Practical Issues in ML:

Generative and Discriminative Learning
Feature Selection
Regularization
Cross Validation
Estimating Error

Machine Learning 10-601
Feb 9, 2009

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

Questions:

- Can you use Naïve Bayes for a combination of discrete and real-valued X_i?
- How can we easily model just 2 of n attributes as dependent?
- What does the decision surface of a Naïve Bayes classifier look like? What about Logistic Regression?
- How would you select a subset of X_i's?

Relaxing Cond Indep in Naïve Bayes: HW4, Q1.3

- What if we have Y boolean, X=<X1, X2, … Xn>, and we believe all Xi are cond indep given Y, *except* for X1, X2?

$$P(Y \mid \langle X_1 X_2 \cdots X_n \rangle) = \frac{P(Y)\, P(X_1 X_2 \cdots X_n \mid Y)}{P(X)}$$

chain rule

$$P(X_1 X_2 \cdots X_n \mid Y) = P(X_1 \mid X_2 \cdots X_n Y)\, P(X_2 \cdots X_n \mid Y)$$

$$= P(X_1 \mid X_2 \cdots X_n Y)\, P(X_2 \mid X_3 \cdots X_n Y)\, P(X_3 \mid X_4 \cdots X_n) \cdots$$

$P(X_1 \mid Y)$

$$\vdots$$

$$P(X_8 \mid X_9 X_{10} Y)\, P(X_9 X_{10} \mid Y)$$

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Generative classifiers (e.g., Naïve Bayes)

- Assume some functional form for P(X|Y), P(Y)
- Estimate parameters of P(X|Y), P(Y) directly from training data
- Use Bayes rule to calculate P(Y|X= x)

Discriminative classifiers (e.g., Logistic regression)

- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

Use Naïve Bayes or Logisitic Regression?

Consider

- Restrictiveness of modeling assumptions

- Rate of convergence (in amount of training data) toward asymptotic hypothesis

Naïve Bayes vs Logistic Regression

Consider Y boolean, X_i continuous, $X=<X_1 \ldots X_n>$

Number of parameters:

- NB: 4n +1
- LR: n+1

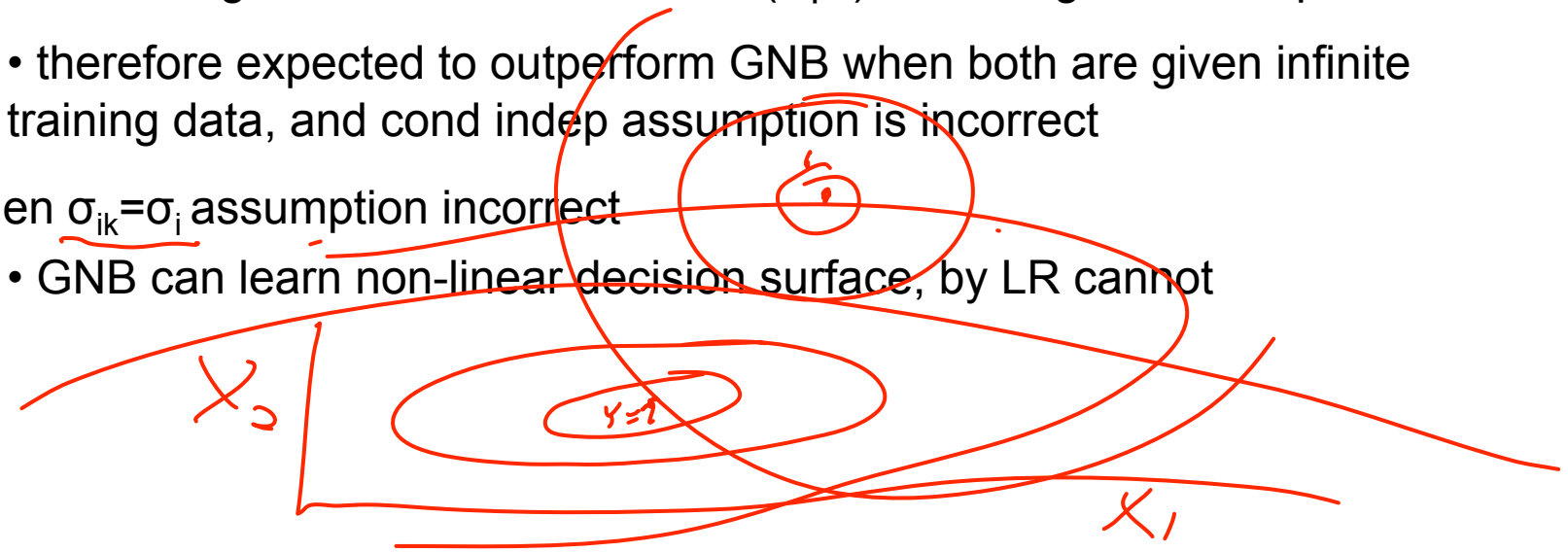
Estimation method:

- NB parameter estimates are uncoupled
- LR parameter estimates are coupled

G.Naïve Bayes vs. Logistic Regression

- Generative and Discriminative classifiers $P(Y=1|X) = \frac{1}{1+\exp(\vec{w}\cdot\vec{X})}$

- *Asymptotic comparison* (# training examples → infinity)
  - when conditional independence assumptions correct, and $\sigma_{ik}=\sigma_i$, i.e. when LR P(Y|X) is same as derived from NB.
    - GNB, LR produce identical classifiers
  - when conditional independence assumptions incorrect
    - LR is less biased – does not assume cond indep. in its parameter estimation method for $\langle w_0, w_1, \ldots w_n \rangle$
      - though we did derive form of P(Y|X) assuming cond indep
    - therefore expected to outperform GNB when both are given infinite training data, and cond indep assumption is incorrect
  - when $\sigma_{ik}=\sigma_i$ assumption incorrect
    - GNB can learn non-linear decision surface, by LR cannot

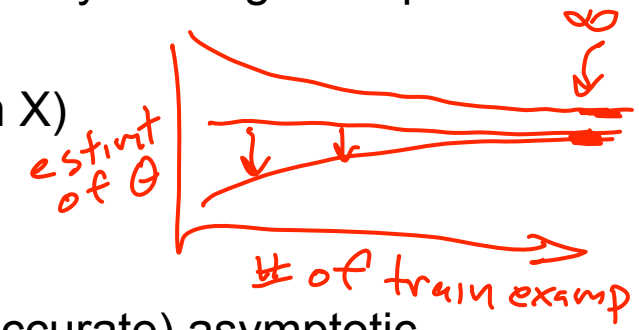


Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative and Discriminative classifiers

- *Non-asymptotic analysis* (see [Ng & Jordan, 2002])
  - convergence rate of parameter estimates – how many training examples needed to assure good estimates?
    - GNB order log n (where n = # of attributes in X)
    - LR order n



GNB converges more quickly to its (perhaps less accurate) asymptotic estimates

Informally: because LR's parameter estimates are coupled, but GNB's are not

Rate of covergence: logistic regression

[Ng & Jordan, 2002]

Let $h_{Dis,m}$ be logistic regression trained on m examples in n dimensions. Then with high probability:

$$\epsilon(h_{Dis,m}) \leq \epsilon(h_{Dis,\infty}) + O(\sqrt{\frac{n}{m} \log \frac{m}{n}})$$

Implication: if we want $\epsilon(h_{Dis,m}) \leq \epsilon(h_{Dis,\infty}) + \epsilon_0$
for some constant ϵ_0, it suffices to pick order n examples

→ Convergences to its asymptotic classifier, in order n examples
(result follows from Vapnik's structural risk bound, plus fact that VCDim of n dimensional linear separators is n)

Rate of covergence: naïve Bayes parameters

[Ng & Jordan, 2002]

Let any $\epsilon_1, \delta > 0$ and any $l \geq 0$ be fixed. Assume that for some fixed $\rho_0 > 0$, we have that $\rho_0 \leq p(y = T) \leq 1 - \rho_0$. Let $m = O((1/\epsilon_1^2) \log(n/\delta))$. Then with probability at least $1 - \delta$, after m examples:

1. For discrete inputs, $|\widehat{p}(x_i|y = b) - p(x_i|y = b)| \leq \epsilon_1$, and $|\widehat{p}(y = b) - p(y = b)| \leq \epsilon_1$, for all i, b.

2. For continuous inputs, $|\widehat{\mu}_{i|y=b} - \mu_{i|y=b}| \leq \epsilon_1$, and $|\widehat{\sigma}_i^2 - \sigma_i^2| \leq \epsilon_1$, for all i, b.

