

REGRESSION PART 3: REGULARIZATION

Hsin-Min Lu

盧信銘

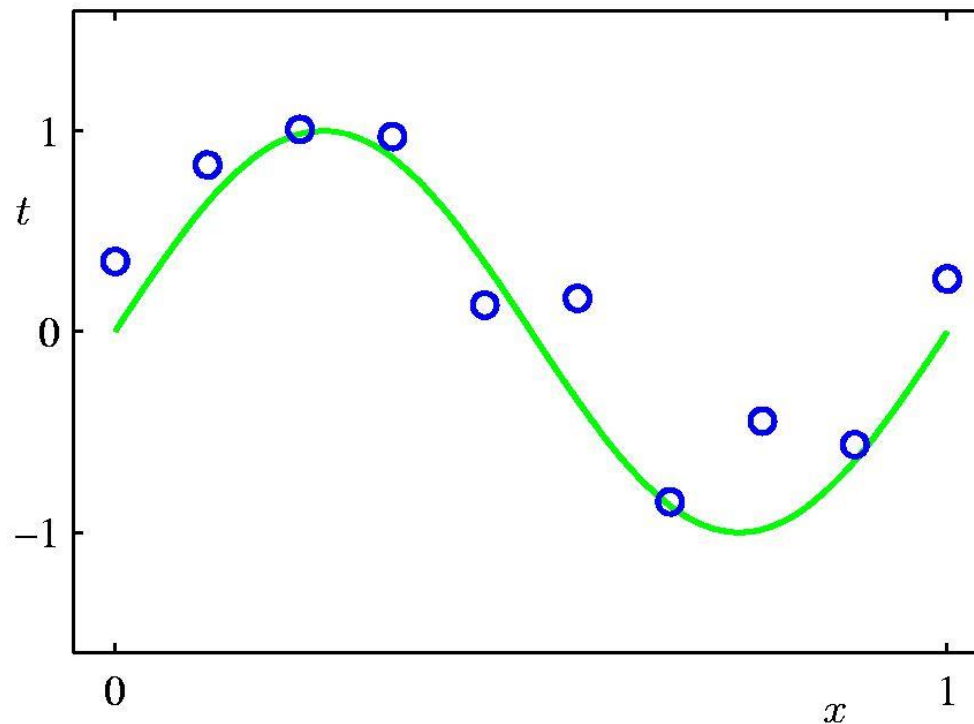
台大資管系

Should We Always Prefer More Features?

- In the previous example, we have seen that additional features allow us to capture additional variations of the outcome, and thus provides better prediction.
- The key question is should we always prefer more features when constructing a model?
- Is there any drawback when we add large amounts of features?



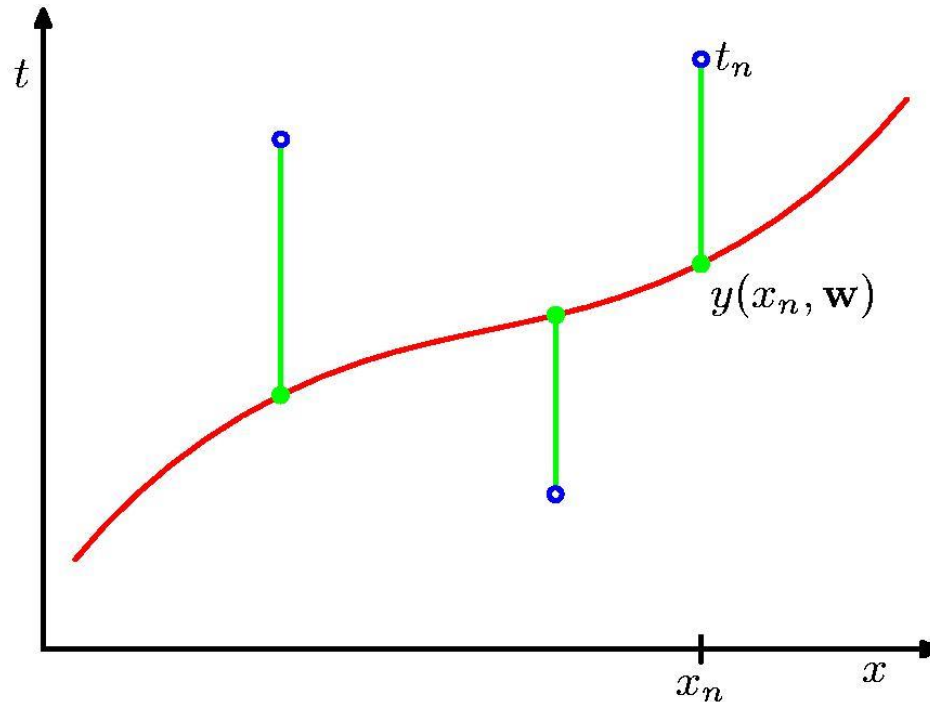
Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_j x^j$$



