DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 24 2019

Logistics

- HW 1 is due on Friday 01/25
- Project proposal: due Feb 21
  - 1 page description of problem you will solve, dataset, and ML algorithms
  - Individual project
  - Project template and potential ideas are on Piazza
- Project milestone: due March 21
  - 2 page description on progress
- Project report at the end of semester and project presentations in class (10 minute per project)

Outline

- Gradient Descent comparison with closed-form solution
- Non-linear regression
- Regularization
  - Ridge and Lasso regression
  - Lab example
- Classification
  - K Nearest Neighbors (kNN)
  - Cross-validation

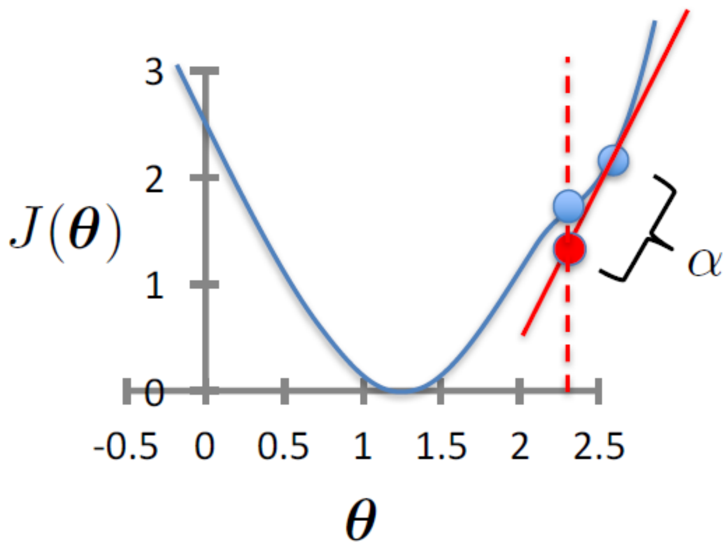
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



- Gradient = slope of line tangent to curve
- Function decreases faster in negative direction of gradient
- Larger learning rate => larger step

GD for Linear Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$||\theta_{new} - \theta_{old}|| < \epsilon$ or
iterations == MAX_ITER

$$\theta_j \leftarrow \theta_j - \alpha \frac{2}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

simultaneous
update
for j = 0 ... d

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right)$
  - Use this stored value in the update step loop

- Assume convergence when $\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$

L_2 norm: $$\|\boldsymbol{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$$

Can also bound number of iterations

Gradient Descent vs Closed Form

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update for j = 0 ... d

Closed form

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

Gradient Descent	Closed Form Solution
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$ is roughly $O(d^3)$

Issues with Gradient Descent

- Might get stuck in local optimum and not converge to global optimum
  - Restart from multiple initial points
- Only works with differentiable loss functions
- Small or large gradients
  - Feature scaling helps
- Tune learning rate
  - Can use line search for determining optimal learning rate

Beyond Linearity

- Most datasets are not perfectly linear
- Linear Regression results in high MSE
- Generally,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

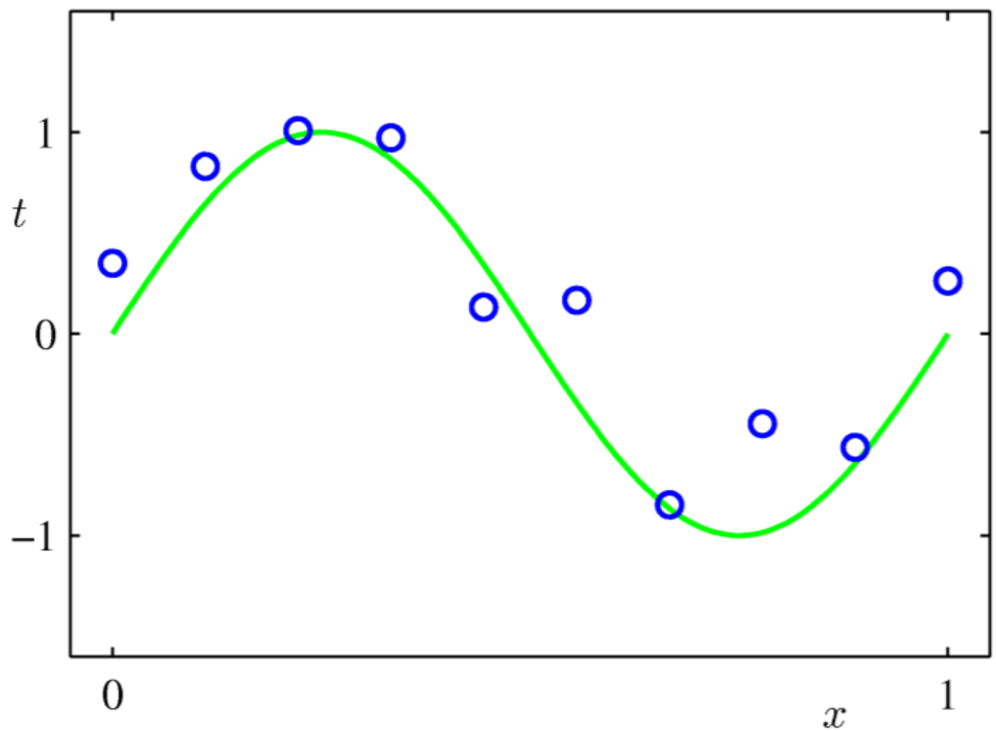
- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

$$\phi_j(\boldsymbol{x}) = x_j$$

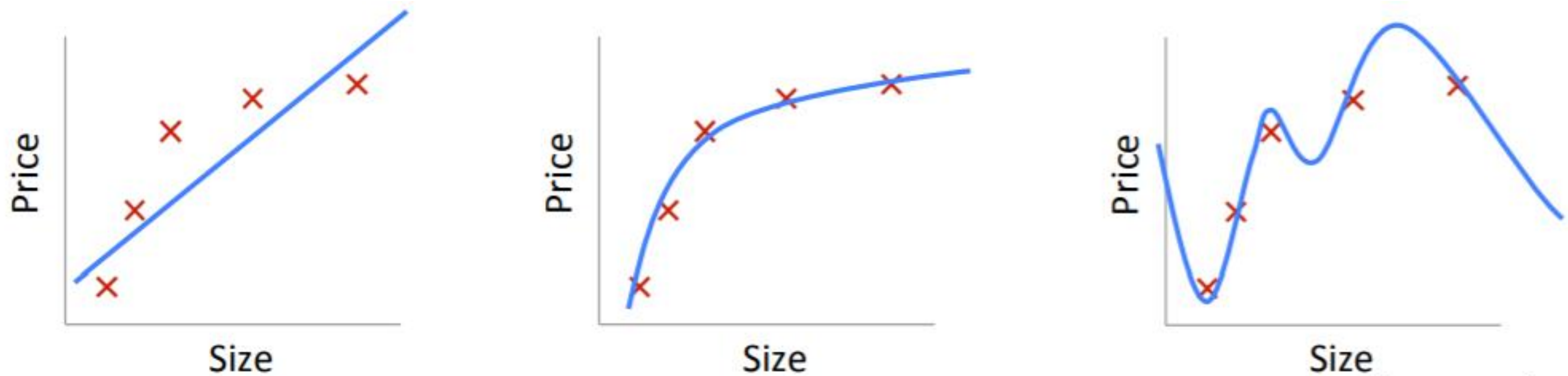
Generalized Additive Models

Polynomial Regression

- Polynomial basis function
  - $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_d x^d$



Polynomial Regression



$\theta_0 + \theta_1 x$

Underfitting (high bias)

