Machine learning - HT 2016
4. Basis Expansion, Regularization, Validation

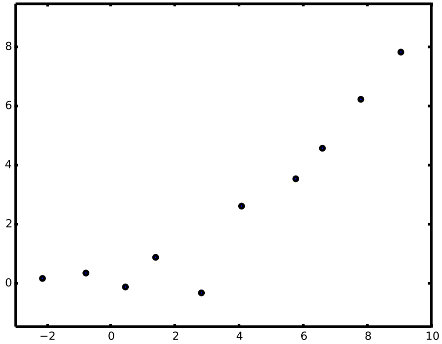
Varun Kanade

University of Oxford
Feburary 03, 2016

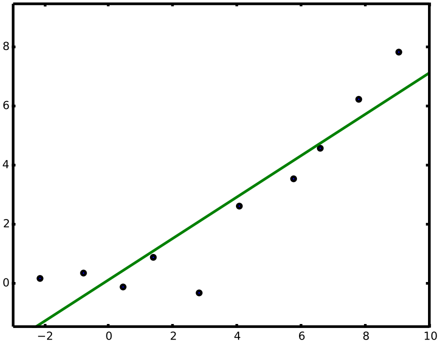
Outline

- Introduce basis function to go beyond linear regression
- Understanding the tradeoff between bias and variance
- Overfitting: What happens when we make models too complex
- Regularization as a means to control model complexity
- Cross-validation to perform model selection

Basis Expansion



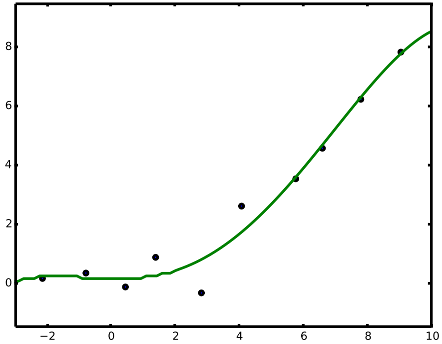
Basis Expansion



Basis Expansion

$$\boldsymbol{\phi}(x) = [1, x, x^2, x^3, x^4]$$

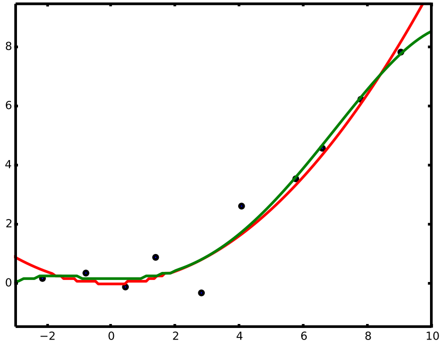
$$w_1 + w_2 x + w_3 x^2 + w_4 x_3 + w_4 x^5 = \boldsymbol{\phi}(x) \cdot [w_1, w_2, w_3, w_4, w_5]$$



Basis Expansion

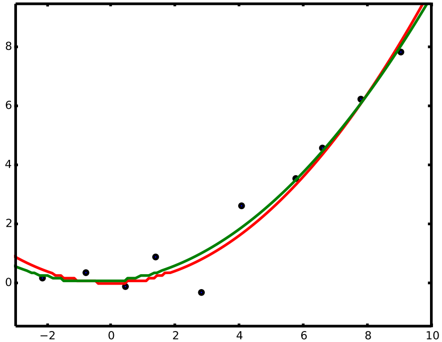
$$\phi(x) = [1, x, x^2, x^3, x^4]$$

$$w_1 + w_2 x + w_3 x^2 + w_4 x_3 + w_4 x^5 = \phi(x) \cdot [w_1, w_2, w_3, w_4, w_5]$$