Some experiments from UCI data sets

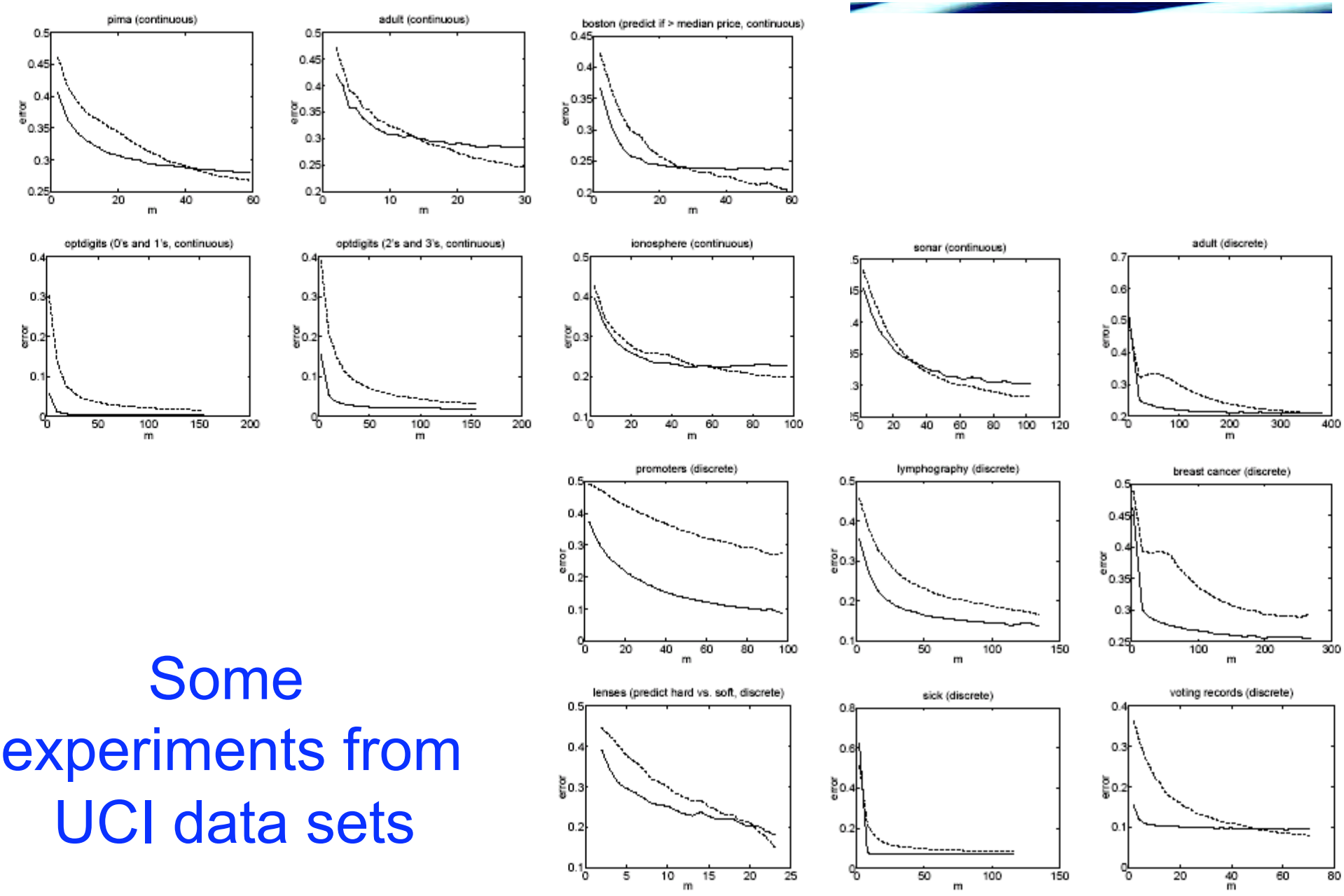


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. m (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

Summary: Naïve Bayes and Logistic Regression

- Modeling assumptions
  - Naïve Bayes more biased (cond. indep)
  - Both learn linear decision surfaces if we assume $\sigma_{ik} = \sigma$
- Convergence rate (n=number training examples)
  - Naïve Bayes ~ O(log n)
  - Logistic regression ~O(n)
- Bottom line
  - Naïve Bayes converges faster to its (potentially too restricted) final hypothesis