$\theta_0 + \theta_1 x + \theta_2 x^2$

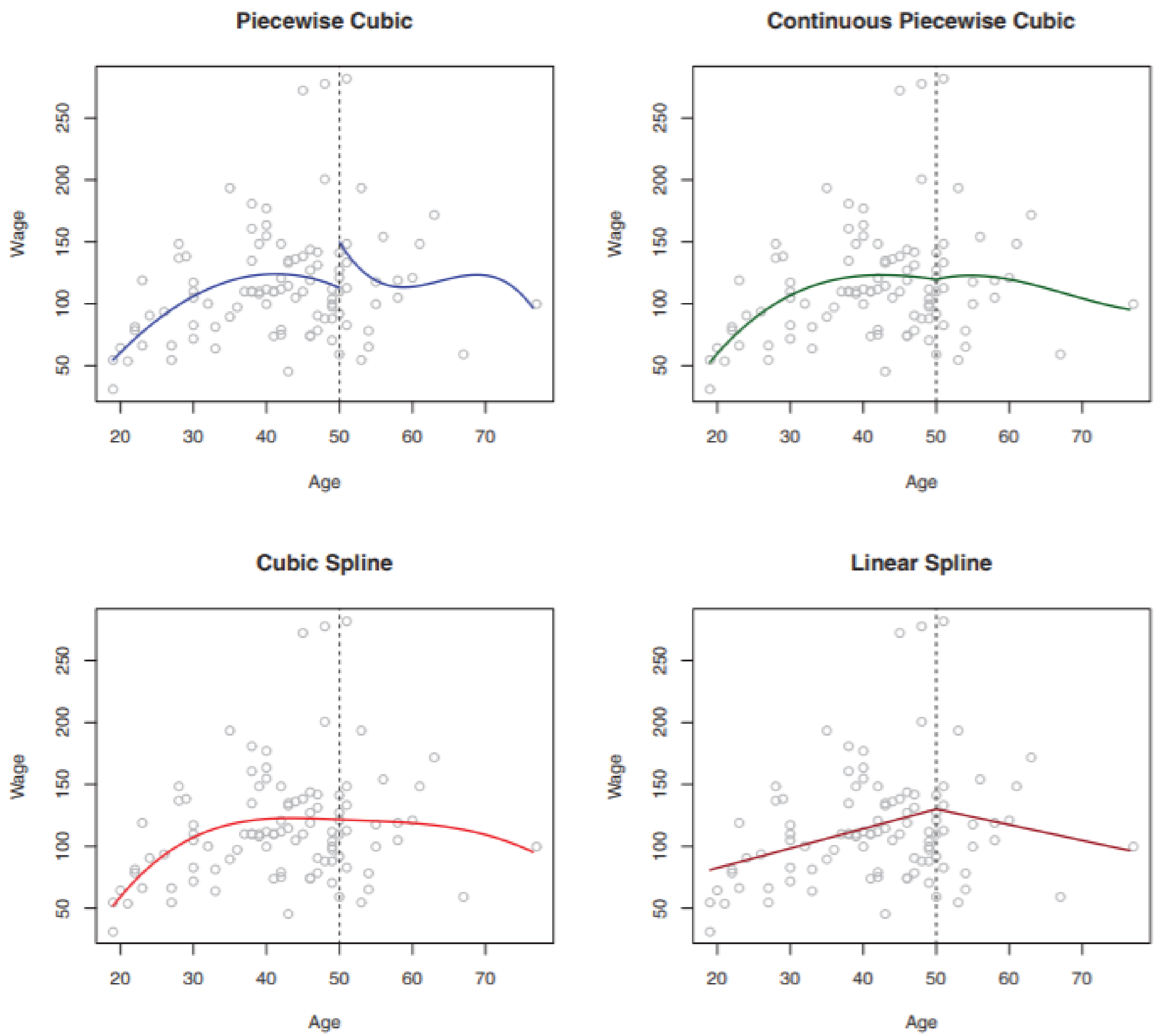
Correct fit

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

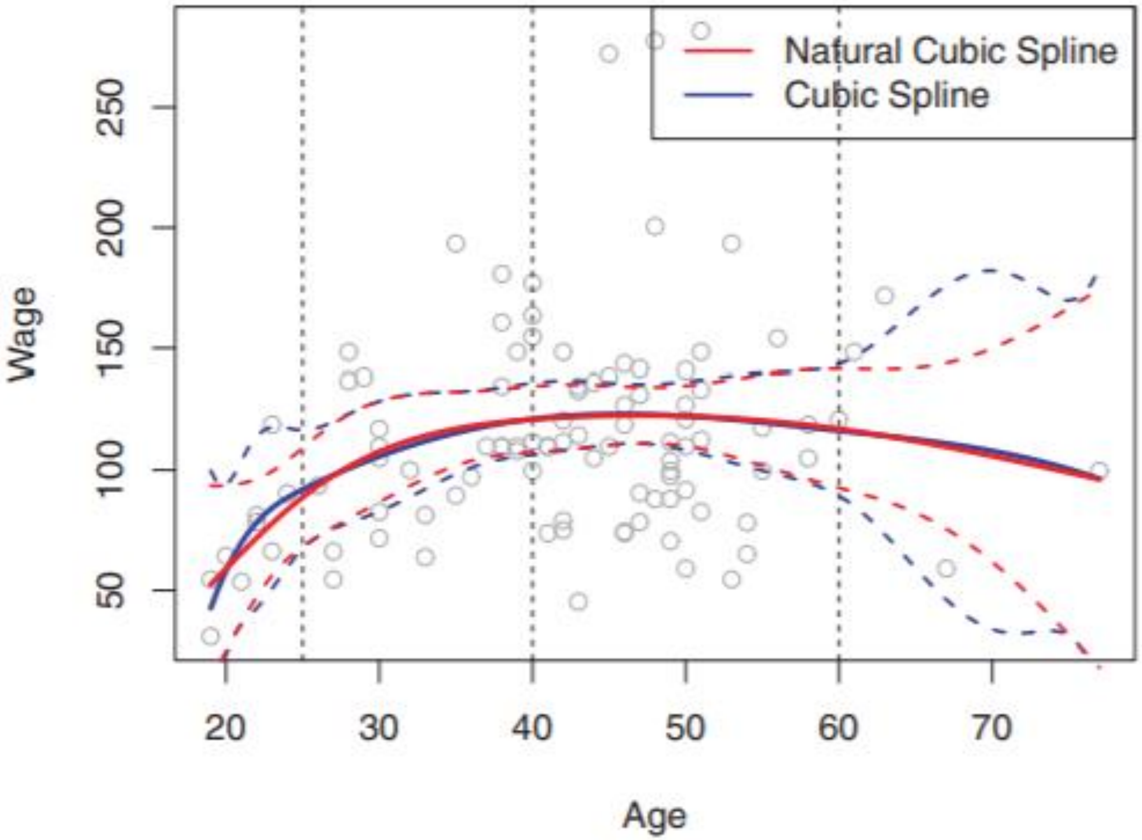
Overfitting (high variance)

- Typically to avoid overfitting $d \leq 4$

Other Regression

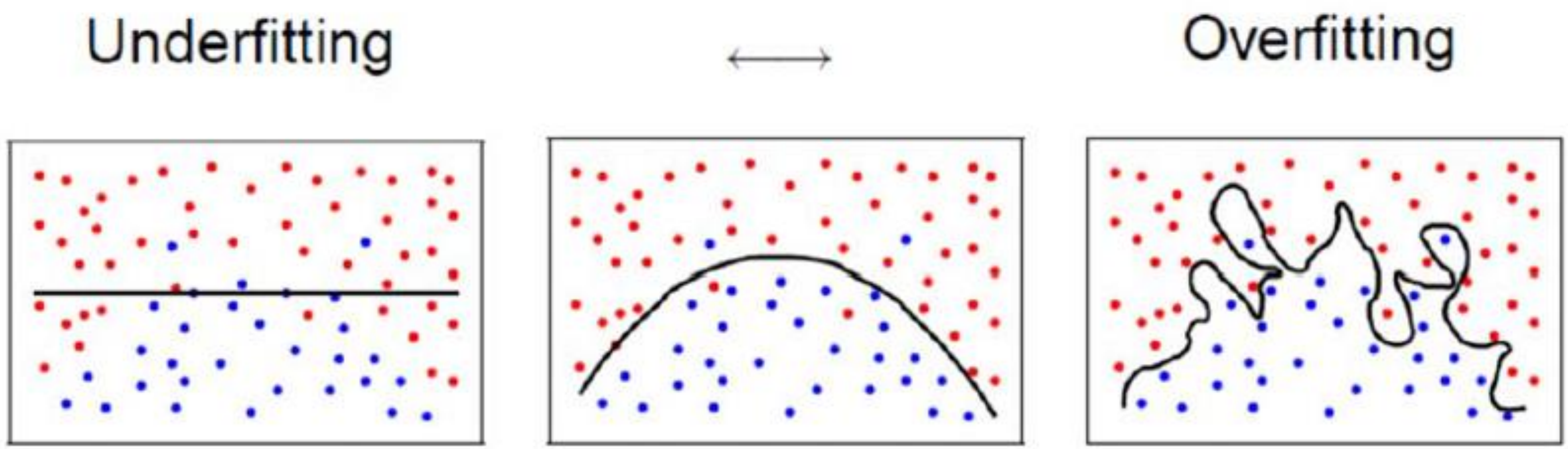


Splines



- Fit polynomial regression on each region (knot)
- Spline - Continuous and differentiable function at boundary
- Natural Spline – linear function at boundary

