# Special Topic:

# Missing Values

# Missing Values Common in Real Data

- Pneumonia:
  - 6.3% of attribute values are missing
  - one attribute is missing in 61% of cases
- C-Section:
  - only about 1/2% of attribute values are missing
  - but 27.9% of cases have at least 1 missing value
- UCI machine learning repository:
  - 31 of 68 data sets reported to have missing values

# “Missing” Can Mean Many Things

- MAR: "Missing at Random":
  - usually best case
  - usually not true
- Non-randomly missing
- Presumed normal, so not measured
- Causally missing:
  - attribute value is missing because of other attribute values (or because of  the outcome value!)

# Dealing With Missing Data

- Some learning methods can handle missing values
- Throw away cases with missing values
  - in some data sets, most cases get thrown away
  - if not missing at random, throwing away cases can bias sample towards certain kinds of cases
- Treat "missing" as a new attribute value
  - what value should we use to code for missing with continuous or ordinal attributes?
  - if missing causally related to what is being predicted?
- Impute (fill-in) missing values
  - once filled in, data set is easy to use
  - if missing values poorly predicted, may hurt performance of subsequent uses of data set

# Imputing Missing Values

- Fill-in with mean, median, or most common value
- Predict missing values using machine learning
- Expectation Minimization (EM):
    - Build model of data values (ignore missing vals)
    - Use model to estimate missing values
    - Build new model of data values (including estimated values from previous step)
    - Use new model to re-estimate missing values
    - Re-estimate model
    - Repeat until convergence

# Potential Problems

- Imputed values may be inappropriate:
  - in medical databases, if missing values not imputed separately for male and female patients, may end up with male patients with 1.3 prior pregnancies, and female patients with low sperm counts
  - many of these situations will not be so humorous/obvious!
- If some attributes are difficult to predict, filled-in values may be random (or worse)
- Some of the best performing machine learning methods are impractical to use for filling in missing values (neural nets)
- Beware of coding - reliably detect missing cases can be difficult

# Research in Handling Missing Values

- Lazy learning:
  - don’t train a model until you know test case
  - missing in test case may “shadow” missing values in train set
- Better algorithms:
  - Expectation maximization (EM)
  - Non-parametric methods (since parametric methods often work poorly when assumptions are violated)
- Faster Algorithms:
  - apply to very large datasets