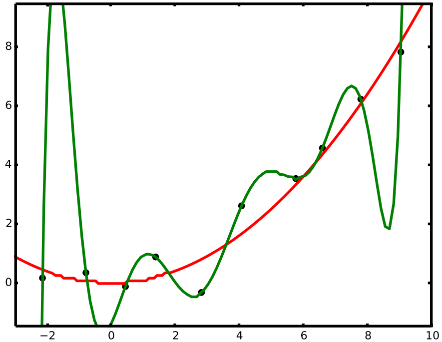
Basis Expansion

$$\phi(x) = [1, x, x^2]$$



Basis Expansion

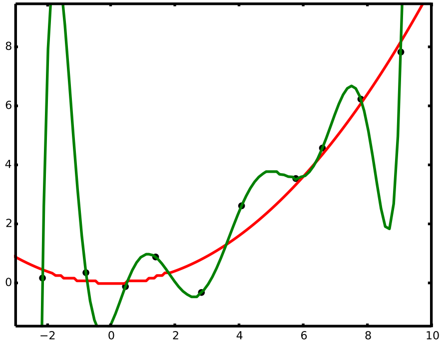
$$\phi(x) = [1, x, x^2, x^3, \ldots, x^9]$$



Basis Expansion

$$\phi(x) = [1, x, x^2, x^3, \ldots, x^d]$$

How can we avoid overfitting?

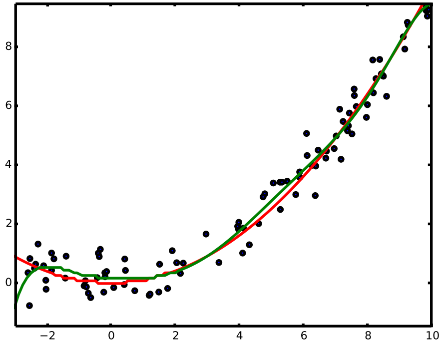


Basis Expansion

$$\phi(x) = [1, x, x^2, x^3, \ldots, x^d]$$

How can we avoid overfitting?

Does more data help?

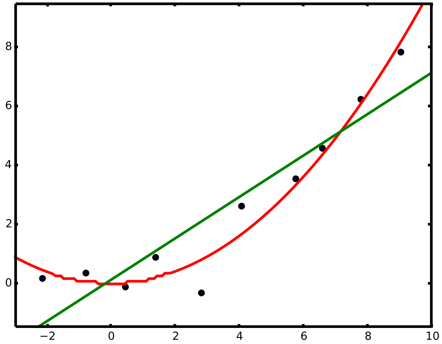


Basis Expansion

$$\phi(x) = [1, x, x^2, x^3, \ldots, x^d]$$

How can we avoid overfitting?

Does more data help?

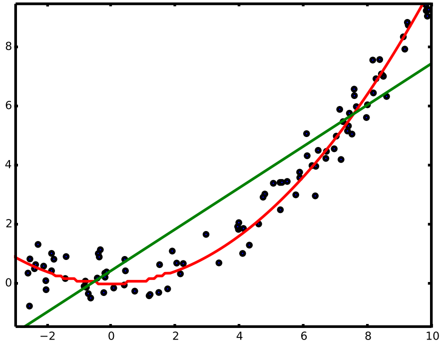


Basis Expansion

$$\phi(x) = [1, x, x^2, x^3, \ldots, x^d]$$

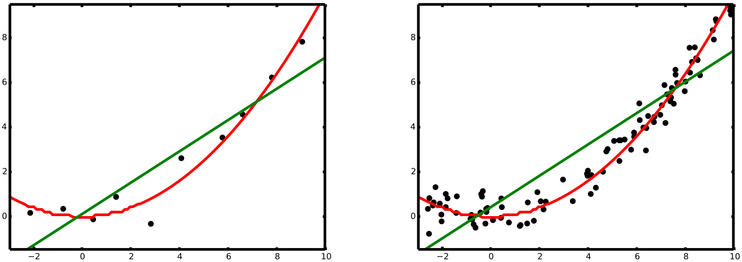
How can we avoid overfitting?

Does more data help?



Bias Variance Tradeoff

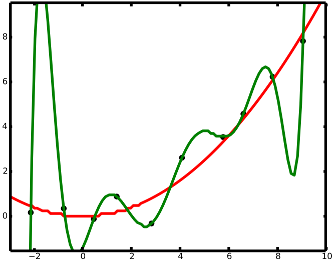
- For linear model, more data would make little difference

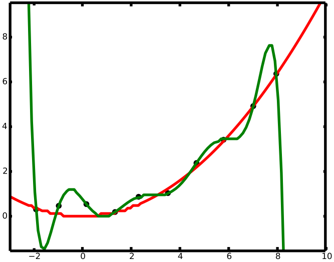


- Bias results from model being simpler than the "truth"
- High bias results in underfitting

Bias Variance Tradeoff

- What happens when we fit model on different (randomly drawn) training datasets?