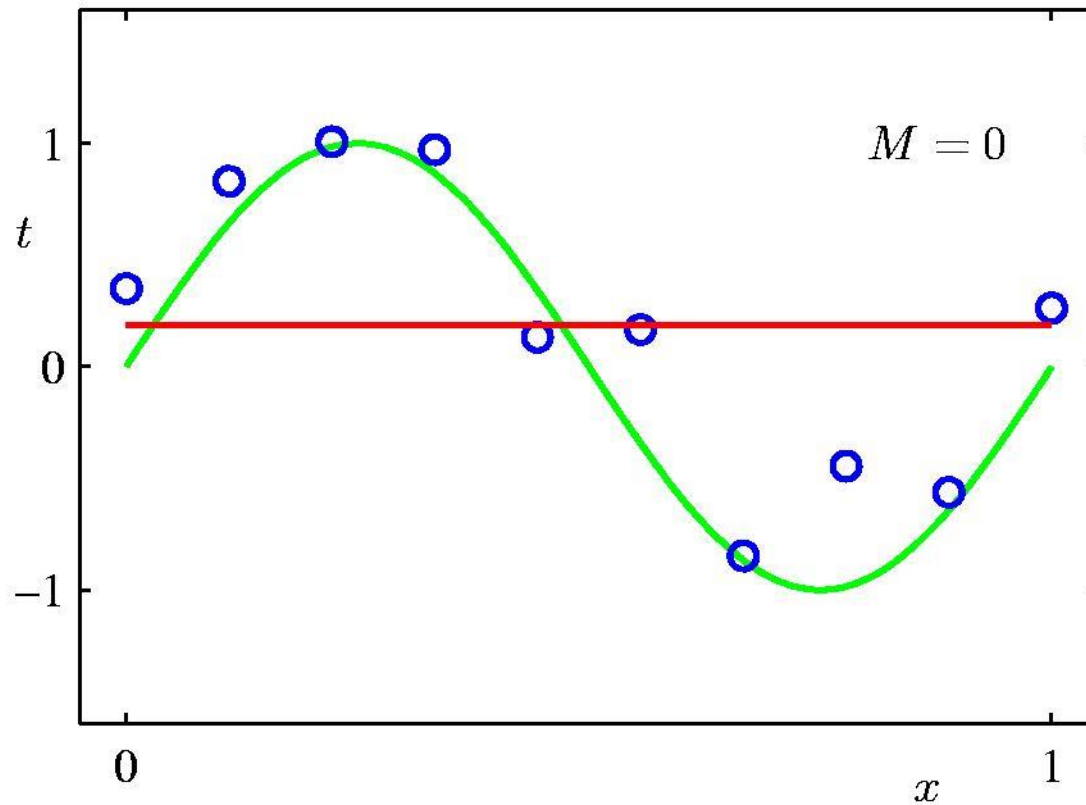
Sum-of-Squares Error Function



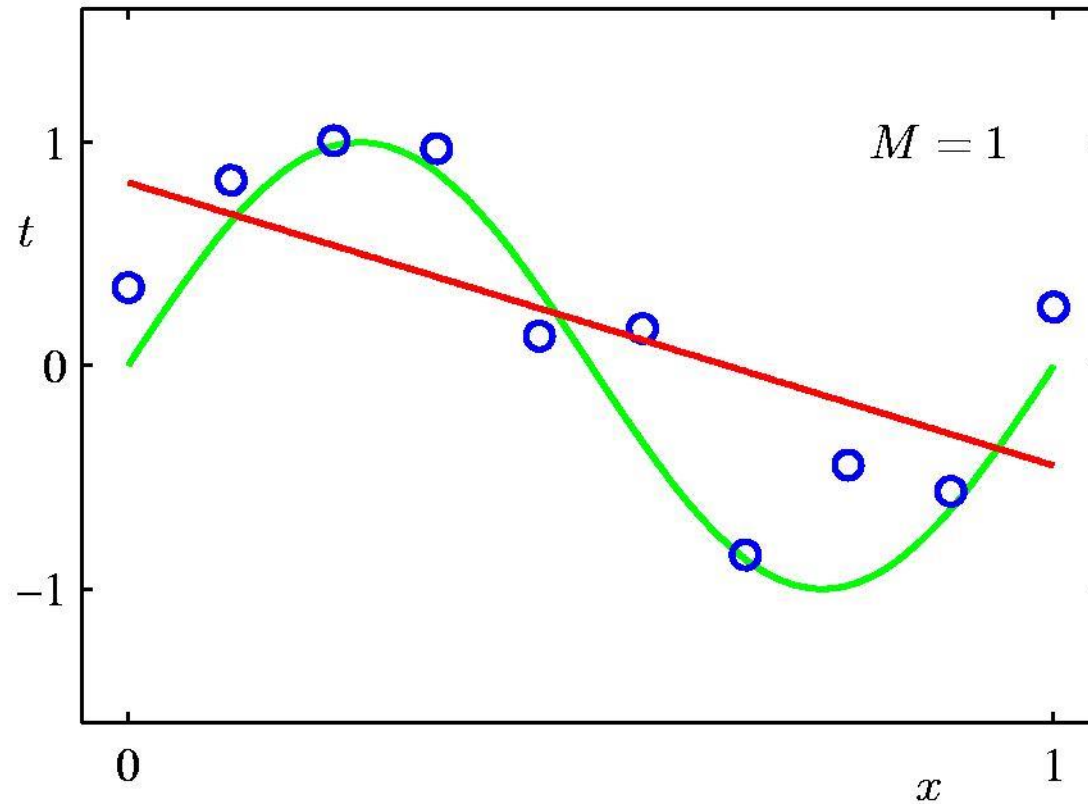
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$