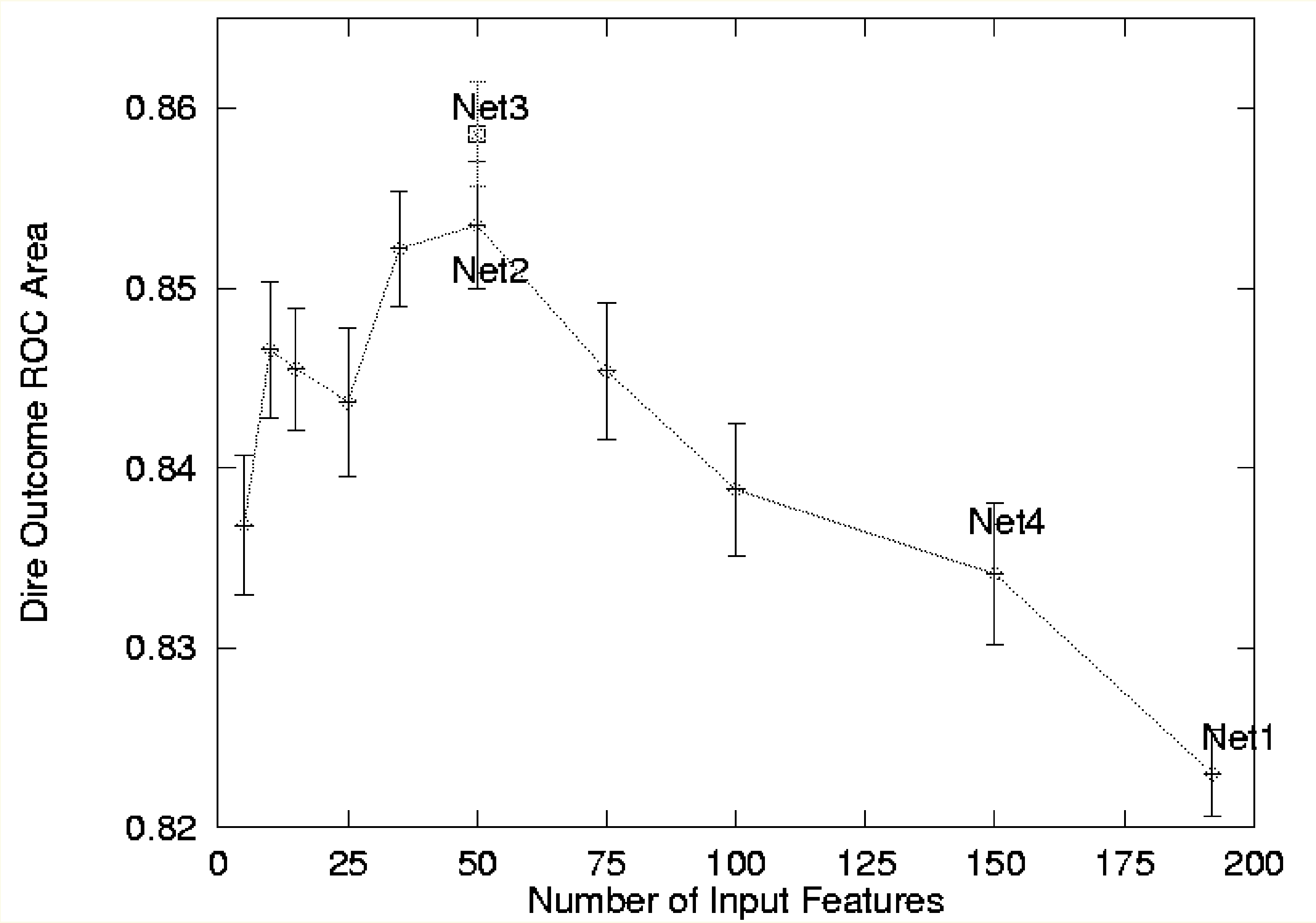
# Special Topic:

# Feature Selection

# Anti-Motivation

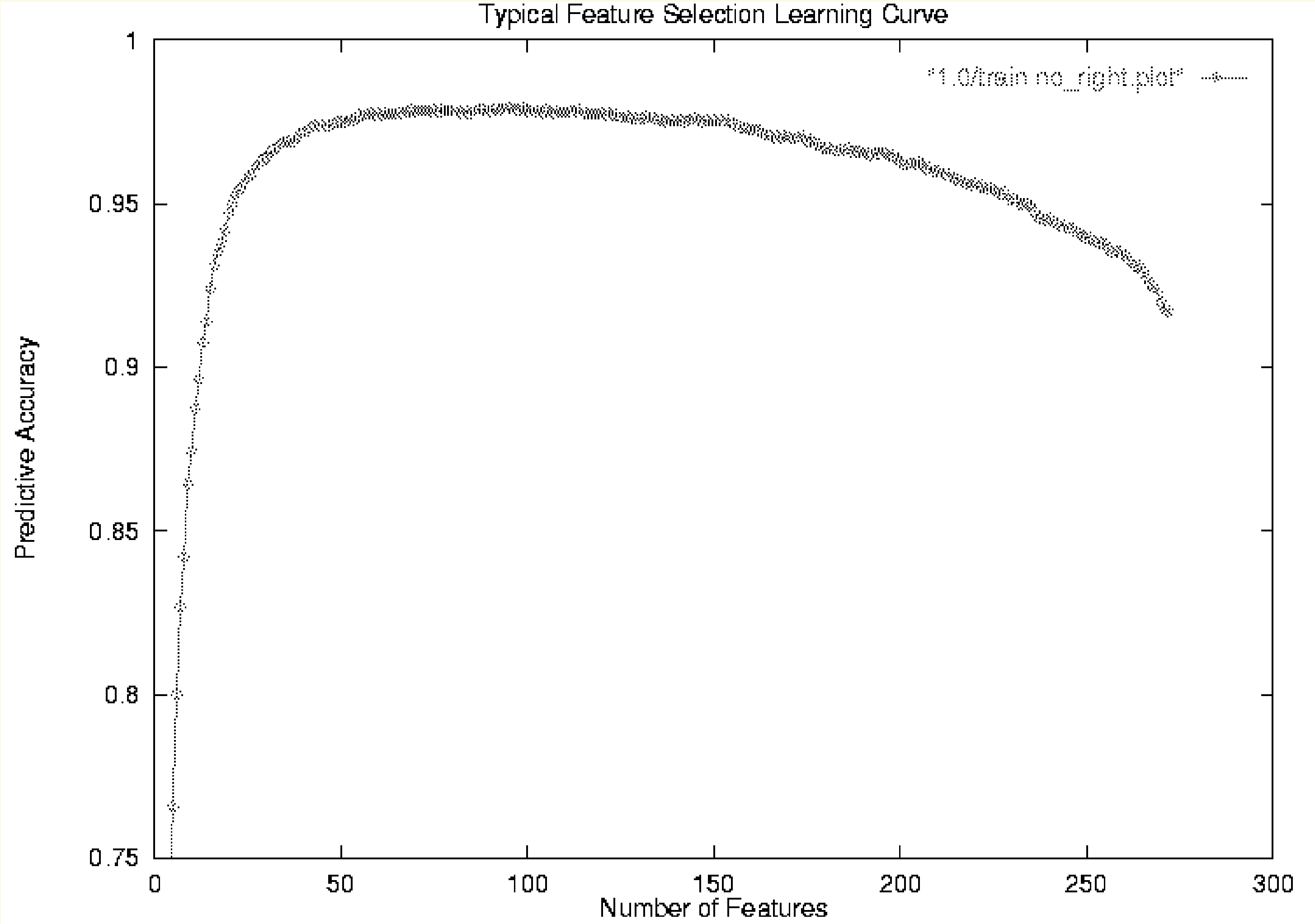
- Most learning methods implicitly do feature selection:
  - decision trees: use info gain or gain ratio to decide what attributes to use as tests. many features don't get used.
  - neural nets: backprop learns strong connections to some inputs, and near-zero connections to other inputs.
  - kNN, MBL: weights in Weighted Euclidean Distance determine how important each feature is. weights near zero mean feature is not used.
  - SVMs: maximum margin hyperplane may focus on important features, ignore irrelevant features.
- So why do we need feature selection?