What you should know:

- Logistic regression
  - Functional form follows from Naïve Bayes assumptions
    - For Gaussian Naïve Bayes assuming variance $\sigma_{i,k} = \sigma_l$
    - For discrete-valued Naïve Bayes too
  - But training procedure picks parameters without the conditional independence assumption
  - MLE training: pick W to maximize P(Y | X, W)
  - MAP training: pick W to maximize P(W | X,Y)
    - 'regularization'
    - helps reduce overfitting

- Gradient ascent/descent
  - General approach when closed-form solutions unavailable

- Generative vs. Discriminative classifiers
  - Bias vs. variance tradeoff

Question

You have 100 medical patients to train a ‘will survive surgery’ classifier

You want the absolute most accurate classifier you can get

You also want a good estimate of how accurate it is (so you know whether or not to use it!)

what do you do?

Estimating Accuracy, and Confidence in this Estimate

Two Definitions of Error

The **true error** of hypothesis h with respect to target function f and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$

The **sample error** of h with respect to target function f and data sample S is the proportion of examples h misclassifies

$$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x) \neq h(x))$$

Where $\delta(f(x) \neq h(x))$ is 1 if $f(x) \neq h(x)$, and 0 otherwise.

How well does $error_S(h)$ estimate $error_{\mathcal{D}}(h)$?

Problems Estimating Error

1. *Bias:* If S is the training set, $error_S(h)$ is (almost always) optimistically biased

$$bias \equiv E[error_S(h)] - error_{\mathcal{D}}(h)$$

   This is also true if *any part* of the training procedure used *any part* of S, e.g. for feature engineering, feature selection, parameter tuning, ...

   For an unbiased estimate, h and S must be chosen independently

2. *Variance:* Even with unbiased S, $error_S(h)$ may still *vary* from $error_{\mathcal{D}}(h)$

   Variance of X is $Var(X) \equiv E[(X - E[X])^2]$

Example

Hypothesis h misclassifies 12 of the 40 examples in S

$$error_S(h) = \frac{12}{40} = .30$$

What is $error_{\mathcal{D}}(h)$?

Estimators

Experiment:

1. choose sample S of size n according to distribution $\mathcal{D}$
2. measure $error_S(h)$

$error_S(h)$ is a random variable (i.e., result of an experiment)

$error_S(h)$ is an unbiased *estimator* for $error_{\mathcal{D}}(h)$

Given observed $error_S(h)$ what can we conclude about $error_{\mathcal{D}}(h)$?

Confidence Intervals

If

- S contains n examples, drawn independently of h and each other
- $n \geq 30$

Then

- With approximately 95% probability, $error_{\mathcal{D}}(h)$ lies in interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Confidence Intervals

If

- S contains n examples, drawn independently of h and each other
- $n \geq 30$

Then

- With approximately N% probability, $error_{\mathcal{D}}(h)$ lies in interval

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

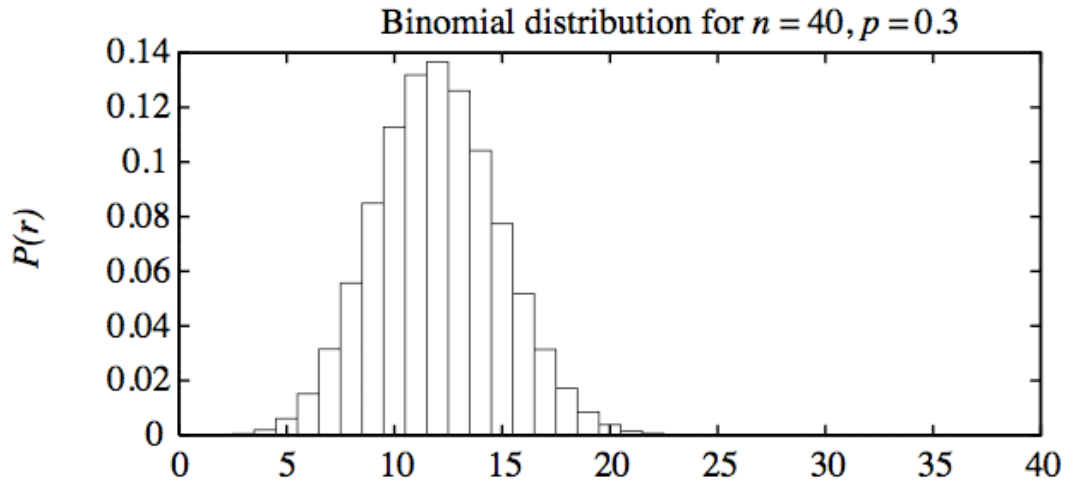
where

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

$error_S(h)$ is a Random Variable

Rerun the experiment with different randomly drawn S (of size n)