Generalization in ML



Simple model

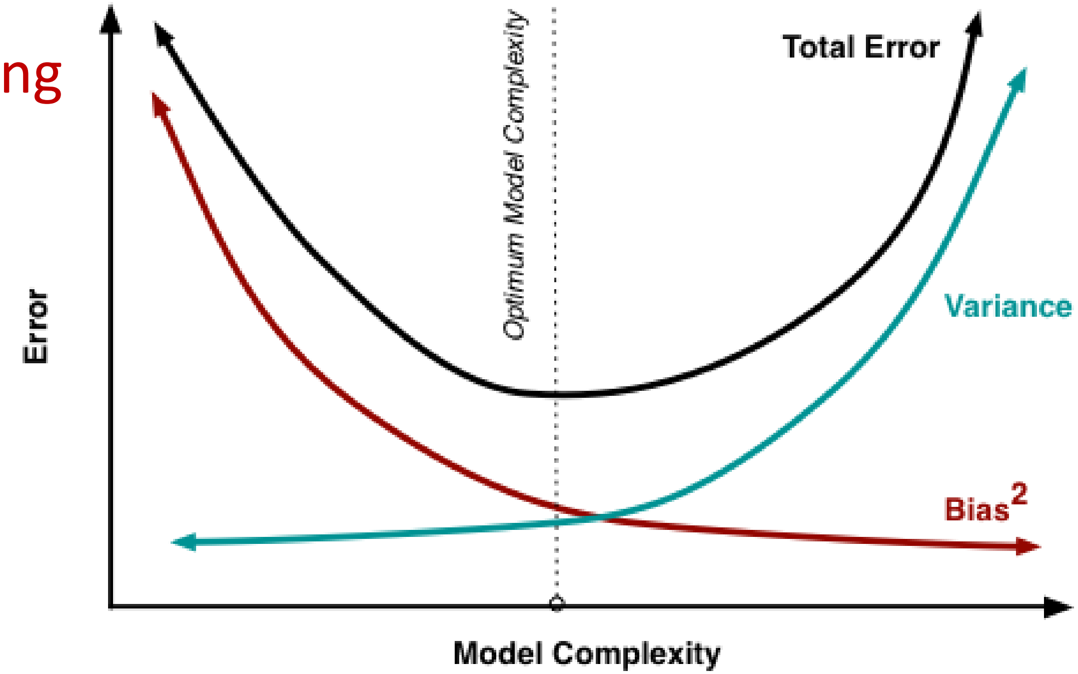
Complex model

- Goal is to generalize well on new testing data
- Risk of overfitting to training data
  - MSE close to 0, but performs poorly on test data

Bias-Variance Tradeoff

Under-fitting

Over-fitting



- Bias = Difference between estimated and true models
- Variance = Model difference on different training sets

MSE is proportional to Bias + Variance

Bias-Variance Decomposition

- Let $\hat{f}$ be trained model
- Expected MSE of test point (x_o, y_0):

  $E\left[(y_0 - \hat{f}(x_0))^2\right]$
- Variance: $Var\left[\hat{f}(x_0)\right] = E\left[\hat{f}(x_0)^2\right] - E^2\left[\hat{f}(x_0)\right]$
  - Variance of prediction over training data
- Bias: $Bias\left[\hat{f}(x_0)\right] = E\left[\hat{f}(x_0)\right] - y_0$
  - Bias of prediction over training data
- Verify that:
  - $MSE(x_o, y_0) = Var\left[\hat{f}(x_0)\right] + Bias^2[\hat{f}(x_0)]$

Regularization

- A method for automatically controlling the complexity of the learned hypothesis

- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label

- Can also address overfitting by eliminating features (either manually or via model selection)

Reduce model complexity
Reduce model variance

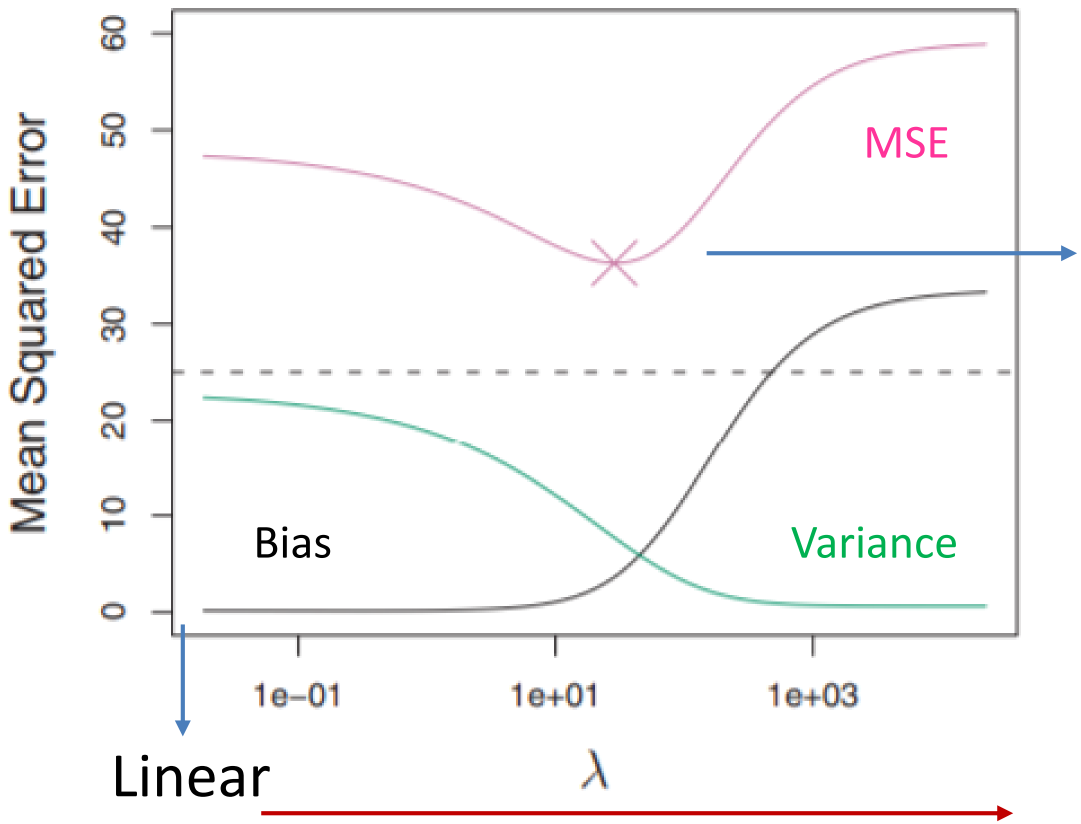
Ridge regression

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2}_{\text{model fit to data}} + \underbrace{\frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2}_{\text{regularization}}$$

  - λ is the regularization parameter ($\lambda \geq 0$)
  - No regularization on θ_0!
    - If λ = 0, we train linear regression
    - If λ is large, the coefficients will shrink close to 0

Bias-Variance Tradeoff



Optimal
Ridge regression

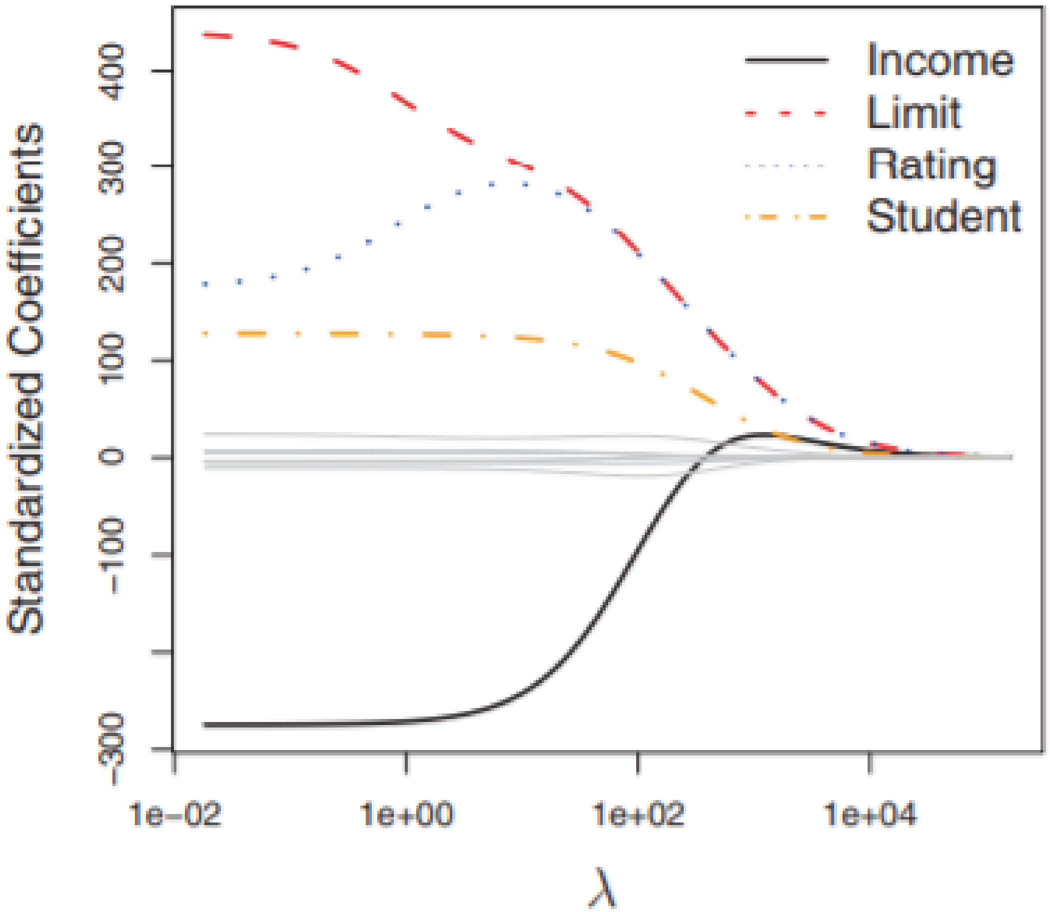
Linear
regression

Reduced model
complexity

Ridge performs better when linear regression has high variance

- Example: d (dimension) is close to n (training set size)