# Motivation

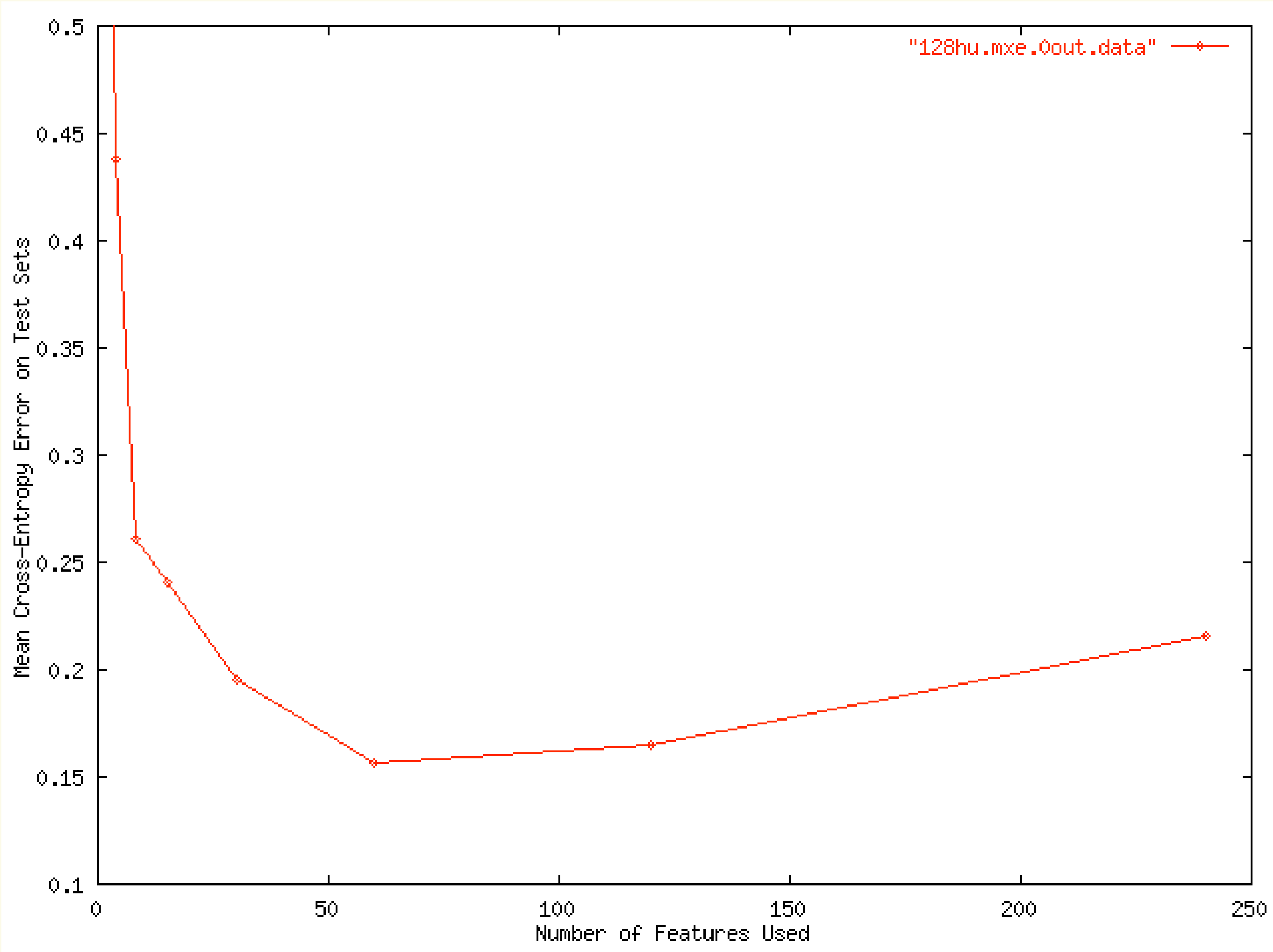


Net3
Net2
Net4
Net1
0.86
0.85
0.84
0.83
0.82
Dire Outcome ROC Area
0
25
50
75
100
125
150
175
200
Number of Input Features