Probability of observing r misclassified examples:



$$P(r) = \frac{n!}{r!(n-r)!} error_D(h)^r (1 - error_D(h))^{n-r}$$

Normal Distribution Approximates Binomial

$error_S(h)$ follows a *Binomial* distribution, with

- mean $\mu_{error_S(h)} = error_{\mathcal{D}}(h)$
- standard deviation $\sigma_{error_S(h)}$

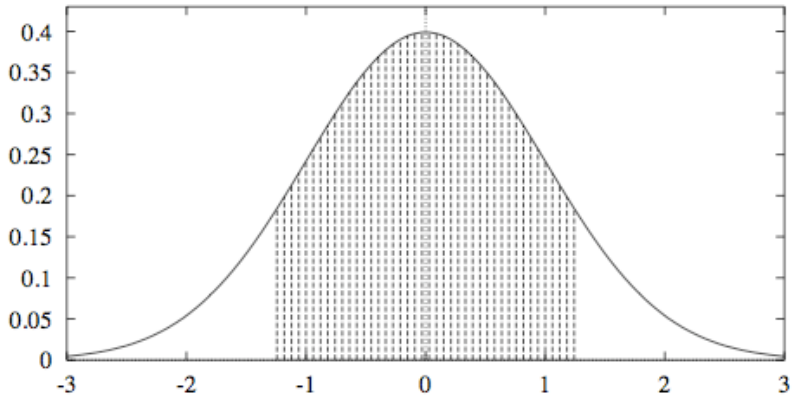
$$\sigma_{error_S(h)} = \sqrt{\frac{error_{\mathcal{D}}(h)(1 - error_{\mathcal{D}}(h))}{n}}$$

Approximate this by a *Normal* distribution with

- mean $\mu_{error_S(h)} = error_{\mathcal{D}}(h)$
- standard deviation $\sigma_{error_S(h)}$

$$\sigma_{error_S(h)} \approx \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Normal Probability Distribution



80% of area (probability) lies in $\mu \pm 1.28\sigma$

N% of area (probability) lies in $\mu \pm z_N\sigma$

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Cross Validation

Question

You have 100 medical patients to train a ‘will survive surgery’ classifier

You want the absolute most accurate classifier you can get

You also want a good estimate of how accurate it is (so you know whether or not to use it!)

what do you do?

K-Fold Cross Validation

Idea: train multiple times, leaving out a disjoint subset of data each time for testing. Average the test accuracies.

Partition data into K disjoint subsets

For k=1 to K

testData = kth subset

h ← classifier trained on all data except for testData

accuracy(k) = accuracy of h on testData

end

FinalAccuracy = mean of the K recorded accuracies

Leave-One-Out Cross Validation

This is just k-fold cross validation leaving out one example each iteration

---

Partition data into K disjoint subsets, *each containing one example*

For k=1 to K

testData = kth subset

h ← classifier trained on all data except for testData

accuracy(k) = accuracy of h on testData

end

FinalAccuracy = mean of the K recorded accuracies

K-Fold Cross Validation

Given a data set containing N examples, k-fold cross validation yields a nearly unbiased estimate of the accuracy expected when training on a randomly drawn sample of size N * (k-1) / k

What should we do with our 100 medical patient examples?

1. estimate accuracy using leave-one-out cross validation
   - this will provide an estimate of expected accuracy when training on 99 examples
2. train final classifier using all 100 examples
   - this will provide classifier with higher expected accuracy than training on 99 examples, but high accuracy is our goal

Supervised Feature Selection

Supervised Feature Selection

Problem: Wish to learn f: X → Y, where X=<X_1, …X_N>

But suspect not all X_i are relevant

Approach: Preprocess data to select only a subset of the X_i

- Score each feature, or subsets of features
  - How?
- Search for useful subset of features to represent data
  - How?