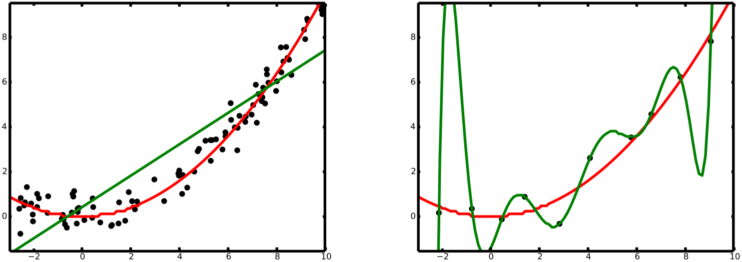




- Variance arises when the (complex) model is sensitive to fluctuations in training dataset
- Variance results in overfitting

Bias Variance Tradeoff

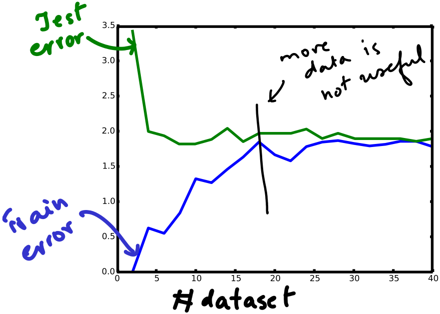
- When does more data help?
- Error = Bias^2 + Variance + Noise (Exercise for linear regression)

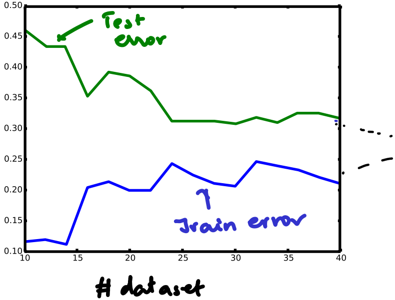


- For more complex models, difficult to visually overfitting and underfitting
- Keep aside some points as "test set"

Learning Curves

- Suppose we have a training set and test set
- Train on increasing sizes of the training set, and plot the errors





- Once training and test error approach each other, then more data won't help!

Basis Expansion Using Kernels

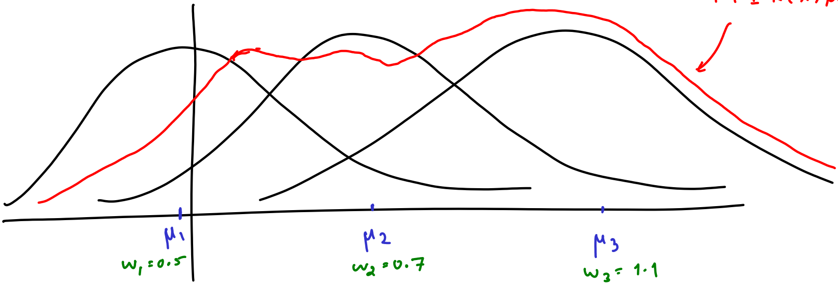
We can use kernels as features, *e.g.,* radial basis functions (RBFs)

Feature expansion:

$$\boldsymbol{\phi}(\mathbf{x}) = [1, \kappa(\mathbf{x}, \boldsymbol{\mu}_1, \sigma), \ldots, \kappa(\mathbf{x}, \boldsymbol{\mu}_d, \sigma)], \quad \textit{\textbf{e.g.,}}\ \kappa(\mathbf{x}, \boldsymbol{\mu}_i, \sigma) = e^{-\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma^2}}$$