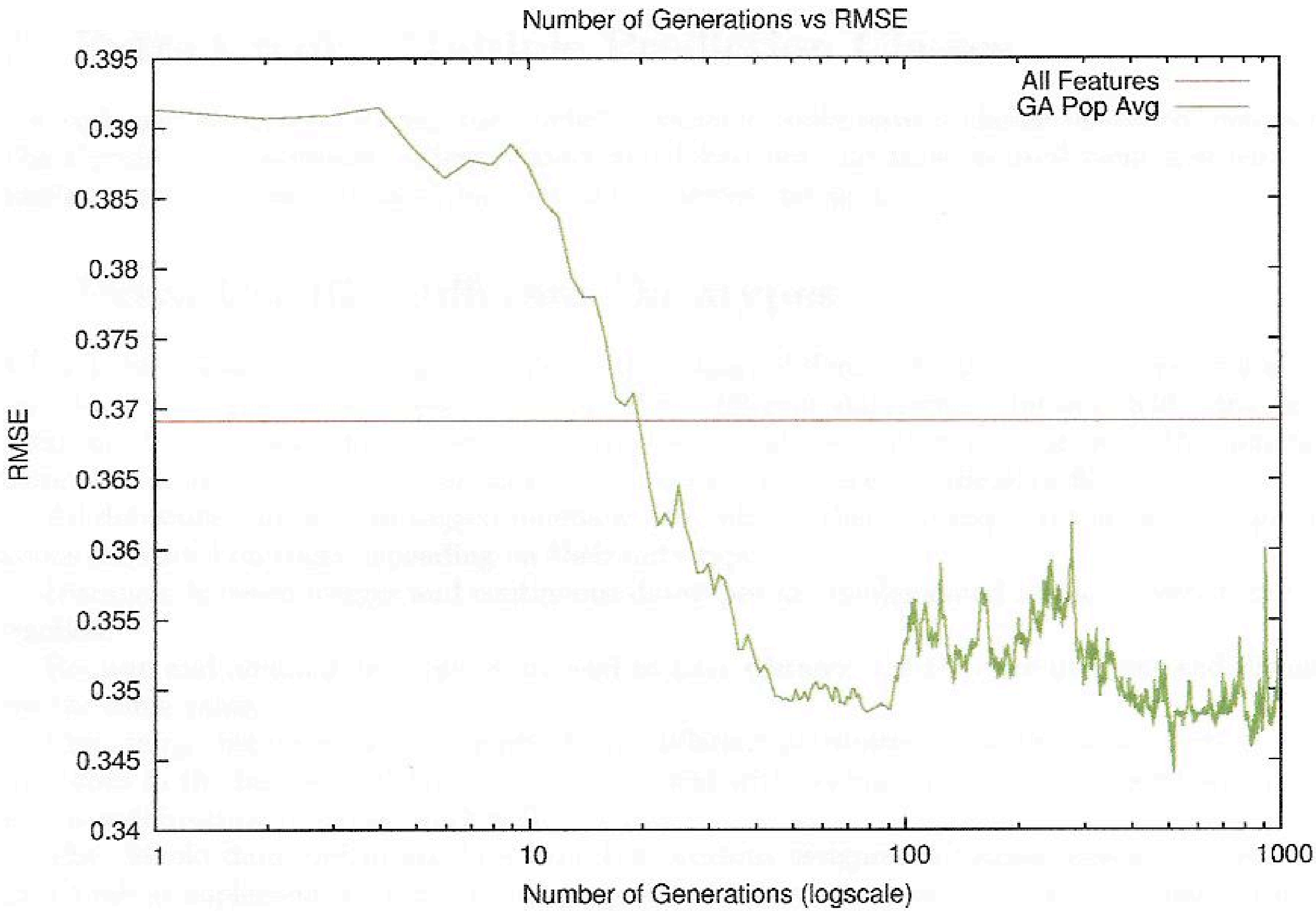
# Motivation


Typical Feature Selection Learning Curve
"1.0/train no_right.plot"
Predictive Accuracy
1
0.95
0.9
0.85
0.8
0.75
0
50
100
150
200
250
300
Number of Features