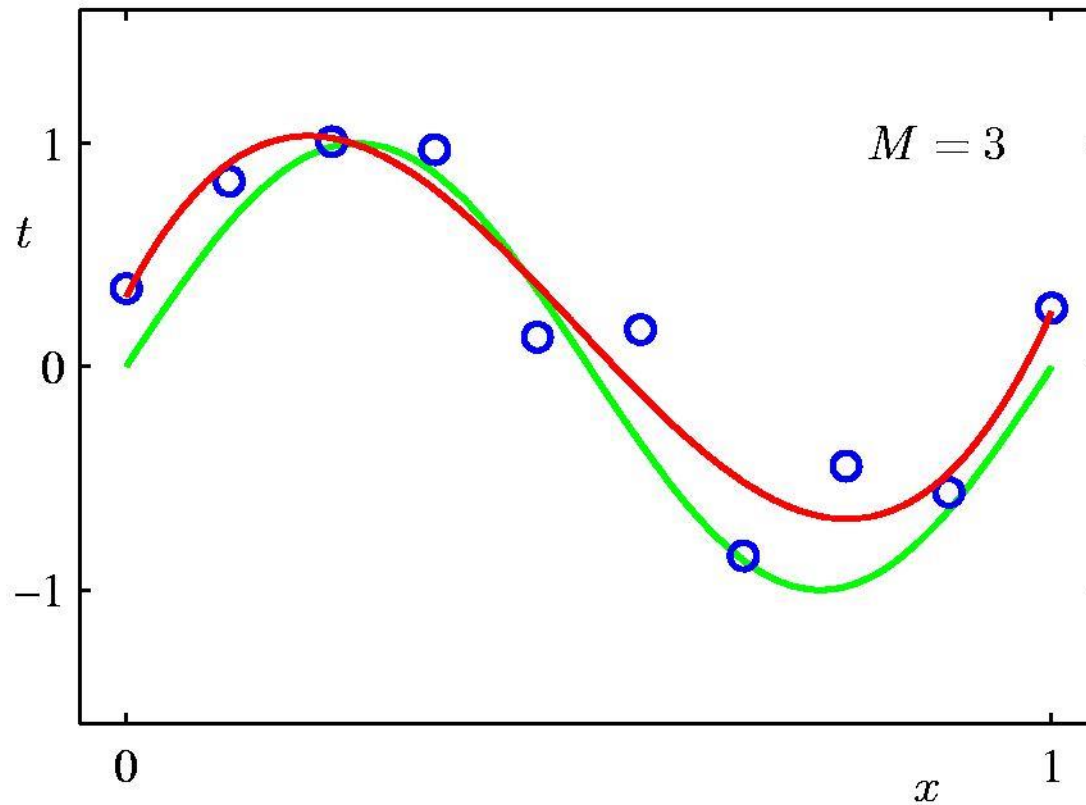
0th Order Polynomial



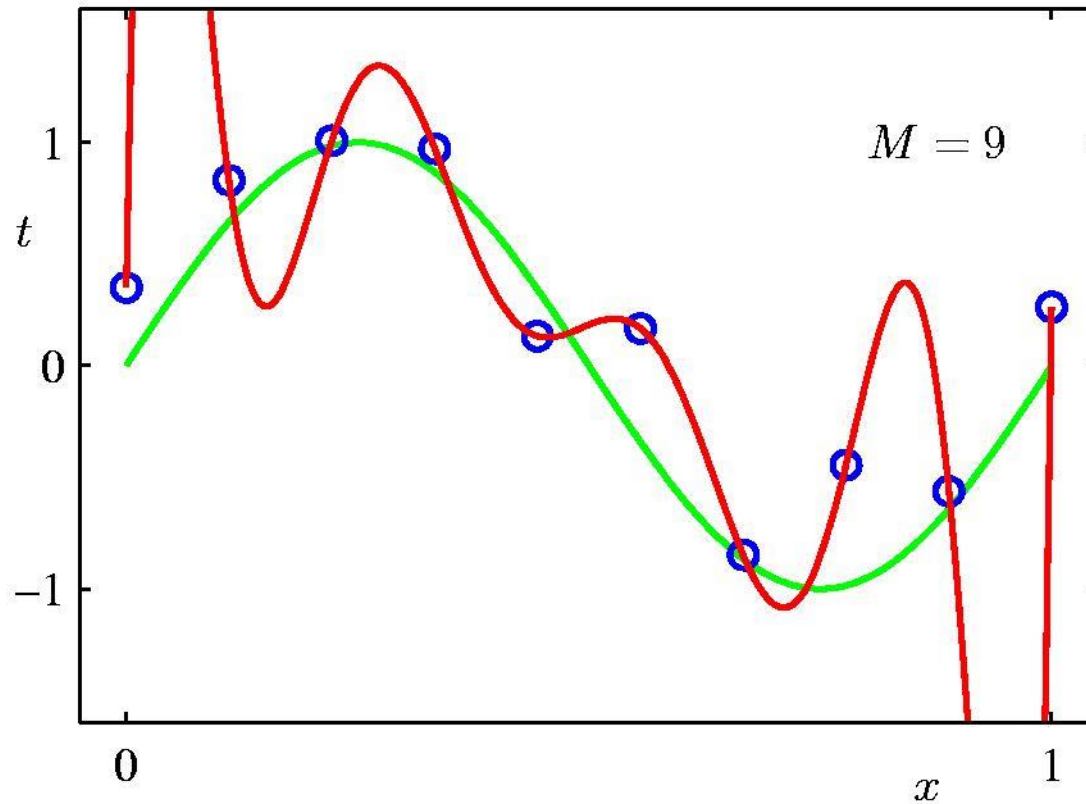
1st Order Polynomial



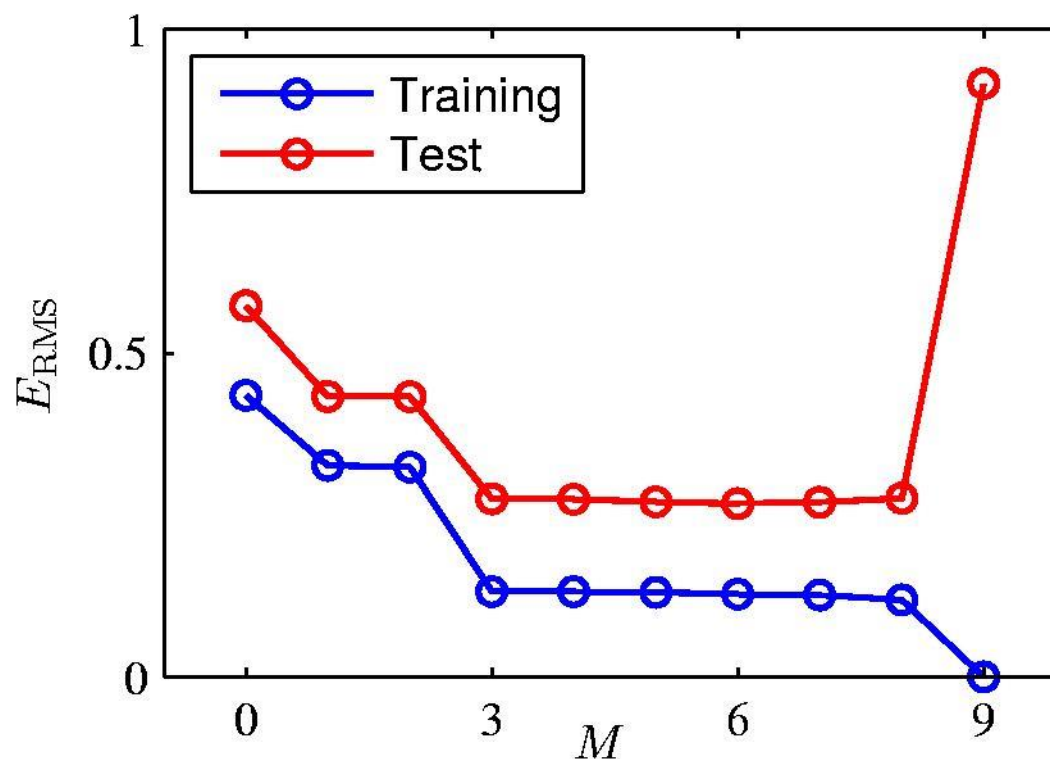
3rd Order Polynomial



9th Order Polynomial



Over-fitting



Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$



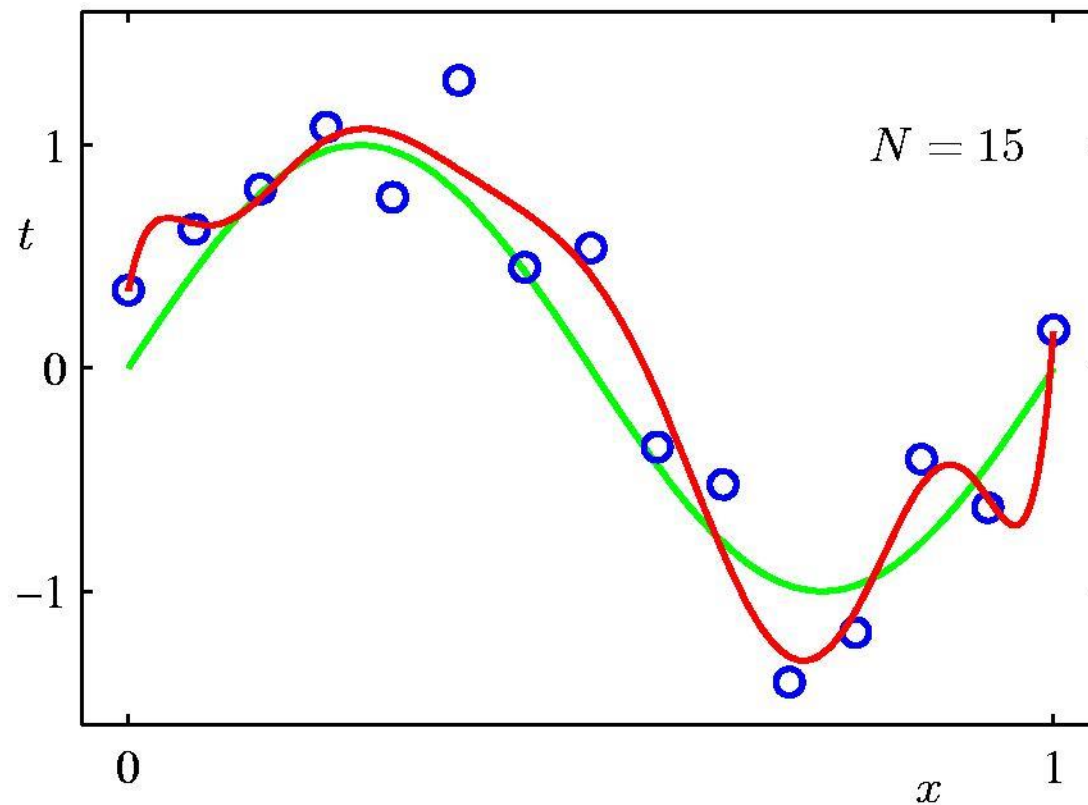
Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43



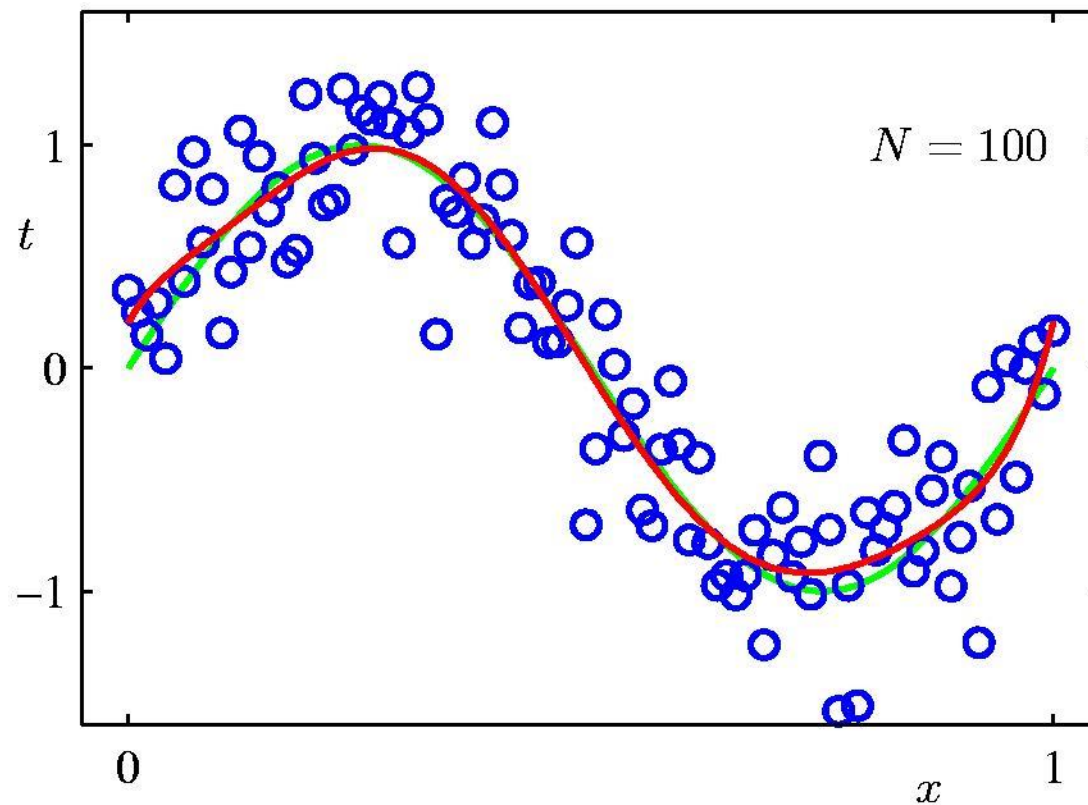
Data Set Size: $N=15$

9th Order Polynomial



Data Set Size: $N=100$

9th Order Polynomial



Feature Size vs Sample Size

- Richer feature set is a blessing if we have enough data points to make use of these features.
- When the sample size is limited, adding large amounts of features leads to overfitting.
- An overfitted model performs well for known data points.
- However, its prediction is usually bad for unseen data points.
- Overfitting is a situation that we should always try to avoid.