Coefficient shrinkage



Predict credit card balance

GD for Ridge Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

GD for Ridge Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} J(\theta) \qquad \theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \qquad \theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \underbrace{-\alpha \lambda \theta_j}_{\text{regularization}}$$

$$\theta_j \leftarrow \theta_j (1 - \alpha\lambda) - \alpha \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

Lasso regression

$$J(\theta) = \underbrace{\sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2}_{\text{Squared Residuals}} + \lambda \underbrace{\sum_{j=1}^{d} |\theta_j|}_{\text{Regularization}}$$

- L1 norm for regularization
- Cannot compute gradients
- Algorithms based on quadratic programming or other optimization techniques

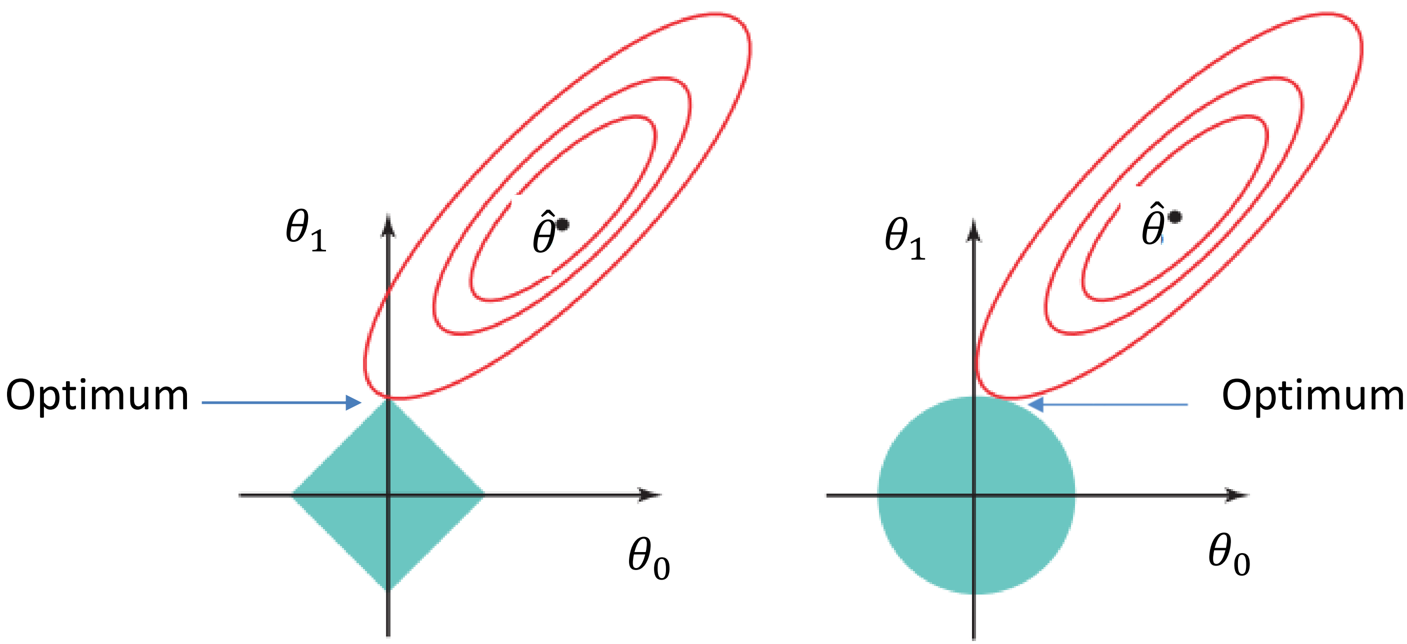
Alternative Formulations

- Ridge
  - L2 Regularization
  - $\min_{\theta} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ subject to $\sum_{j=1}^{d} \left|\theta_j\right|^2 \leq \epsilon$

- Lasso
  - L1 regularization
  - $\min_{\theta} \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ subject to $\sum_{j=1}^{d} \left|\theta_j\right| \leq \epsilon$

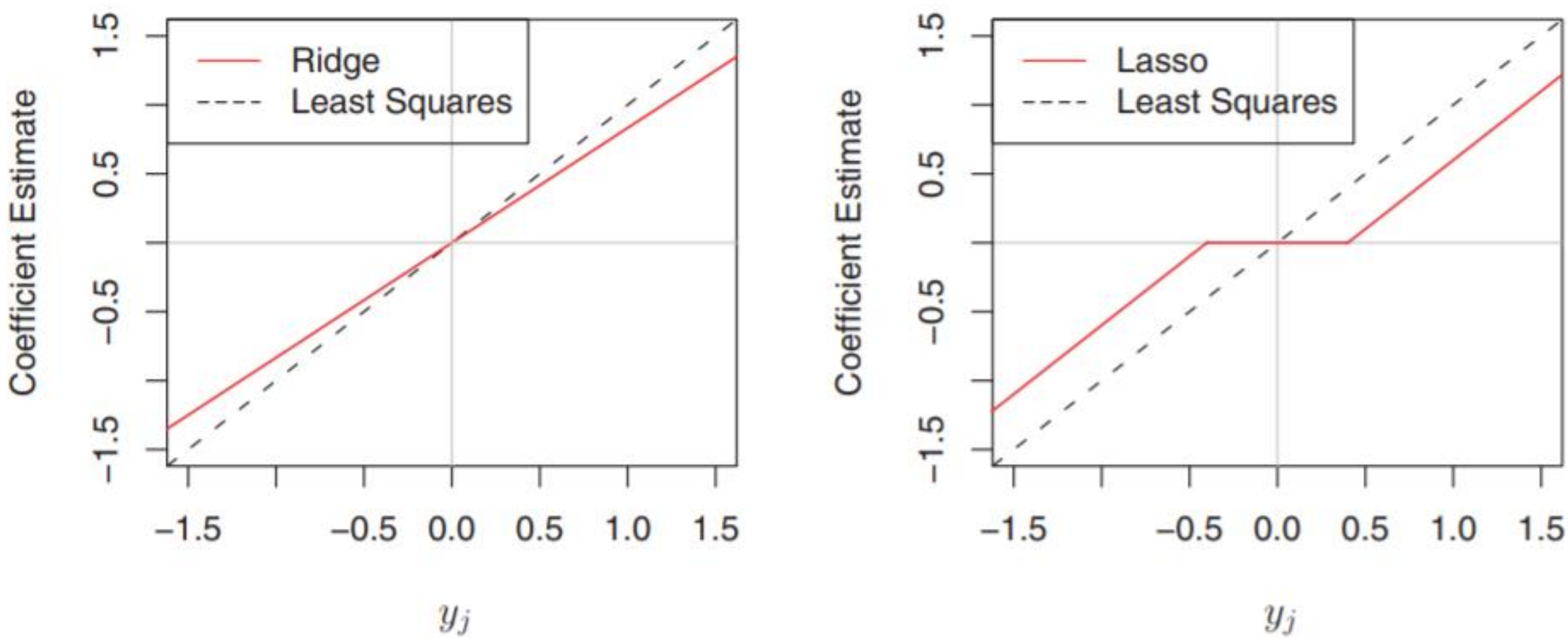
Lasso vs Ridge

- Ridge shrinks all coefficients
- Lasso sets some coefficients at 0 (sparse solution)
  - Perform feature selection