# Motivation

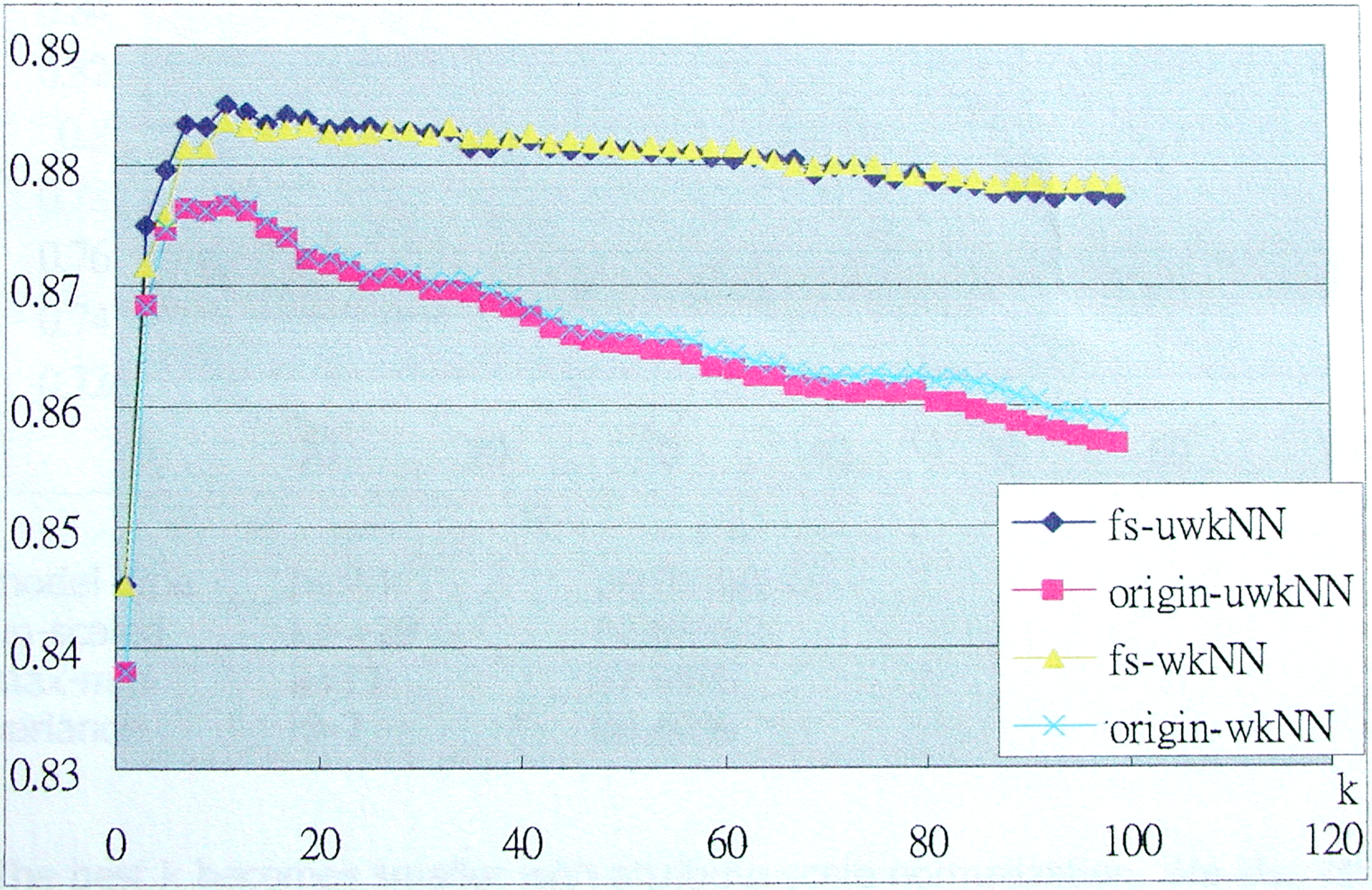


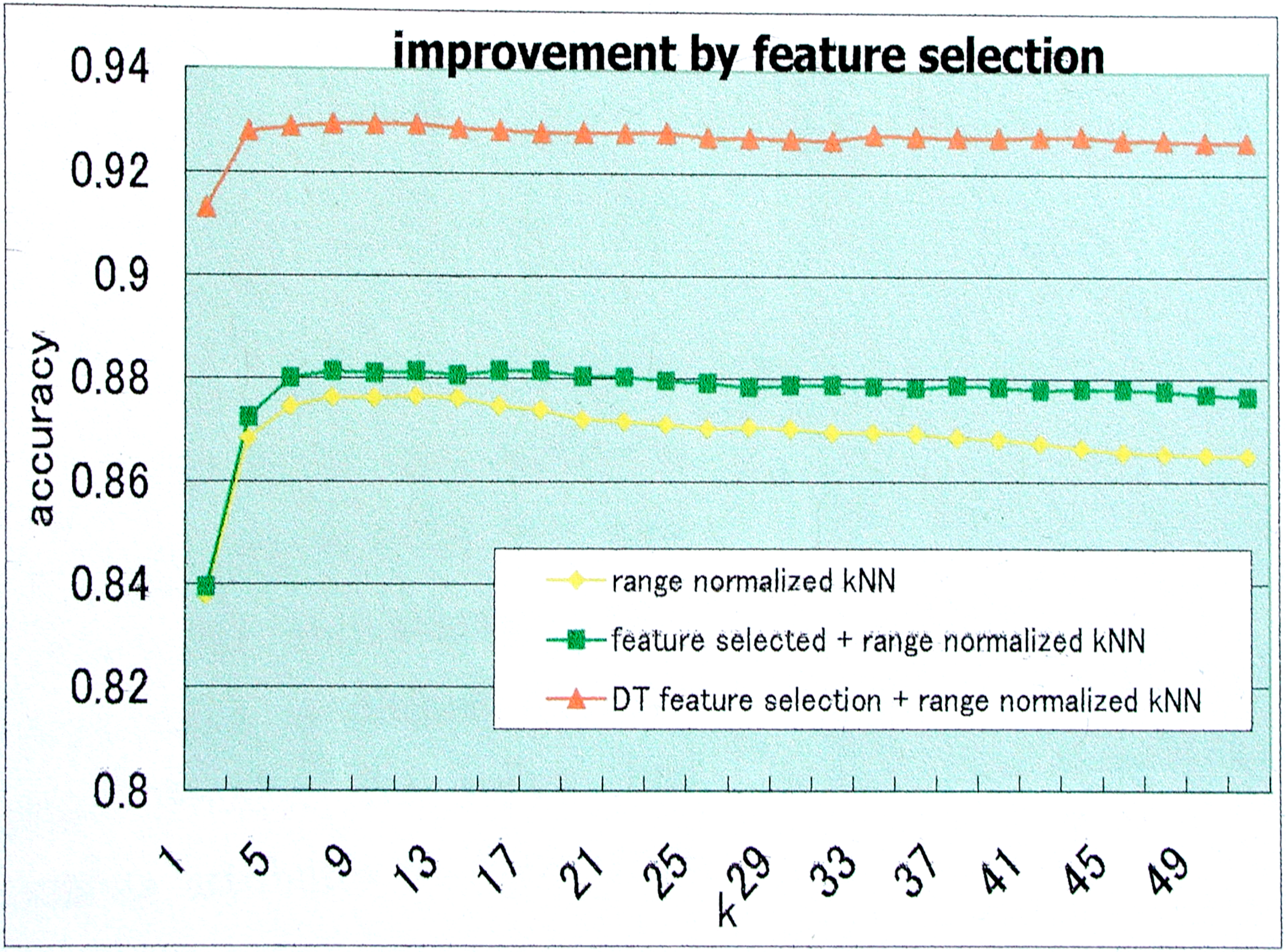
"128hu.mxe.0out.data"
Mean Cross-Entropy Error on Test Sets
Number of Features Used
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
50
100
150
200
250