Regression: Probability-based Perspective

- We derived regression solution based on RSS minimization.
- An alternative, and sometimes more useful perspective, is to view the problem as a probability-based model
- Use maximum likelihood estimator (MLE) to estimate the model.
- For linear regression, RSS minimization and MLE gives the same result.
- But MLE and probability-based perspective are also useful in other extensions.
- Will spend some time on this.



3.1.1 Maximum Likelihood and Least Squares

- Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad \text{where} \quad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

- which is the same as saying,

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

- Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and targets, $\mathbf{t} = [t_1, \dots, t_N]^T$, we obtain the likelihood function

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

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x - \mu)^2 \right\}$$



Maximum Likelihood and Least Squares

- Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

- where

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- is the sum-of-squares error.

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x - \mu)^2 \right\}$$



Maximum Likelihood and Least Squares

- Computing the gradient and setting it to zero yields

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- $\frac{\partial \ln p(t|\mathbf{w}, \beta)}{\partial \mathbf{w}^T} = 0 = \frac{\partial \beta E_D(\mathbf{w})}{\partial \mathbf{w}^T}$
- $= \beta \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\} \boldsymbol{\phi}(\mathbf{x}_n)^T$
- $= \beta \sum_{n=1}^N \{t_n \boldsymbol{\phi}(\mathbf{x}_n)^T - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^T\}$
- Adopting vector notation
- $= \beta \{\mathbf{t}^T \Phi - \mathbf{w}^T \Phi^T \Phi\} = 0$

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix} \cdot \overset{N \times M}{=} \begin{pmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{pmatrix}$$



Maximum Likelihood and Least Squares

- From $\beta\{\mathbf{t}^T \Phi - \mathbf{w}^T \Phi^T \Phi\} = 0$
- $\Rightarrow \mathbf{t}^T \Phi - \mathbf{w}^T \Phi^T \Phi = 0$
- $\Rightarrow \mathbf{t}^T \Phi = \mathbf{w}^T \Phi^T \Phi \Rightarrow \Phi^T \mathbf{t} = \Phi^T \Phi \mathbf{w}$
- Solving for \mathbf{w} :

$$\mathbf{w}_{\text{ML}} = \left(\Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}$$

The Moore-Penrose pseudo-inverse, Φ^\dagger .

- where

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix} = \begin{pmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{pmatrix}$$



MLE for Precision (β)

- $\ln p(t|w, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi - \beta E_D(w)$
- $E_D(w) = \frac{1}{2} \sum_{n=1}^N \{t_n - w^T \phi(x_n)\}^2$
- $\frac{\partial \ln p(t|w, \beta)}{\partial \beta} = \frac{N}{2} \frac{1}{\beta} - E_D(w) = 0$
- $\frac{N}{\beta} = \sum_{n=1}^N \{t_n - w^T \phi(x_n)\}^2$
- $\frac{1}{\beta} = \frac{1}{N} \sum_{n=1}^N \{t_n - w^T \phi(x_n)\}^2$
- We can substitute w with \hat{w} , which gives us
- $\frac{1}{\hat{\beta}} = \frac{1}{N} \sum_{n=1}^N \{t_n - \hat{w}^T \phi(x_n)\}^2$
- Note that the variance ($1/\hat{\beta}$) computed this way is consistent but biased. An unbiased estimator for variance ($1/\beta$) is $\frac{1}{N-M} \sum_{n=1}^N \{t_n - \hat{w}^T \phi(x_n)\}^2$



3.1.4 Regularized Least Squares (1)

- Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

Avoid over-fitting on small training size

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

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

- which is minimized by

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

λ is called the regularization coefficient.



Regularized Least Squares (2)

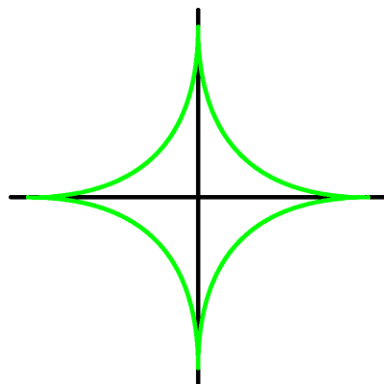
- With a more general regularizer, we have

Similar to constraint optimization

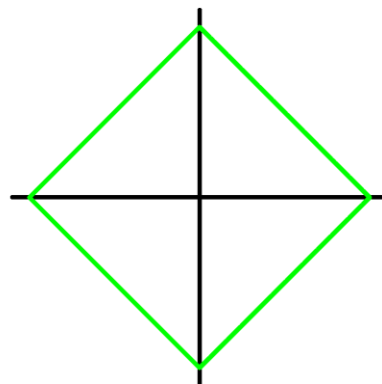
$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

$$f(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

subject to $\sum_{j=1}^M |w_j|^q \leq \eta$

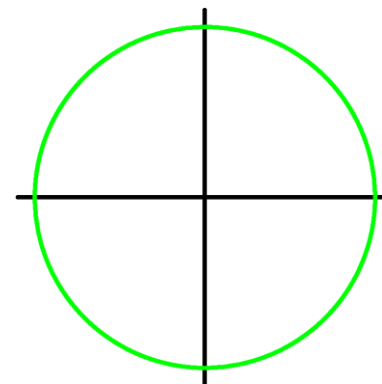


$q = 0.5$



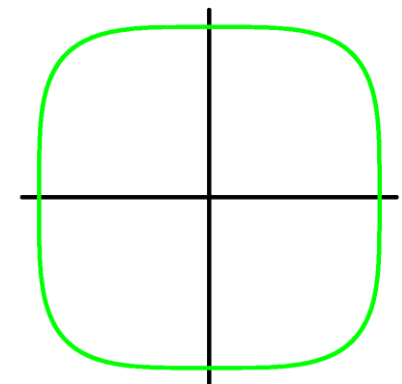
$q = 1$

Lasso



$q = 2$

Quadratic (Ridge
Regression)