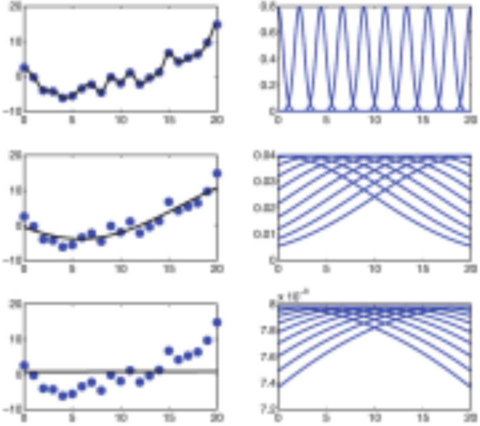
Model: $y = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{w} + \text{noise}$

$y = 0 + 0.5\,k(x, \mu_1, \sigma^2) + 0.7\,k(x, \mu_2, \sigma^2) + 1.1\,k(x, \mu_3, \sigma^2)$



Basis Expansion Using Kernels

As in the case of polynomials, the width σ can cause overfitting or underfitting



σ too small
overfitting

σ just right

σ too large
underfitting

Image Source: K. Murphy (2012)

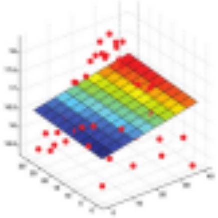
Polynomial Basis Expansion in Higher Dimensions

We are basically fitting linear models (at the cost of increasing dimensions)

$$y = \phi(\mathbf{x}) + \text{noise}$$

Linear Model: $\phi(\mathbf{x}) = [1, x_1, x_2]$

Quadratic Model: $\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$



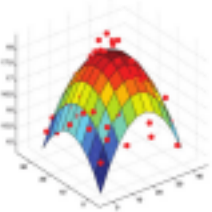


Image Source: K. Murphy (2012)

How many dimensions do you get for degree d polynomials over n variables? grows as n^d $\rightarrow$ if $n = 1000$, $d = 20$, this is 10^{60}!

Overfitting!

In high dimensions, we can have many many parameters!

With 100 variables and degree 10 we have $\sim 10^{20}$ parameters!

Enrico Fermi to Freeman Dyson

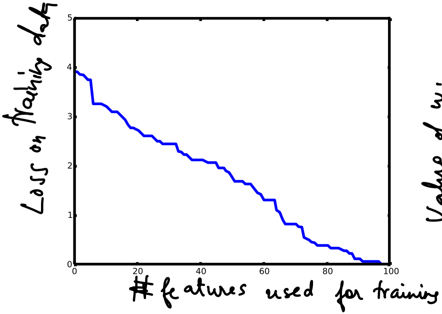
"I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk." [video]

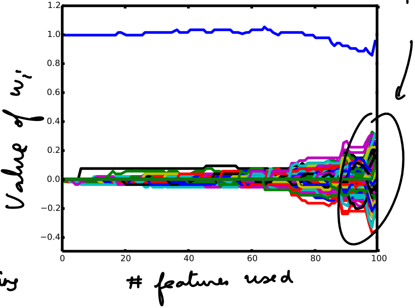
How do we prevent overfitting?

How does overfitting occur?

Suppose $\mathbf{X}_{100\times 100}$ with every entry $\mathcal{N}(0,1)$

And let $y_i = x_{i,1} + \mathcal{N}(0,\sigma^2)$, for $\sigma = 0.2$





All w_i except w_1 are fitting noise!!

Ridge Regression

Say our data is $\langle(\mathbf{x}_i, y_i)\rangle_{i=1}^m$, where $\mathbf{x} \in \mathbb{R}^N$ where N is really really large!

We used the squared loss

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

and obtained the estimate

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

NN - terminology
(Also in torch)

Suppose we want $\mathbf{w}$ to be "small" (weight decay)

$$L(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda\mathbf{w}^T\mathbf{w} - \lambda w_1^2$$

Don't regularize
the bias term.

We will not regularize the "bias" (or constant) term

<u>Exercise</u>: If all x_i (except x_1) have mean 0 and y has mean 0, $w_1 = 0$

Deriving Estimate for Ridge Regression

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

$$\nabla_w L = 2X^TXw - 2X^Ty + 2\lambda w = 0$$

$$\Rightarrow \quad w = (X^TX + \lambda I)^{-1} X^Ty$$

Positive semi-definite

$$\therefore \quad z^TX^TXz + \lambda z^Tz \geqslant 0 + \lambda > 0$$

$$\Rightarrow \nexists z \neq 0, \text{ s.t } (X^TX + \lambda I)z = 0$$

$$\Rightarrow X^TX + \lambda I \text{ is non-singular!}$$

Ridge Regression

Objective/Loss: $L(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T\mathbf{w}$

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

Estimate depends on the scaling of the inputs

Common practice: Normalise all input dimensions

Ridge penalizes all weights equally.