Number of Generations vs RMSE
All Features
GA Pop Avg
RMSE
0.395
0.39
0.385
0.38
0.375
0.37
0.365
0.36
0.355
0.35
0.345
0.34
1
10
100
1000
Number of Generations (logscale)

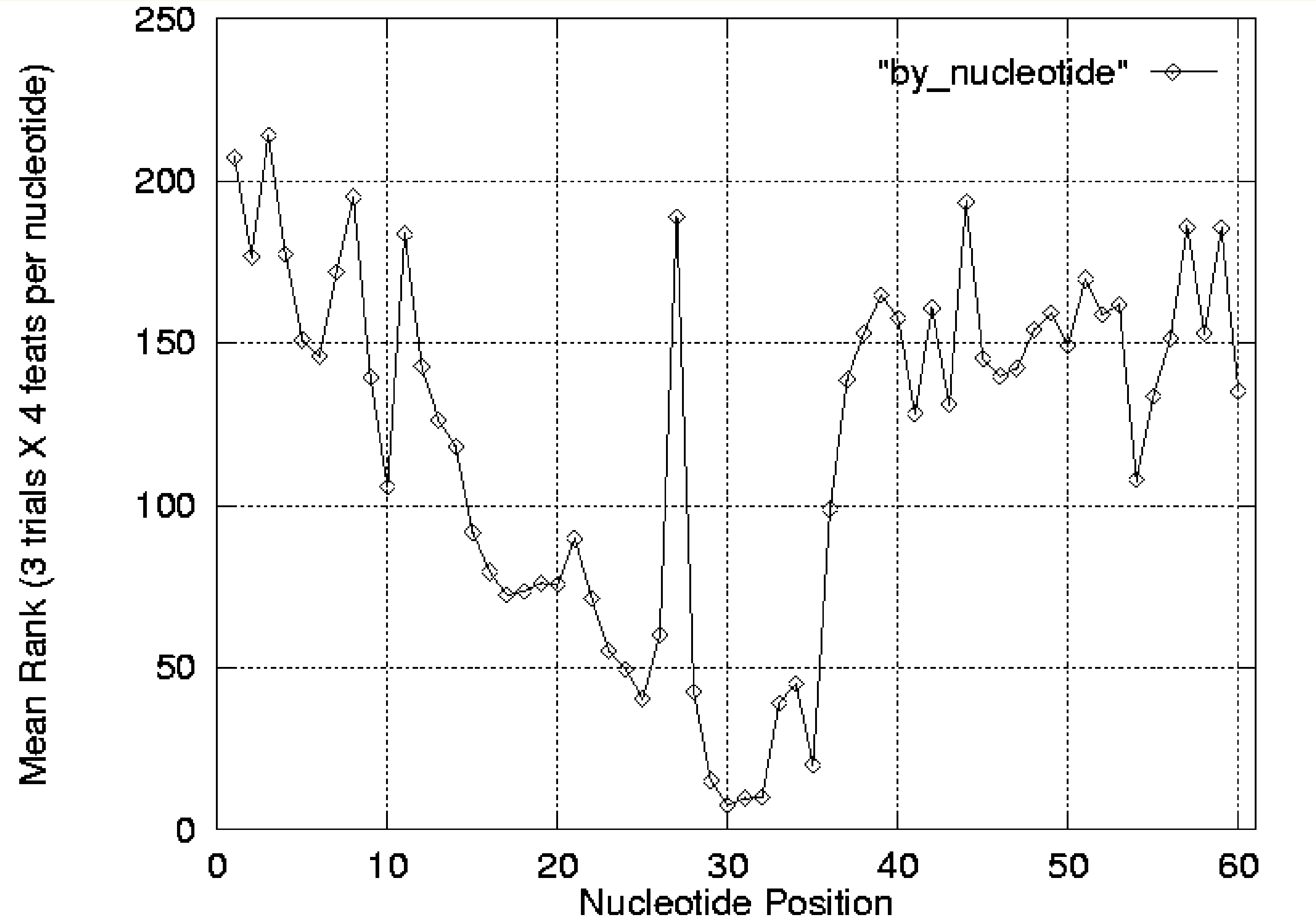
Features selected: Total 65 removed, 78 used. Please refer to Appendix A for