Lasso Ridge

Lasso vs Ridge



Lab example

```
> library(ISLR)
> fix(Hitters)
```

Data Editor

	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary
1	14	3449	835	69	321	414	375	N	W	632	43	10	475
2	3	1624	457	63	224	266	263	A	W	880	82	14	480
3	11	5628	1575	225	828	838	354	N	E	200	11	3	500
4	2	396	101	12	48	46	33	N	E	805	40	4	91.5
5	11	4408	1133	19	501	336	194	A	W	282	421	25	750
6	2	214	42	1	30	9	24	N	E	76	127	7	70
7	3	509	108	0	41	37	12	A	W	121	283	9	100
8	2	341	86	6	32	34	8	N	W	143	290	19	75
9	13	5206	1332	253	784	890	866	A	E	0	0	0	1100
10	10	4631	1300	90	702	504	488	A	E	238	445	22	517.143
11	9	1876	467	15	192	186	161	N	W	304	45	11	512.5
12	4	1512	392	41	205	204	203	N	E	211	11	7	550
13	6	1941	510	4	309	103	207	A	E	121	151	6	700

Ridge regression

```
> Hitters=na.omit(Hitters)
> x=model.matrix(Salary~.,Hitters)[,-1]
> y=Hitters$Salary
> ridge.mod=glmnet(x,y,alpha=0,lambda=5000)
> coef(ridge.mod)
20 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept) 305.016480230
AtBat         0.065738413
Hits          0.255494042
HmRun         0.902148872
Runs          0.419912564
RBI           0.428768355
Walks         0.533942922
Years         1.892781352
CAtBat        0.005532745
CHits         0.020876841
CHmRun        0.156069996
CRuns         0.041877748
CRBI          0.043262917
CWalks        0.043634641
LeagueN       1.117728148
DivisionW   -13.063063667
PutOuts       0.033021805
Assists       0.004993208
Errors       -0.061932828
NewLeagueN    1.269197088
> sqrt(sum(coef(ridge.mod)[-1]^2))
[1] 13.36602
```

Data processing (omit N/A)

Fit ridge regression

Coefficient values

Coefficient norm

Ridge regression

Fit ridge regression

```
> ridge.mod=glmnet(x,y,alpha=0,lambda=50)
> coef(ridge.mod)
20 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  4.800582e+01
AtBat       -3.532997e-01
Hits         1.950804e+00
HmRun       -1.286413e+00
Runs         1.158693e+00
RBI          8.114814e-01
Walks        2.709241e+00
Years       -6.179435e+00
CAtBat       6.262426e-03
CHits        1.072029e-01
CHmRun       6.284707e-01
CRuns        2.155421e-01
CRBI         2.148524e-01
CWalks      -1.483366e-01
LeagueN      4.585236e+01
DivisionW   -1.182395e+02
PutOuts      2.501361e-01
Assists      1.206414e-01
Errors      -3.277654e+00
NewLeagueN  -9.424451e+00
> sqrt(sum(coef(ridge.mod)[-1]^2))
[1] 127.4217
```

Coefficient values

Coefficient norm

λ controls parameter size

Lasso regression

```
> lasso.mod=glmnet(x,y,alpha=1,lambda=50)
> coef(lasso.mod)
20 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)   88.6306382
AtBat          .
Hits           1.5877156
HmRun          .
Runs           .
RBI            .
Walks          1.8197051
Years          .
CAtBat         .
CHits          .
CHmRun         .
CRuns          0.1711419
CRBI           0.3709268
CWalks         .
LeagueN        .
DivisionW    -43.3646551
PutOuts        0.1341253
Assists        .
Errors         .
NewLeagueN     .
> sqrt(sum(coef(lasso.mod)[-1]^2))
[1] 43.43398
```

Fit Lasso regression

13 coefficients set at zero

Coefficient norm

Outline

- Gradient Descent comparison with closed-form solution
- Non-linear regression
- Regularization
  - Ridge and Lasso regression
  - Lab example
- Classification
  - K Nearest Neighbors (kNN)
  - Cross-validation

Supervised learning

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$

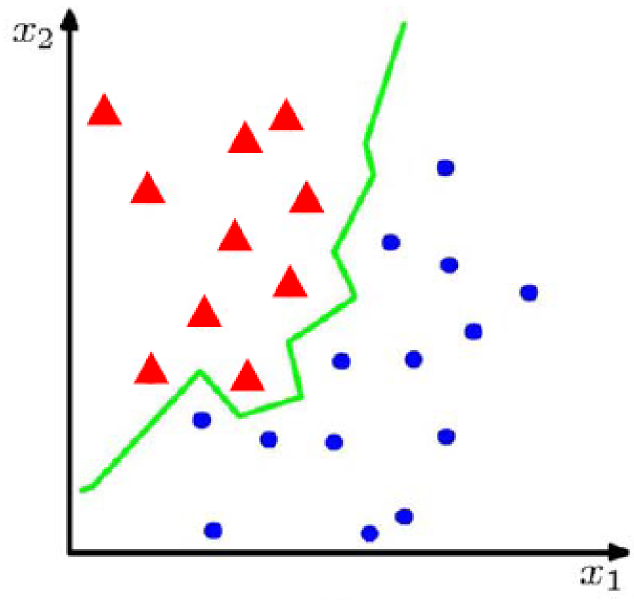
Input: Training examples of unknown target function f

$$\{x^{(i)}, y^{(i)}\}, \text{for } i = 1, \dots, n$$

Output: Hypothesis $\hat{f} \in H$ that best approximates f

$$\hat{f}\left(x^{(i)}\right) \approx y^{(i)}$$

Classification



Binary or discrete

- Suppose we are given a training set of N observations

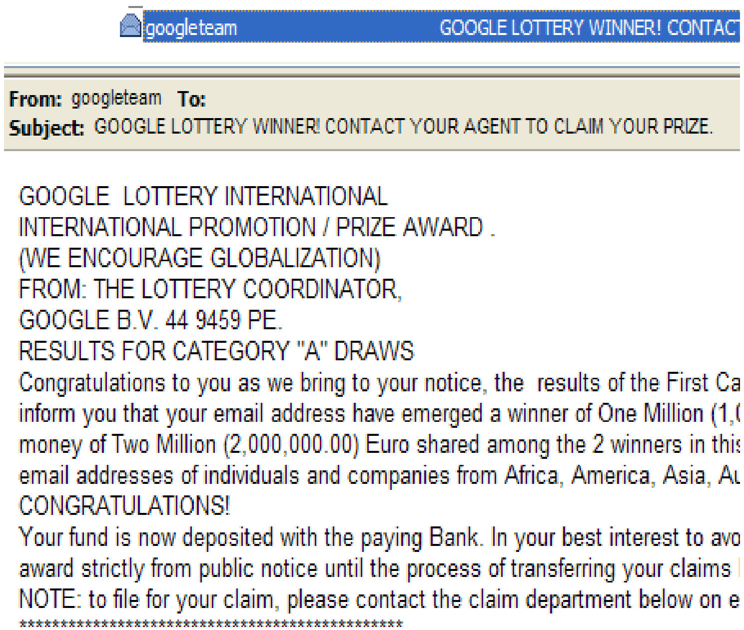
$$\{x^{(1)}, \dots, x^{(n)}\} \text{ and } \{y^{(1)}, \dots, y^{(n)}\}, x^{(i)} \in R^d, y^{(i)} \in \{-1, 1\}$$

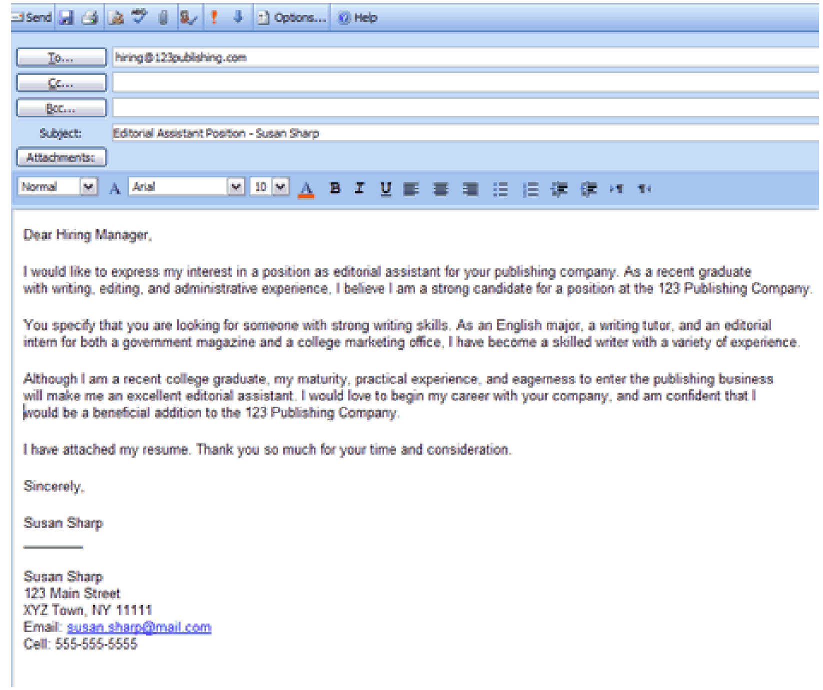
- Classification problem is to estimate f(x) from this data such that