Good idea to make all features have 0 mean & variance 1.

Ridge Regression and MAP Estimation

Objective/Loss: $L(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T\mathbf{w}$

$$\exp\left(\frac{-1}{2\sigma^2} L(\mathbf{w})\right) = \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T \Sigma^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})\right) \cdot \exp\left(-\frac{1}{2}\mathbf{w}^T \Lambda^{-1}\mathbf{w}\right)$$

$$\Sigma = \begin{pmatrix} \sigma^2 & & & 0 \\ & \sigma^2 & & \\ & & \ddots & \\ 0 & & & \sigma^2 \end{pmatrix} \qquad \Lambda = \begin{pmatrix} \sigma^2/\lambda & & 0 \\ & \ddots & \\ 0 & & \sigma^2/\lambda \end{pmatrix}$$

$$= \underbrace{C}_{\text{normalization}} \cdot \underbrace{\mathcal{N}(y \mid Xw, \Sigma)}_{\text{likelihood of } y \mid Xw} \cdot \underbrace{\mathcal{N}(w \mid 0, \Lambda)}_{\text{prior over } w}$$

Bayesian View of Machine Learning

General Formulation

Prior $p(\mathbf{w})$ on $\mathbf{w}$

Model $p(y \mid \mathbf{x}, \mathbf{w})$

Linear Regression

$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \tau^2 \mathbf{I}_n)$

$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(y \mid \mathbf{x}^T \mathbf{w}, \sigma^2)$

Compute posterior given data $\mathcal{D} = \langle (\mathbf{x}_i, y_i) \rangle_{i=1}^m$

$$p(\mathbf{w} \mid \mathcal{D}) \propto \underbrace{p(\mathcal{D} \mid w)}_{\text{likelihood given model parameters}} \cdot \underbrace{p(w)}_{\text{prior}}$$

Bayes Rule:

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

The normalization $p(\mathcal{D}) = \int_w p(\mathcal{D} \mid w)\, p(w)$ can be difficult to compute in general. However, not needed for MAP estimate

Mode of posterior distribution.

Maximum A Posteriori (MAP) estimate

Ridge Regression $\equiv$ MAP estimation with Gaussian priors.

Bayesian View of Regression

Making a prediction on a new point $\mathbf{x}_{\text{new}}$

$\mathcal{D} = (X, \bar{y})$

$$p(y \mid \mathcal{D}, \mathbf{x}_{\text{new}}) = \int_{\mathbf{w}} p(y \mid \mathbf{w}, \mathbf{x}_{\text{new}}) p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w}$$

For linear regression

$$p(y \mid \mathcal{D}, x_{new}) = \int_{w} \mathcal{N}(y \mid x_{new}^T w, \sigma^2) \cdot \mathcal{N}(w \mid w_{MAP}, (X^T X + \lambda I)^{-1})$$

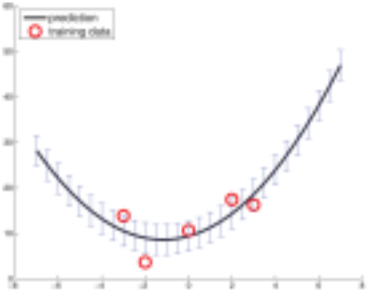
$$= \mathcal{N}\left(y \mid x_{new}^T w_{MAP}, \sigma^2 \left(1 + x_{new}^T \left(X^T X + \frac{\sigma^2}{\tau^2} I\right)^{-1} x_{new}\right)\right)$$

(For details see Murphy Sec. 7.6.1 & 7.6.2)