$q = 4$



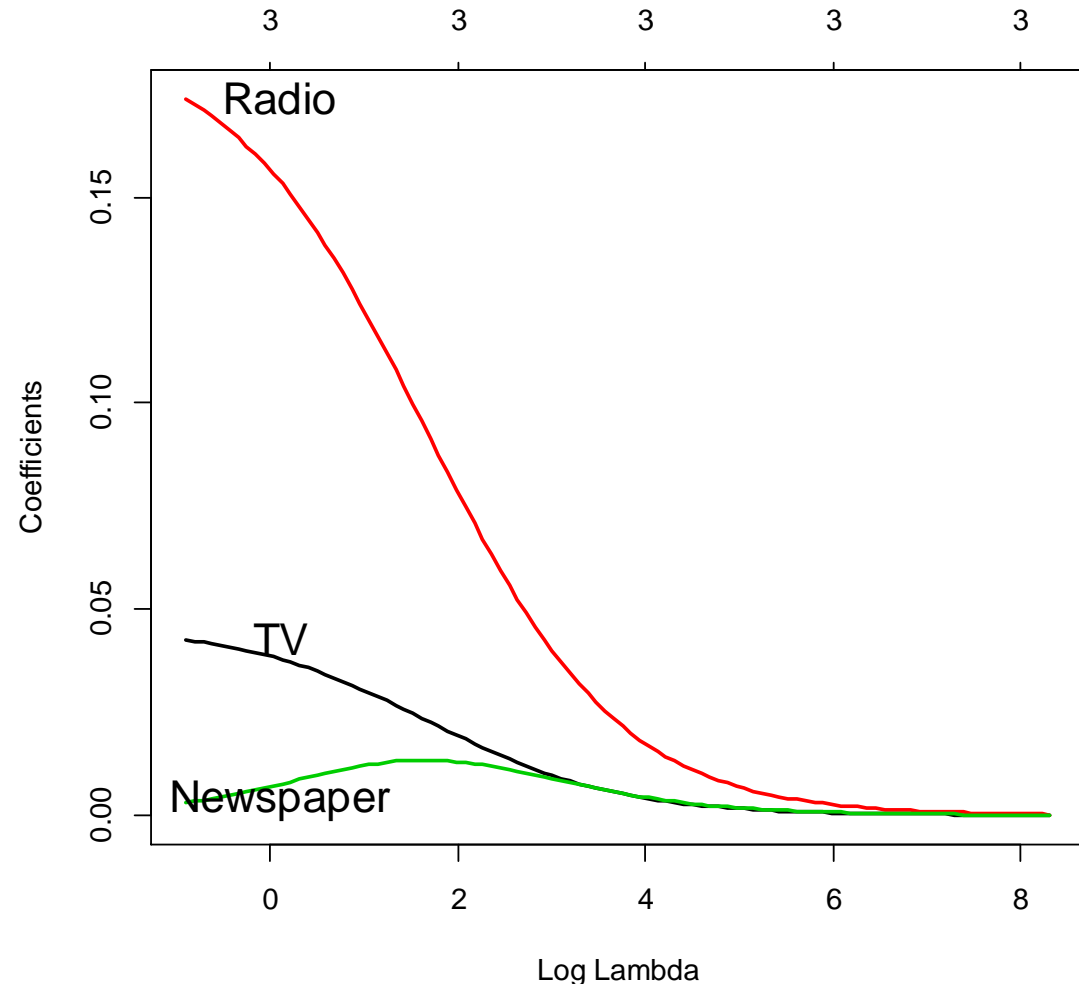
Ridge Regression Example

- Fitting data via minimizing $\frac{1}{2} \sum (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \frac{\lambda}{2} \sum w_j^2$
- Changing λ can gives different fitted \mathbf{w}
- Using glmnet package to demonstrate the relationship between λ and \mathbf{w} .
- `import glmnet_python`
- `from glmnet import glmnet`
- `from glmnetPlot import glmnetPlot`
- `df1 = pd.read_csv('Advertising.csv')`
- `allfeatures = ['TV', 'Newspaper', 'Radio']`
- `fit = glmnet(x = df1[allfeatures].as_matrix(),`
- `y = df1['Sales'].as_matrix(),`
- `alpha = 0, family = 'gaussian')`
- `glmnetPlot(fit, xvar = 'lambda', label = True)`



Ridge Regression Example

- As λ increases, all coefficients become smaller
- When λ is large enough, all coefficients are essentially 0.
- The relationship between λ and a coefficient is not always monotonous.
- We need to find a best λ when constructing a prediction model.