$$f\left(x^{(i)}\right) = y^{(i)}$$

Example 1

Classifying spam email





Content-related features

- Use of certain words
- Word frequencies
- Language
- Sentence

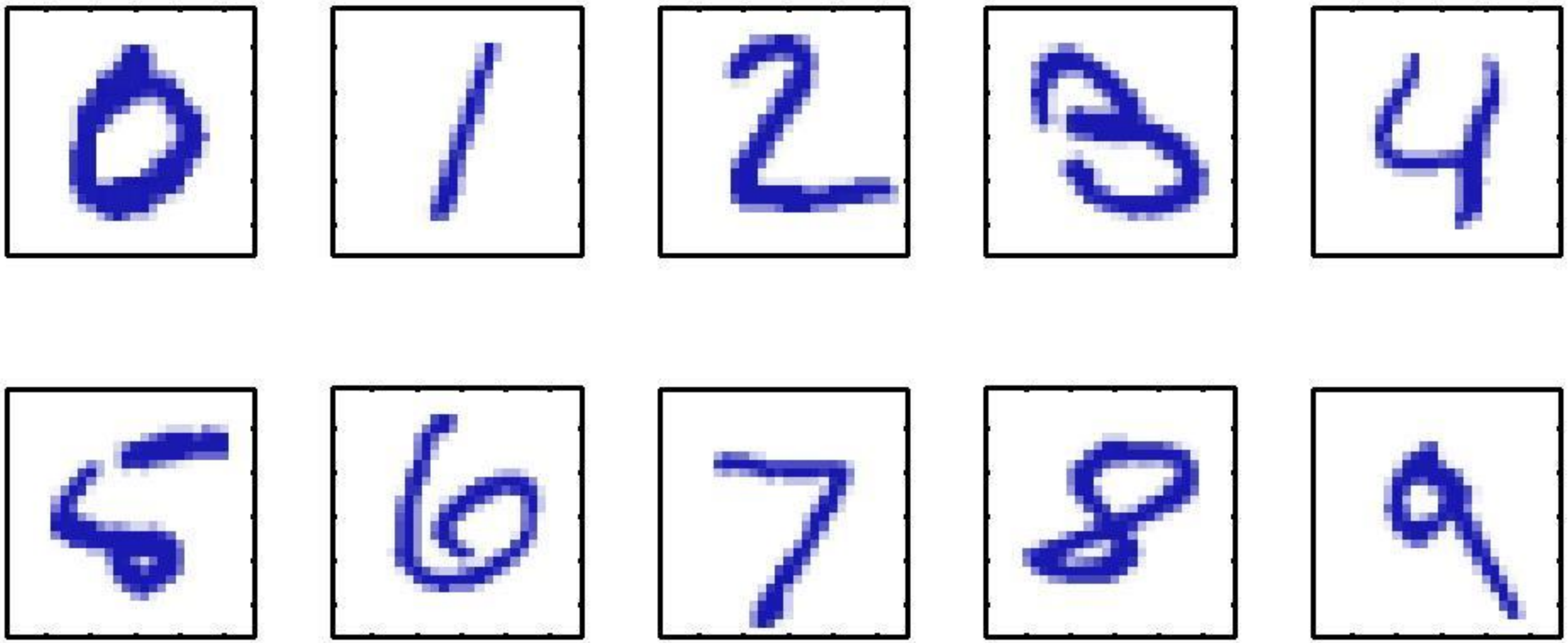
Structural features

- Sender IP address
- IP blacklist
- DNS information
- Email server
- URL links (non-matching)