Performance comparison to original model (without feature selection):




improvement by feature selection
accuracy
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
1
5
9
13
17
21
25
29
33
37
41
45
49
k
range normalized kNN
feature selected + range normalized kNN
DT feature selection + range normalized kNN

# Motivation



"by_nucleotide"
Mean Rank (3 trials X 4 feats per nucleotide)
Nucleotide Position
0
50
100
150
200
250
0
10
20
30
40
50
60

# Brute-Force Approach

- Try all possible combinations of features
- Given N features, $2^N$ subsets of features
  - usually too many to try
  - danger of overfitting
- Train on train set, evaluate on test set (or use cross-validation)
- Use set of features that performs best on test set(s)

# Two Basic Approaches

- Wrapper Methods:
  - give different sets of features to the learning algorithm and see which works better
  - *algorithm dependent*
- Proxy Methods (relevance determination methods)
  - determine what features are important or not important for the prediction problem without knowing/using what learning algorithm will be employed
  - *algorithm independent*

# Wrapper Methods

- Wrapper methods find features that work best with some particular learning algorithm:
  - best features for kNN and neural nets may not be best features for decision trees
  - can eliminate features learning algorithm "*has trouble with*"
- Forward stepwise selection
- Backwards elimination
- Bi-directional stepwise selection and elimination

# Relevance Determination Methods

- Rank features by information gain
  - Info Gain = reduction in entropy due to attribute

$$Entropy = -p_{+} \log_2 p_{+} - p_{-} \log_2 p_{-}$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Try first 10, 20, 30, …, N features with learner
- Evaluate on test set (or use cross validation)
- May be only practical method if thousands of attributes

# Advantages of Feature Selection

- Improved accuracy!
- Less complex models:
  - run faster
  - easier to understand, verify, explain
- Feature selection points you to most important features
- Don’t need to collect/process features not used in models

# Limitations of Feature Selection

- Given many features, feature selection can overfit
  - consider 10 relevant features, and $10^9$ random irrelevant features
- Wrapper methods require running base learning algorithm many times, which can be expensive!
- Just because feature selection doesn't select a feature, doesn't mean that feature isn't a strong predictor
  - redundant features
- May throw away features domain experts want in model
- Most feature selection methods are greedy and won't find optimal feature set

# Current Research in Feature Selection

- Speeding-up feature selection (1000’s of features)
- Preventing overfitting (1000’s of features)
- Better proxy methods
  - would be nice to know what the good/relevant features are independent of the learning algorithm
- Irrelevance detection:
  - truly irrelevant attributes can be ignored
  - better algorithms
  - better definition(s)

# Bottom Line

- Feature selection almost always improves accuracy on real problems
- Plus:
  - simpler, more intelligible models
  - features selected can tell you about problem
  - less features to collect when using model in future

***Feature selection usually is a win.***