Prediction MAP vs Fully Bayesian

MAP Approach

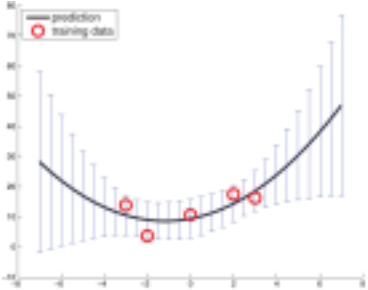
$$y_{\text{new}} \sim \mathcal{N}(\mathbf{x}_{\text{new}}^T \mathbf{w}_{\text{map}}, \sigma^2)$$



Bayesian Approach

$$y_{\text{new}} \sim \mathcal{N}(\mathbf{x}_{\text{new}}^T \mathbf{w}_{\text{map}}, \sigma_X^2)$$

$$\sigma_X^2 = \sigma^2 (1 + \mathbf{x}_{\text{new}}^T (\mathbf{X}^T \mathbf{X} + \tfrac{\sigma^2}{\tau^2} \mathbf{I})^{-1} \mathbf{x}_{\text{new}})$$



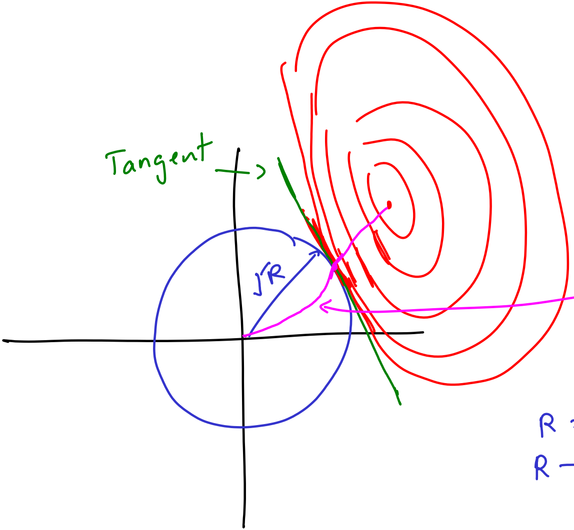
Ridge Regression

Minimize:

$$(\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})^T+\lambda\mathbf{w}^T\mathbf{w}$$

Minimize $(\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})$

such that $\mathbf{w}^T\mathbf{w}\leq R$

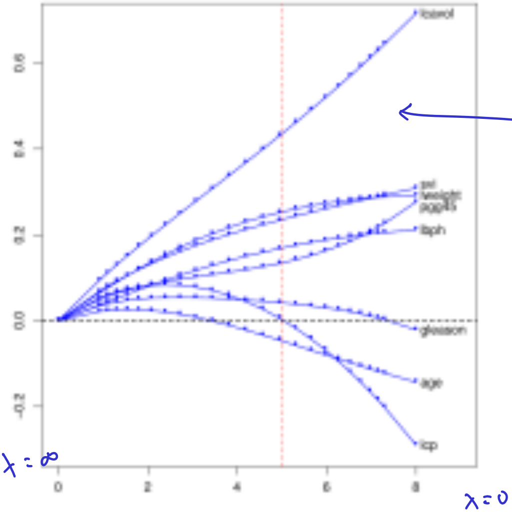


As you vary R from 0 to ∞ solution moves

$R = 0 \equiv \lambda \to \infty$

$R \to \infty \equiv \lambda = 0$

Ridge Regression



weights for different features

Image Source: Hastie, Tibshirani, Friedman (2013)

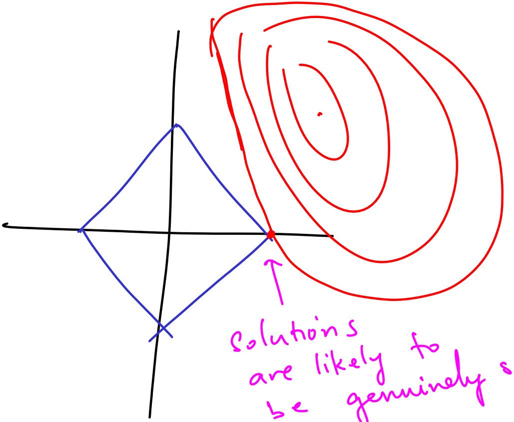
Lasso Regression

Minimize:

$$(\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})^T + \lambda \sum_{i=1}^{N} |w_i|$$