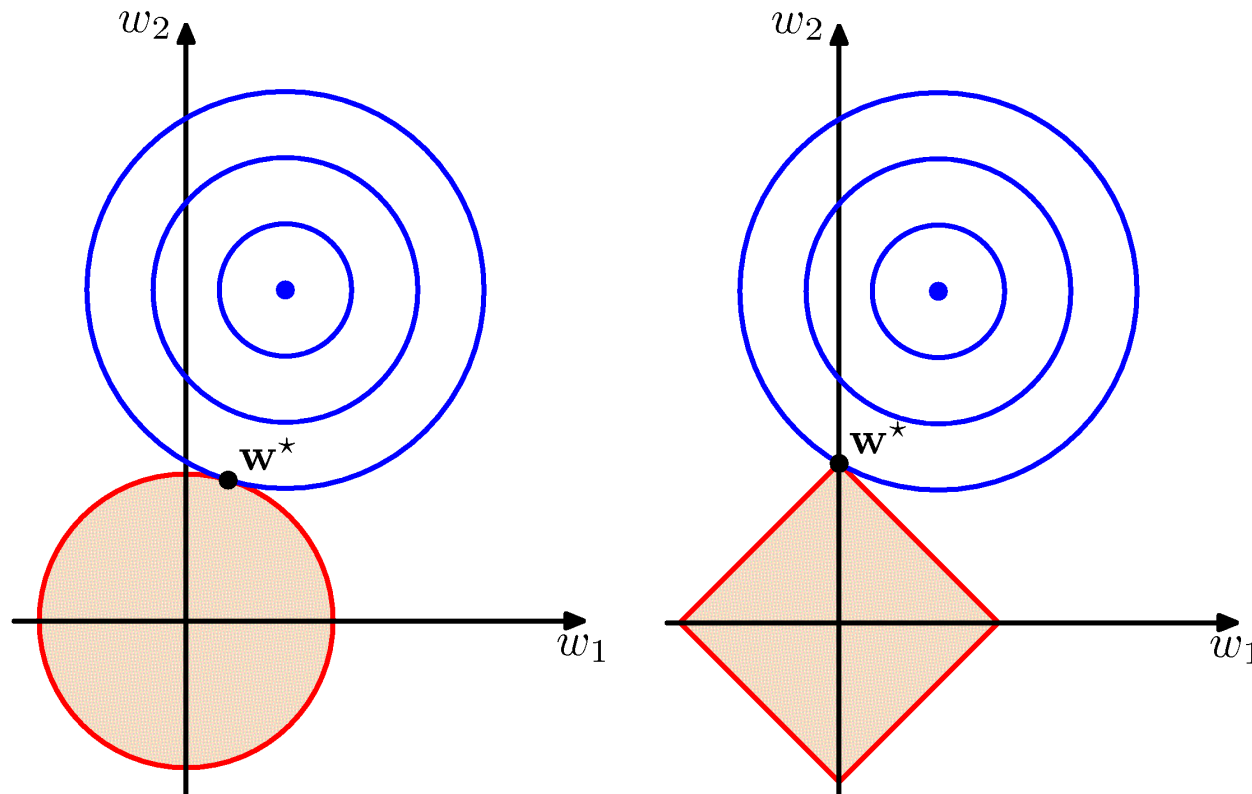
Note: glmnet_py currently only works for Linux (e.g. Ubuntu)



Regularized Least Squares (3)

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

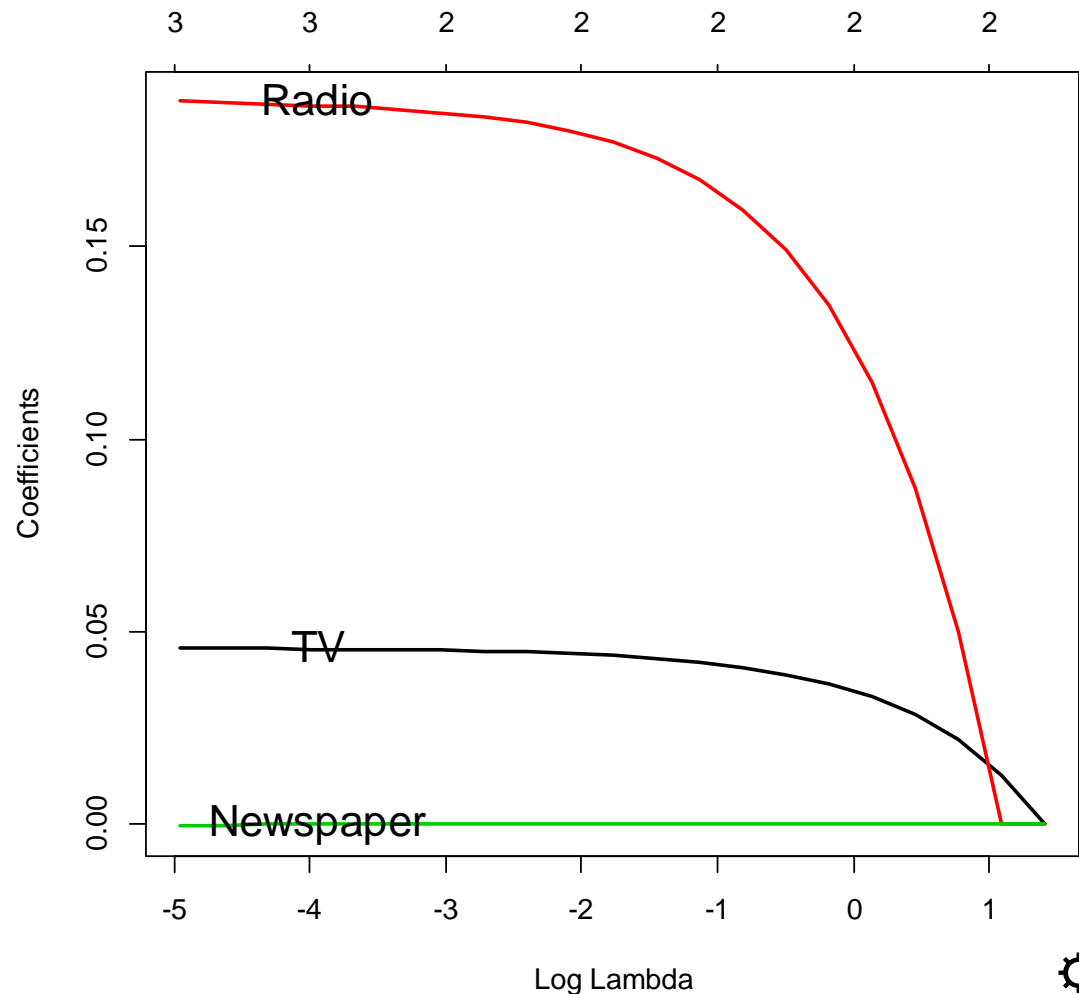
If λ is sufficiently large



Lasso Regression Example

Lasso regression loss: $\frac{1}{2} \sum (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \lambda \sum |w_j|$

- Newspaper was set to zero for all lambda.
- Radio becomes zero when lambda is large enough
- Can be used to select # of parameters included.



Model Training: Lasso Regression

- Lasso regression loss:

$$L = \frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \lambda \sum_{i=1}^M |w_i|$$

- We are trying to minimize L by choose the best \mathbf{w} .
- However, this problem has no closed-form solution.
- The estimation is based on the coordinate descent algorithm.
- That is, we loop through each w_i and solve for a better value that can improve upon current value.