Binary classification: SPAM or HAM

Example 2

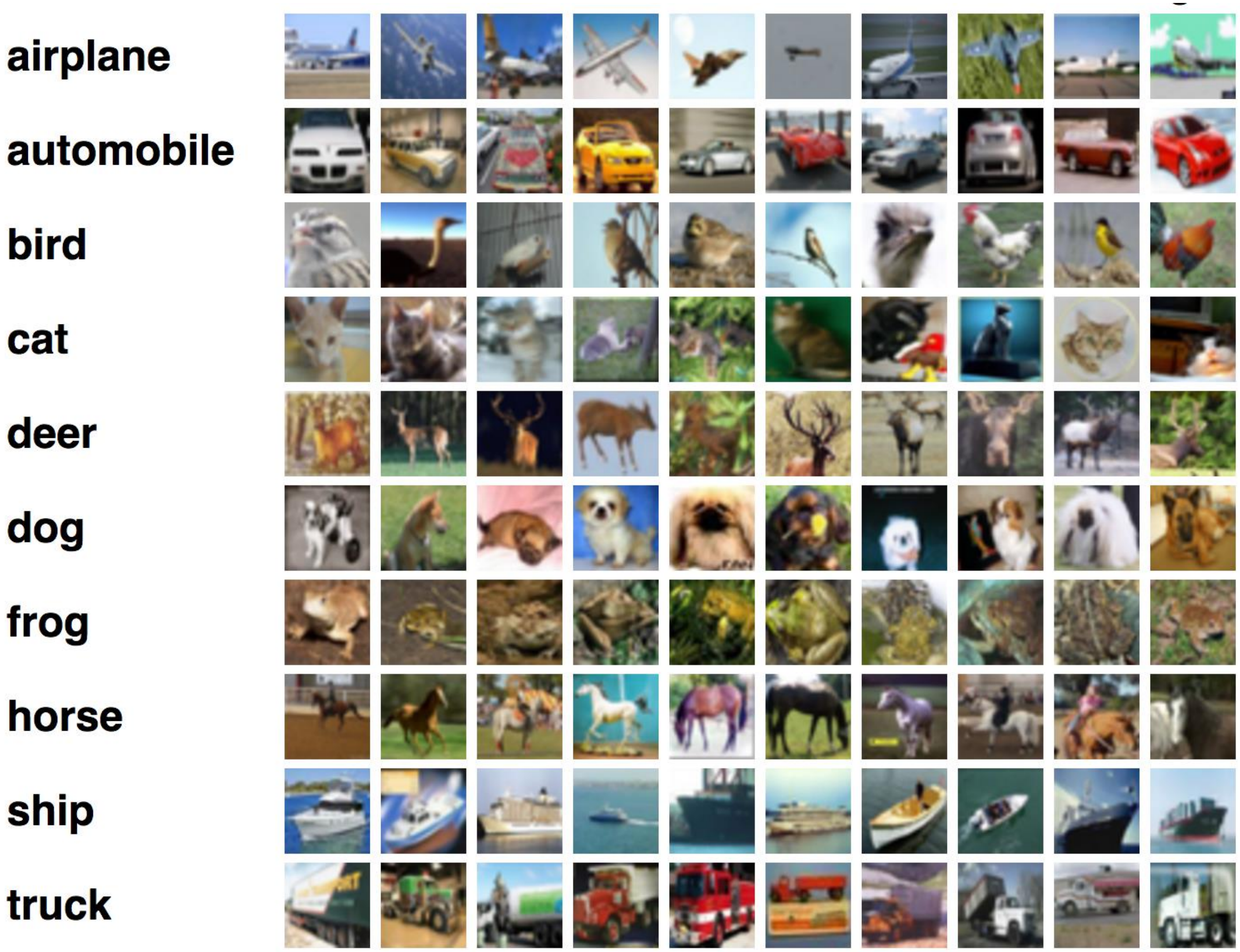
Handwritten Digit Recognition



Multi-class classification

Example 3

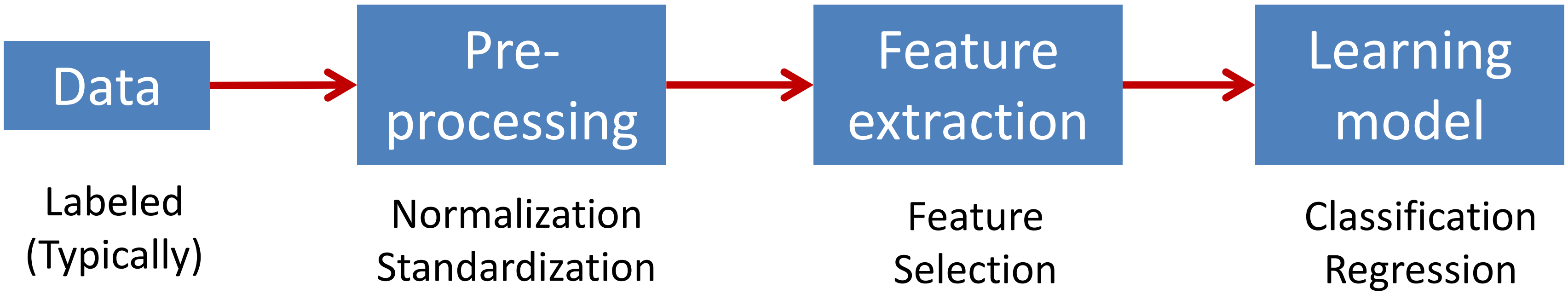
Image classification



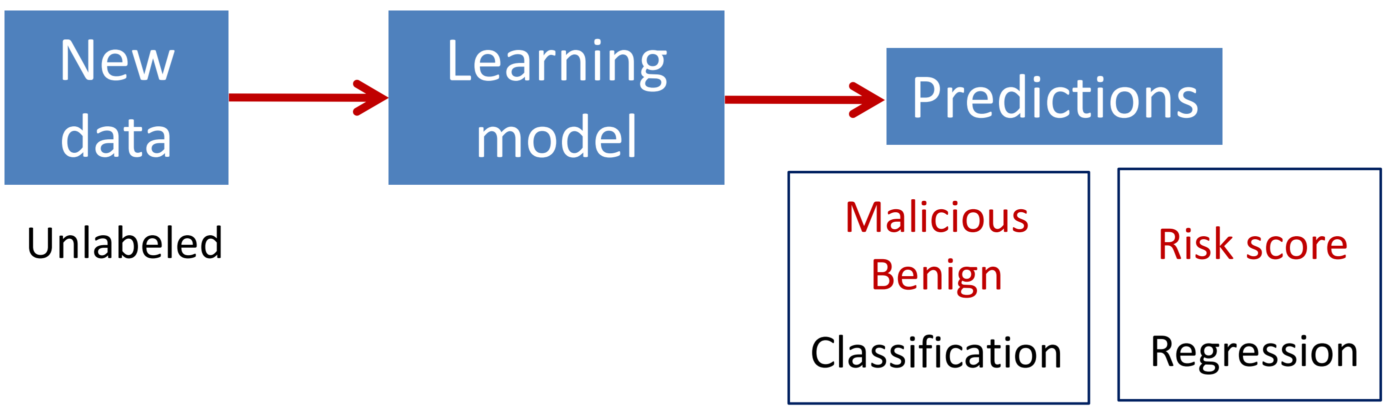
Multi-class classification

Supervised Learning Process

Training



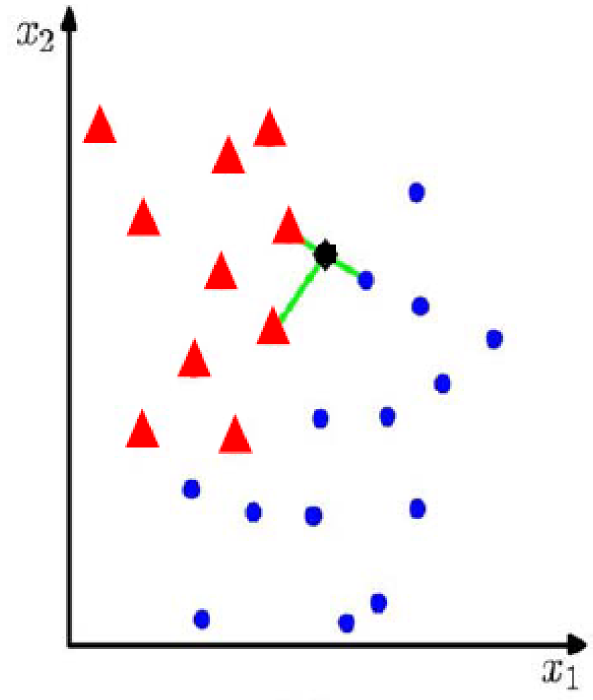
Testing



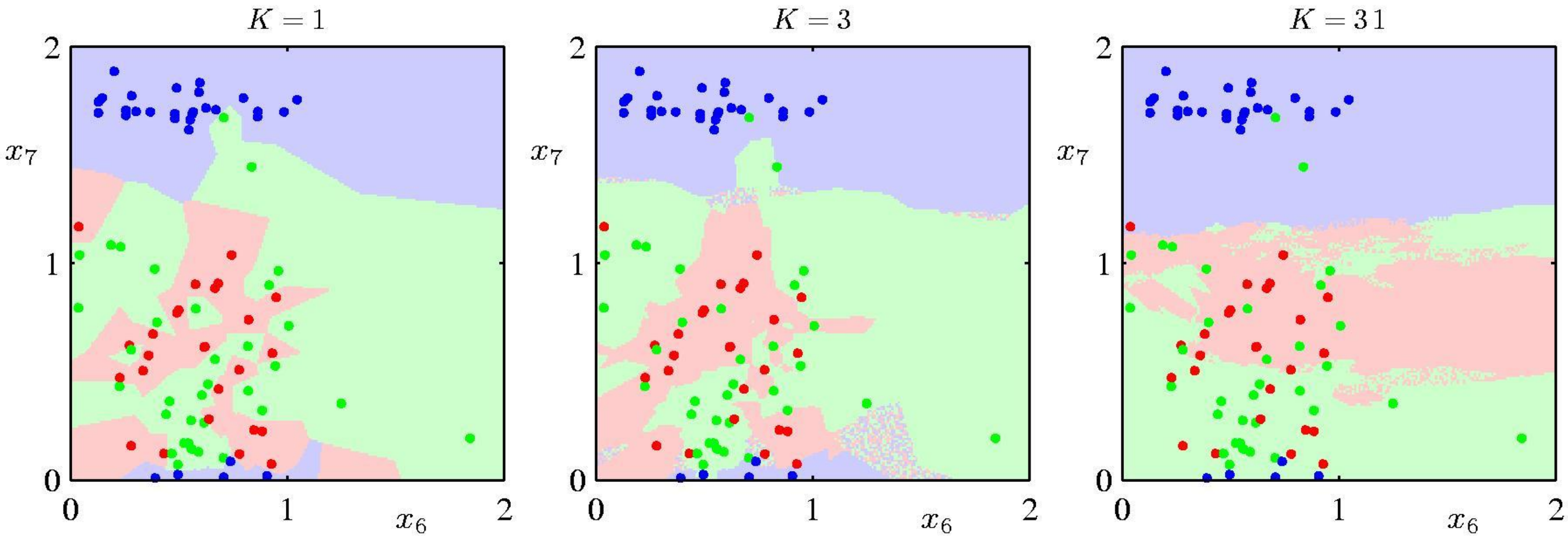
K Nearest Neighbour (K-NN) Classifier

e.g. K = 3

- applicable to multi-class case



K-Nearest-Neighbours for multi-class classification



Vote among multiple classes

Vector distances

Vector norms: A norm of a vector ||x|| is informally a measure of the "length" of the vector.

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

– Common norms: L_1, L_2 (Euclidean)

$$||x||_1 = \sum_{i=1}^{n} |x_i| \qquad ||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

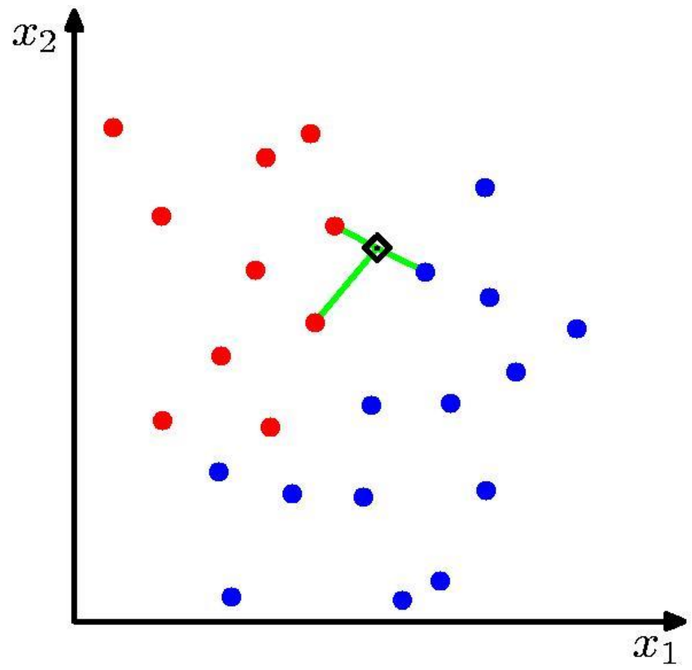
Norm can be used as distance between vectors x and y