Scoring Individual Features X_i

Common scoring methods:

- Training or cross-validated accuracy of single-feature classifiers f_i: $X_i \rightarrow Y$

- Estimated mutual information between X_i and Y :

$$\hat{I}(X_i, Y) = \sum_k \sum_y \hat{P}(X_i = k, Y = y) \log \frac{\hat{P}(X_i = k, Y = y)}{\hat{P}(X_i = k)\hat{P}(Y = y)}$$

- χ^2 statistic to measure independence between X_i and Y

- Domain specific criteria
  - Text: Score “stop” words (“the”, “of”, …) as zero
  - fMRI: Score voxel by T-test for activation versus rest condition
  - …

Choosing Set of Features to learn F: X→Y

Common methods:

Forward1: Choose the n features with the highest scores

Forward2:

- Choose single highest scoring feature X_k
- Rescore all features, conditioned on the set of already-selected features
  - E.g., Score(X_i | X_k) = I(X_i,Y |X_k)
  - E.g, Score(X_i | X_k) = Accuracy(predicting Y from X_i and X_k)
- Repeat, calculating new scores on each iteration, conditioning on set of selected features

Choosing Set of Features

Common methods:

Backward1: Start with all features, delete the n with lowest scores

Backward2: Start with all features, score each feature conditioned on assumption that all others are included. Then:

- Remove feature with the lowest (conditioned) score
- Rescore all features, conditioned on the new, reduced feature set
- Repeat

Feature Selection: Text Classification

Approximately 10^5 words in English

[Rogati&Yang, 2002]