Coordinate Descent and Soft Thresholding

$$L = \frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \lambda \sum_{i=1}^M |w_i|$$

- Assume that we start from $\tilde{\mathbf{w}}$ for \mathbf{w} .
- For each w_j , $j=1, 2, \dots, M$, we try to find a better value that can decrease L (fixing other w_i).
- We solve the problem by looking for a w_j that can give us zero gradient w.r.t. w_j .
- $\frac{\partial L}{\partial w_j} = \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))(-\phi(\mathbf{x}_n)_j) + \lambda \text{sgn}(w_j)$.
- Here $\text{sgn}(w_j) = 1$ if $w_j > 0$,
 $\text{sgn}(w_j) = -1$ if $w_j < 0$
 $\text{sgn}(w_j) = 0$ if $w_j = 0$

Coordinate Descent and Soft Thresholding (Cont'd.)

- To facilitate subsequent discussion, we write $\mathbf{w}^T \phi(\mathbf{x}_n)$ as
- $\mathbf{w}^T \phi(\mathbf{x}_n) = w_1 \phi(\mathbf{x}_n)_1 + w_2 \phi(\mathbf{x}_n)_2 + \cdots + w_M \phi(\mathbf{x}_n)_M = w_1 \phi_{n,1} + w_2 \phi_{n,2} + \cdots + w_M \phi_{n,M} = \mathbf{w}_{-j}^T \boldsymbol{\phi}_{n,-j} + w_j \phi_{n,j}$.
- Here $\mathbf{w}_{-j}^T = [w_1 \ w_2 \ \dots \ w_{j-1} \ w_{j+1} \ \dots \ w_M]$
- Similarly, we define $\boldsymbol{\phi}_{n,-j}$ as the vector that exclude the j-th element from $\boldsymbol{\phi}_n$

Coordinate Descent and Soft Thresholding (Cont'd.)

- Using the new notation,
- $\frac{\partial L}{\partial w_j} = \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n)) (-\phi(\mathbf{x}_n)_j) + \lambda \text{sgn}(w_j)$
- $= \sum_{n=1}^N (t_n - \mathbf{w}_{-j}^T \boldsymbol{\phi}_{n,-j} - w_j \phi_{n,j}) (-\phi_{n,j}) + \lambda \text{sgn}(w_j)$
- $= \sum_{n=1}^N (t_n - \mathbf{w}_{-j}^T \boldsymbol{\phi}_{n,-j}) (-\phi_{n,j}) + w_j \sum_{n=1}^N \phi_{n,j}^2 + \lambda \text{sgn}(w_j)$
- Since we are using $\tilde{\mathbf{w}}$ as the starting value, we are solving
- $\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) (-\phi_{n,j}) + w_j \sum_{n=1}^N \phi_{n,j}^2 + \lambda \text{sgn}(w_j) = 0$
- $\rightarrow w_j = \frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j} - \lambda \text{sgn}(w_j)}{\sum_{n=1}^N \phi_{n,j}^2}$

Coordinate Descent and Soft Thresholding (Cont'd.)

- That is, given \tilde{w} , we want to update w_j by

- $\rightarrow w_j = \frac{\sum_{n=1}^N (t_n - \tilde{w}_{-j}^T \phi_{n,-j}) \phi_{n,j} - \lambda \operatorname{sgn}(w_j)}{\sum_{n=1}^N \phi_{n,j}^2}$

- However, w_j is on both sides...

- If the resulting w_j is positive,

- that is if $\frac{\sum_{n=1}^N (t_n - \tilde{w}_{-j}^T \phi_{n,-j}) \phi_{n,j} - \lambda}{\sum_{n=1}^N \phi_{n,j}^2} > 0$,

- then $w_j = \frac{\sum_{n=1}^N (t_n - \tilde{w}_{-j}^T \phi_{n,-j}) \phi_{n,j} - \lambda}{\sum_{n=1}^N \phi_{n,j}^2}$

- Do similar things for negative w_j .

Coordinate Descent and Soft Thresholding (Cont'd.)

- To summarize:
- if $\frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j} - \lambda}{\sum_{n=1}^N \phi_{n,j}^2} > 0$,
- then $w_j = \frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j} - \lambda}{\sum_{n=1}^N \phi_{n,j}^2}$
- if $\frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j} + \lambda}{\sum_{n=1}^N \phi_{n,j}^2} < 0$,
- then $w_j = \frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j} + \lambda}{\sum_{n=1}^N \phi_{n,j}^2}$
- Otherwise, $w_j = 0$

Coordinate Descent and Soft Thresholding (Cont'd.)

- If we set $w_j^* = \frac{\sum_{n=1}^N (t_n - \tilde{\mathbf{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}) \phi_{n,j}}{\sum_{n=1}^N \phi_{n,j}^2}$, then

$$w_j = \begin{cases} w_j^* - \frac{\lambda_j}{\sum_{n=1}^N \phi_{n,j}^2} & \text{if } w_j^* - \frac{\lambda}{\sum_{n=1}^N \phi_{n,j}^2} > 0 \\ w_j^* + \frac{\lambda_j}{\sum_{n=1}^N \phi_{n,j}^2} & \text{if } w_j^* + \frac{\lambda}{\sum_{n=1}^N \phi_{n,j}^2} < 0 \\ 0 & \text{otherwise} \end{cases}$$

- In words: set w_j to 0 if w_j^* it is not too far from zero.
- This is called soft thresholding.

Lasso Training

- Start from an initial \tilde{w} , iteratively update each w_j using the soft thresholding equation until converge.

Hyper-parameter Tuning

- There are three variation of the models
- Ridge regression: $\frac{1}{2} \sum (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \frac{\lambda}{2} \sum w_j^2$
- Lasso regression: $\frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \lambda \sum_{i=1}^M |w_i|$
- Elastic-net regression (fixed α) :
$$\frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \lambda \left(\alpha \sum_{i=1}^M |w_i| + \frac{(1-\alpha)}{2} \sum w_j^2 \right).$$
- For each model, need to tune λ for the best prediction performance. You have several options:
 - 1. Brute force method: grid search λ
 - 2. Use the glmnet algorithms
 - 3. Use theoretical value implied by the evidence function (will not cover, for ridge regression only)