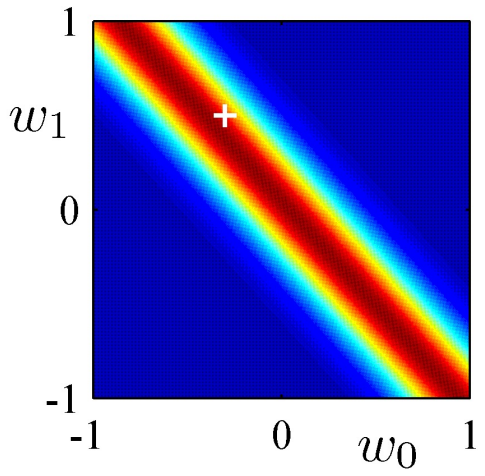
$S_0 = \begin{bmatrix} \alpha^{-1} & 0 \\ 0 & \alpha^{-1} \end{bmatrix}$

Bayesian Linear Regression

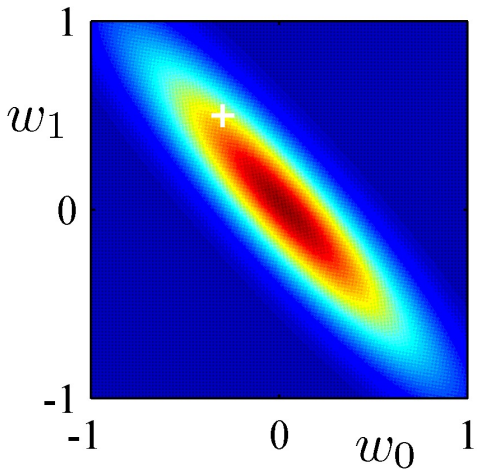
1 data point observed

prior for 2nd data

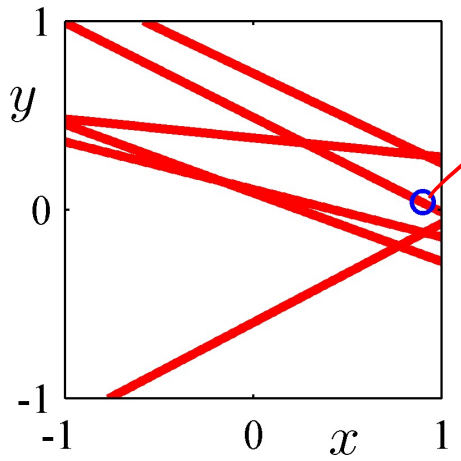
Likelihood



Posterior



Data Space



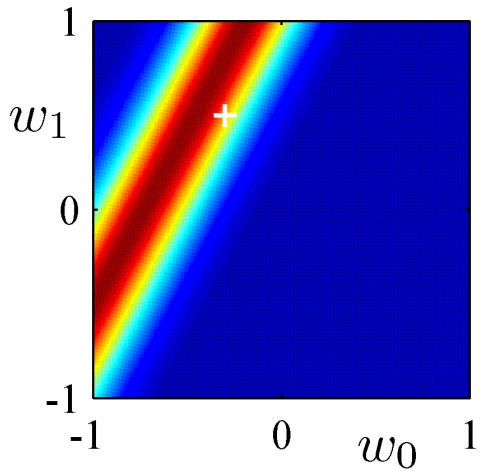
data

Likelihood . Prior

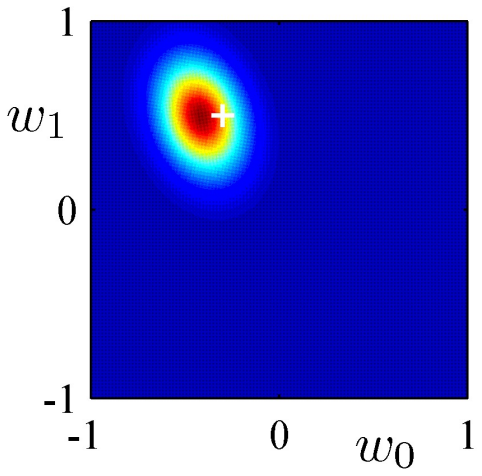
Bayesian Linear Regression

2 data points observed

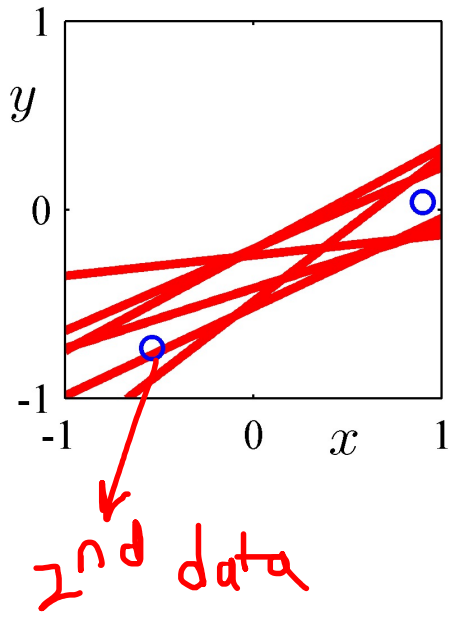
Likelihood



Posterior



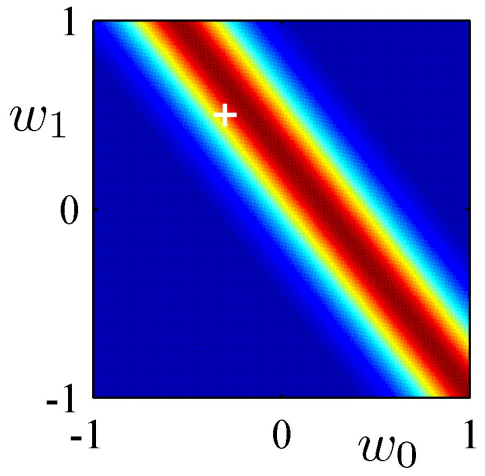
Data Space



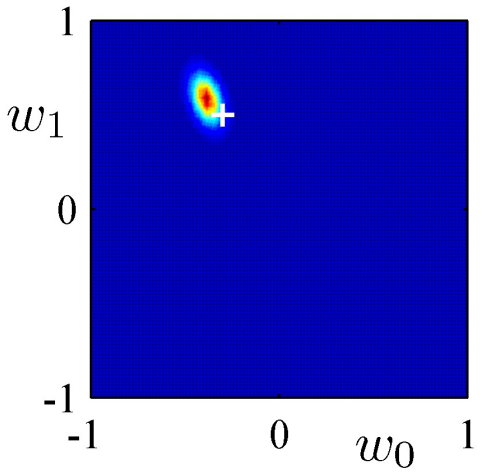
Bayesian Linear Regression

20 data points observed

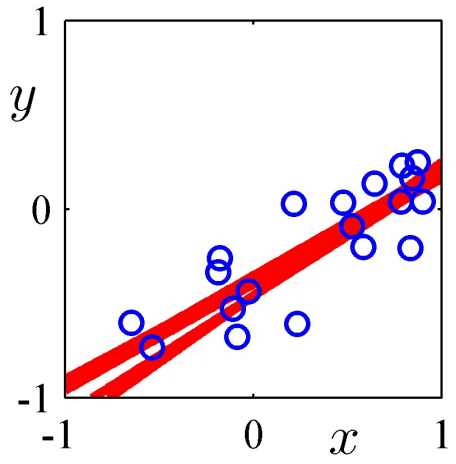
Likelihood



Posterior



Data Space



Sequential Learning

$$MSE = \mathbb{E}\left[(t_n - w^T\phi(x_n))^2\right], \quad \hat{w}_{MSE} = \arg\min_w MSE(w)$$

$$\Rightarrow \hat{w}_{MSE} = \mathbb{E}[\phi\phi^T]^{-1}\mathbb{E}[\phi^T t]$$

estimates

$$\hat{w}_{LS} = \arg\min_w \sum_{n=1}^{N} (t_n - w^T\phi(x_n))^2$$

$$= \underbrace{\left(\sum_n \phi\phi^T\right)^{-1}\sum_n \phi t_n}_{\text{Batch Learning}}$$

- Data items considered one at a time (a.k.a. online learning); use stochastic (sequential) gradient descent:

Gradient Descent : Iterative optimization – move towards negative of gradient of cost function

$$\frac{1}{N}\sum_{n=1}^{N}(t_n - w^T\phi)\phi$$

MSE

$$\mathbb{E}\left[\nabla_w (t_n - w^T\phi)^2\right]$$

$$= \mathbb{E}\left[2(t_n - w^T\phi)(-\phi)\right]$$

Difficult to compute online

Stochastic Gradient Descent : Use single-point estimate $(t_n - w^T\phi)\phi$ can be estimated by

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n$$
$$= \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n))\boldsymbol{\phi}(\mathbf{x}_n).$$

- This is known as the *least-mean-squares (LMS) algorithm*. Issue: how to choose η?