- $||x - y||_p$

Distance norms

- Euclidean Distance $$\sqrt{(\sum_{i=1}^{k}(x_i - y_i)^2)}$$
- Mahattan Distance $$\sum_{i=1}^{k}|x_i - y_i|$$
- Minkowski Distance $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$$

kNN

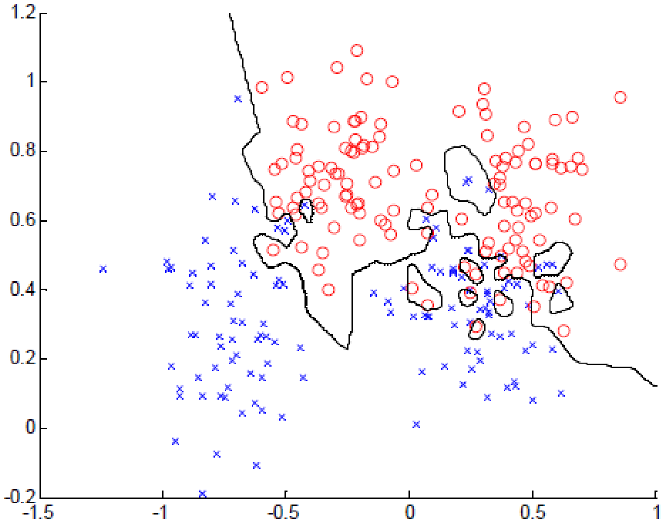


- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

K = 1

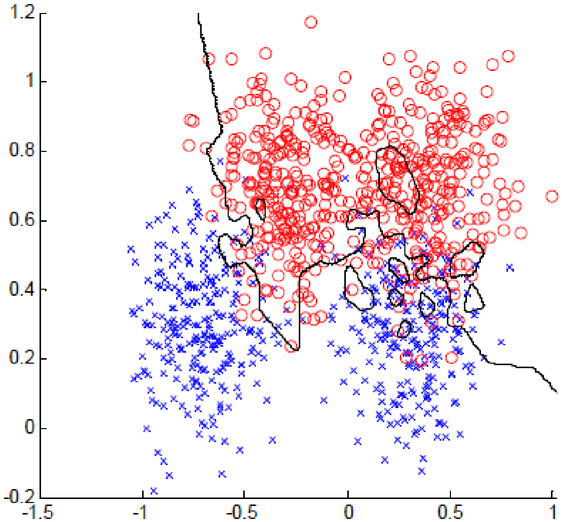
Overfitting!

Training data



error = 0.0

Testing data

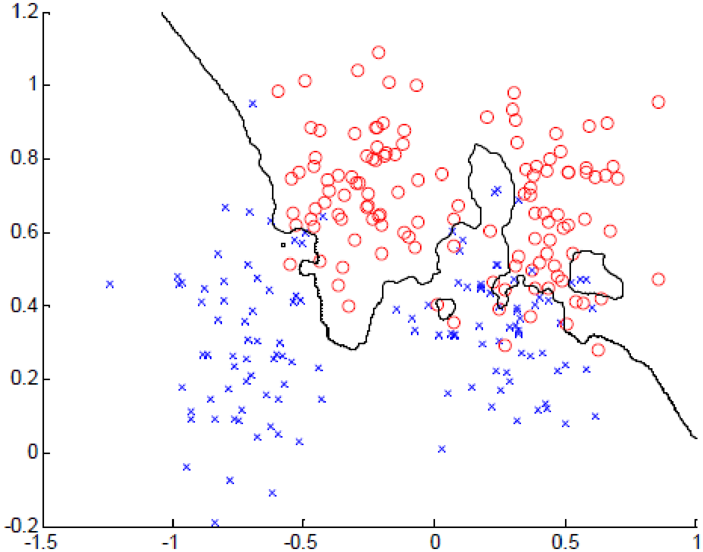


error = 0.15

How to choose k (hyper-parameter)?

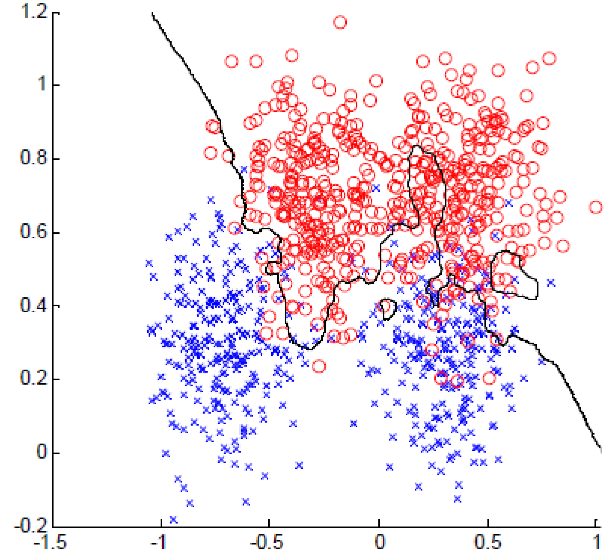
K = 3

Training data



error = 0.0760

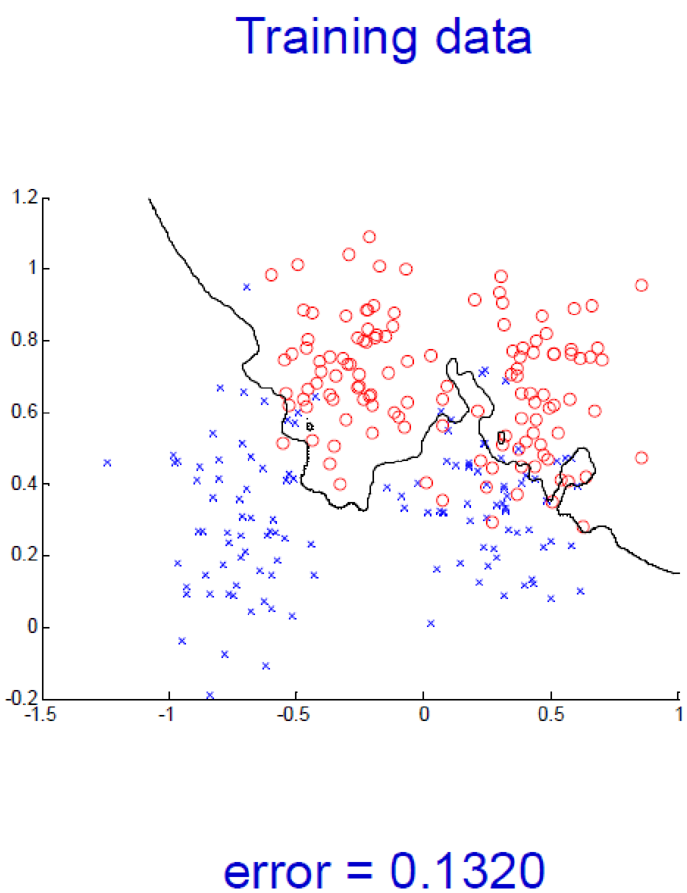
Testing data

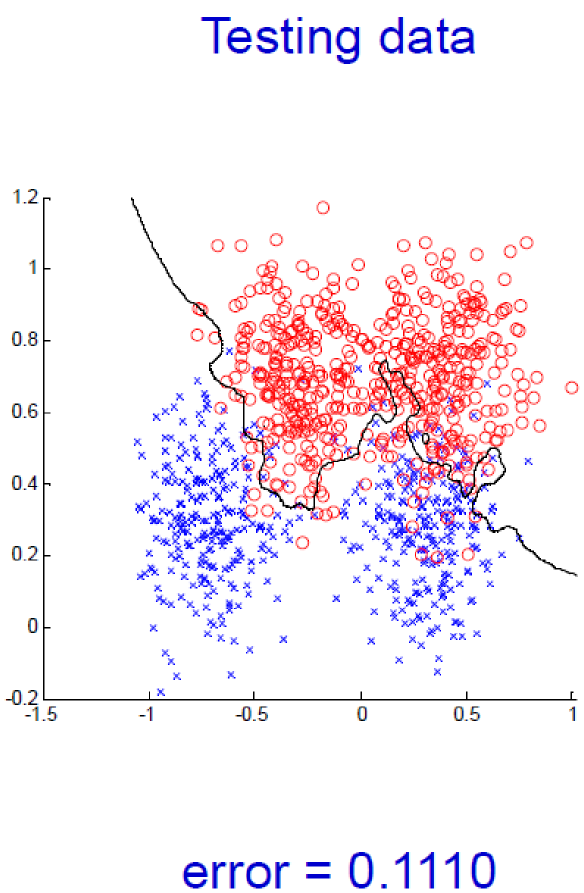


error = 0.1340

How to choose k (hyper-parameter)?

K = 7





How to choose k (hyper-parameter)?

Review

- Gradient descent is an efficient algorithm for optimization and training LR
  – The most widely used algorithm in ML!
- More complex regression models exist
  – Polynomial, spline regression
- Regularization is general method to reduce model complexity and avoid overfitting
  – Add penalty to loss function
  – Ridge and Lasso regression

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!