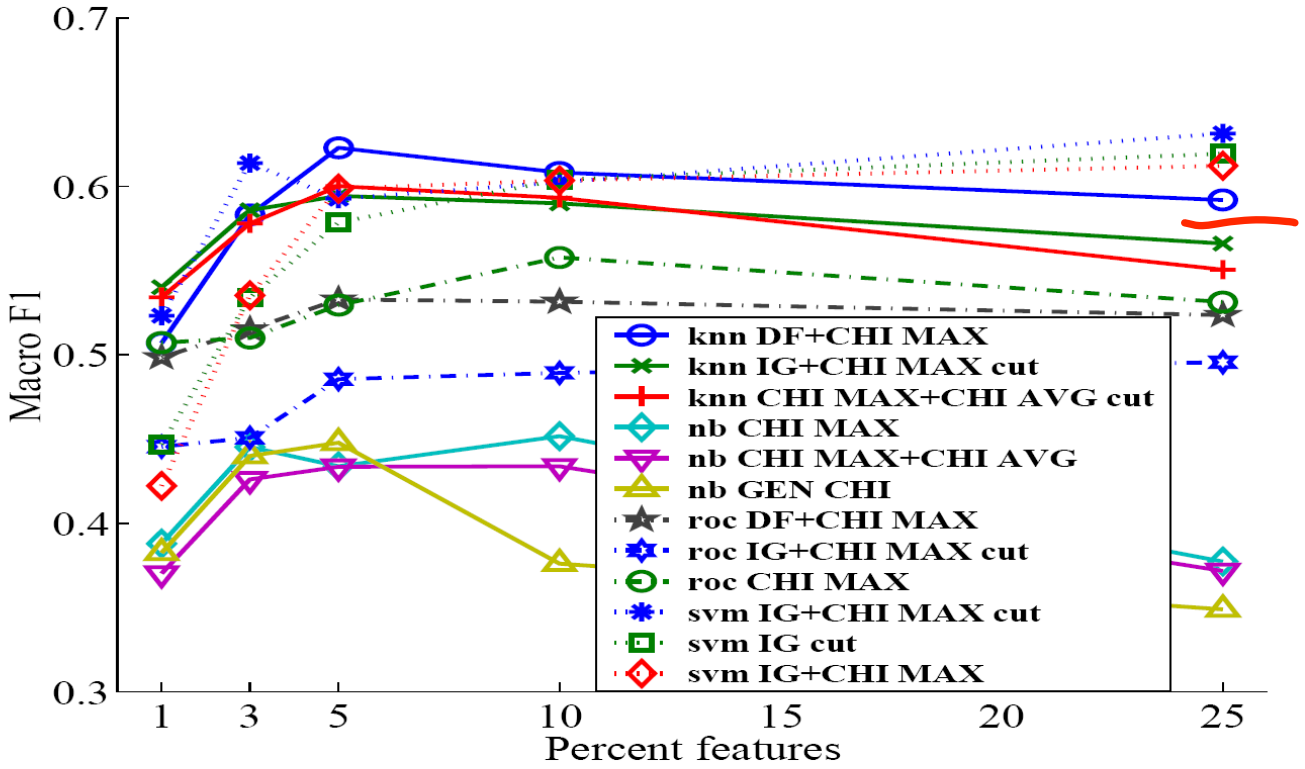


Figure 2: Top 3 feature selection methods for Reuters-21578 (Macro F1)

IG=information gain, chi= χ^2 , DF=doc frequency,

Impact of Feature Selection on Classification of fMRI Data

[Pereira et al., 2005]

Accuracy classifying category of word read by subject

#voxels	mean	subjects 233B	329B	332B	424B	474B	496B	77B	86B
50	0.735	0.783	0.817	0.55	0.783	0.75	0.8	0.65	0.75
100	0.742	0.767	0.8	0.533	0.817	0.85	0.783	0.6	0.783
200	0.737	0.783	0.783	0.517	0.817	0.883	0.75	0.583	0.783
300	**0.75**	**0.8**	**0.817**	**0.567**	**0.833**	**0.883**	**0.75**	**0.583**	**0.767**
400	0.742	0.8	0.783	0.583	0.85	0.833	0.75	0.583	0.75
800	0.735	0.833	0.817	0.567	0.833	0.833	0.7	0.55	0.75
1600	0.698	0.8	0.817	0.45	0.783	0.833	0.633	0.5	0.75
all (~2500)	0.638	0.767	0.767	0.25	0.75	0.833	0.567	0.433	0.733

Table 1: **Average accuracy across all pairs of categories, restricting the procedure to use a certain number of voxels for each subject.** The highlighted line corresponds to the best mean accuracy, obtained using 300 voxels.

Each feature X_i is a voxel, scored by error in regression to predict X_i from Y

Approach 2: Regularization

Key idea: add penalty to learning objective, to penalize large weights.

$$W = \arg\max_W \quad \lambda R(W) + \sum_l \ln P(Y^l | X^l; W)$$

Integrates 'feature selection' style pressure on weights, into learning algorithm – pushes them toward zero

- e.g., try L2 penalty which follows from N(0,σ) prior

$$R(W) = ||W||_2^2 = \sum_i w_i^2$$

Approach 2: Regularization

Integrate ‘feature selection’ style pressure on weights, into learning algorithm

$$W = \arg\max_W \quad \lambda R(W) + \sum_l \ln P(Y^l | X^l; W)$$

- L2 penalty which follows from N(0,σ) prior

$$R(W) = ||W||_2^2 = \sum_i w_i^2$$

- L1 penalty = sum of magnitudes of weights
  - encourages weights of zero

$$R(W) = ||W||_1 = \sum_i |w_i|$$

Approach 2: Regularization

Key idea: add L1 penalty to learning objective, to penalize large weights

- L1 penalty = sum of magnitudes of weights
- L2 penalty = sum of squares of weights

- think about L1 vs L2 for logistic regression…

Interesting Facts about L1, L2 regularization for Logistic Regression [Ng,2004]

- Logistic regression with L1 regularization requires a number of training examples that grows *logarithmically* with the number of *irrelevant* features

- Logistic regression with L2 regularization requires a number of training examples that grows *linearly* with the number of *irrelevant* features

So, if we suspect most of our features are irrelevant then L1 regularization is wise

Summary: Supervised Feature Selection

Approach 1: Preprocess data to select only a subset of the X_i

- Score each feature
  - Mutual information, prediction accuracy, …
- Find useful subset of features based on their scores
  - Greedy addition/deletion of features to pool
  - Considered independently, or in context of other selected features

Always do feature selection using training set only (why?)

- Often use nested cross-validation loop:
  - Outer loop to get unbiased estimate of final classifier accuracy
  - Inner loop to get unbiased feature scores for feature selection

Approach 2: use L1 or L2 regularization of parameters

- put pressure within training algorithm toward weights = 0