Minimize $(\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})$

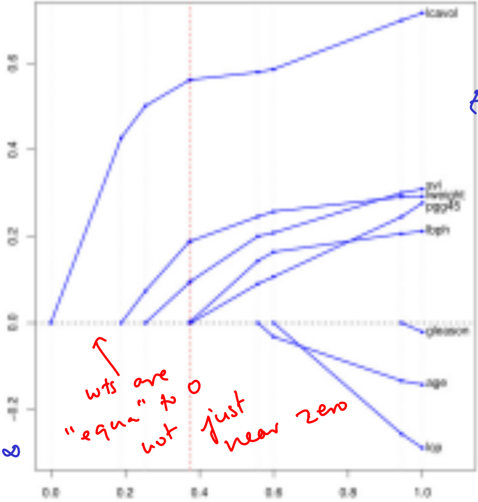
such that $\sum_{i=1}^{N} |w_i| \le R$



solutions are likely to be genuinely sparse if using L_1-regularization

No closed form solution. Needs convex optimization methods

Lasso Regression



← wts for features.

wts are "equal" to 0 not just near zero

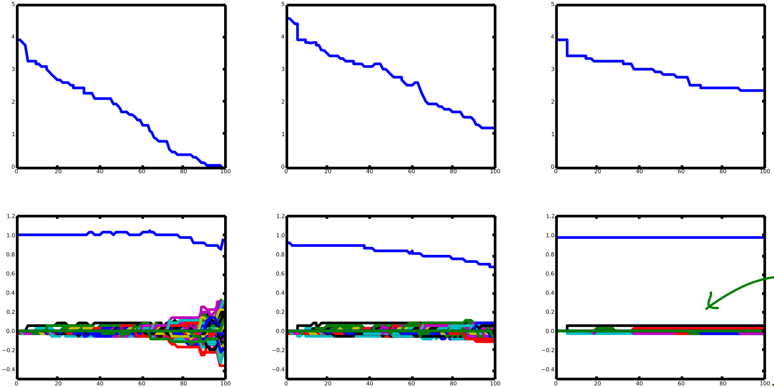
$\lambda = \infty$

$\lambda = 0$

Image Source: Hastie, Tibshirani, Friedman (2013)

Regularization: Ridge or Lasso (Example from slide 12)

No regularization | Ridge | Lasso



Very few non zero features in Lasso.

Choosing Parameters

Before we were just trying to find $\mathbf{w}$

Now, we have to worry about how to choose λ for ridge or Lasso

If we use kernels, we also have to pick the width σ

If we use higher degree polynomials, we have to pick d

Cross Validation

- Keep a part of data as "validation" or "test" set
- Look for error on "training" and "validation" sets

λ	Train Error(%)	Test Error(%)
0.01	0	89
0.1	0	43
1	2	12
10	10	8
100	25	27

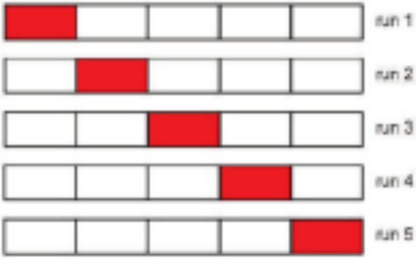
least error
on test
dataset

If you have small datasets
you may want to average
training & test error.
or choose the min max{train error, test error},

k-Fold Cross Validation

What do we do when data is scarce?

- Divide data into k parts
- Use $k - 1$ parts for training and 1 part as validation
- When $k = m$ (the number of datapoints), we get LOOCV (Leave one out cross validation)



Overfitting on the Validation Set

Suppose you do all the right things

- Train on the training set
- Choose hyperparameters using proper validation
- Test on the test set, and your error is high!

What would you do?

Winning Kaggle without reading the data!

Suppose the task is to predict m binary labels

Algorithm (Wacky Boosting):

1. Choose $\mathbf{y}^1, \ldots, \mathbf{y}^k \in \{0,1\}^m$ uniformly
2. Set $I = \{i \mid \text{accuracy}(\mathbf{y}^i) > 51\%\}$
3. Output $\hat{y}_j = \text{majority}\{y_j^i \mid i \in I\}$

Source: blog.mrtz.org