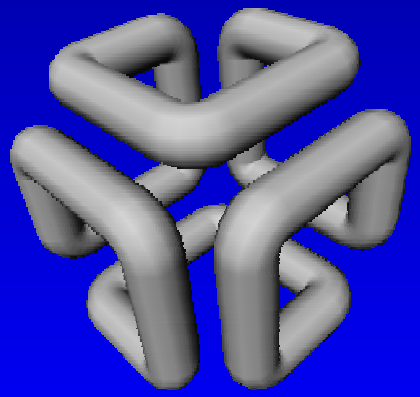
Bias Plus Variance Decomposition for Zero-One Loss Functions

Ronny Kohavi

Data Mining and Visualization Group
Silicon Graphics, Inc.

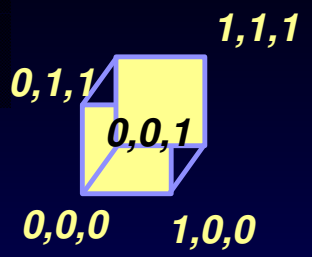


Joint work with

David Wolpert

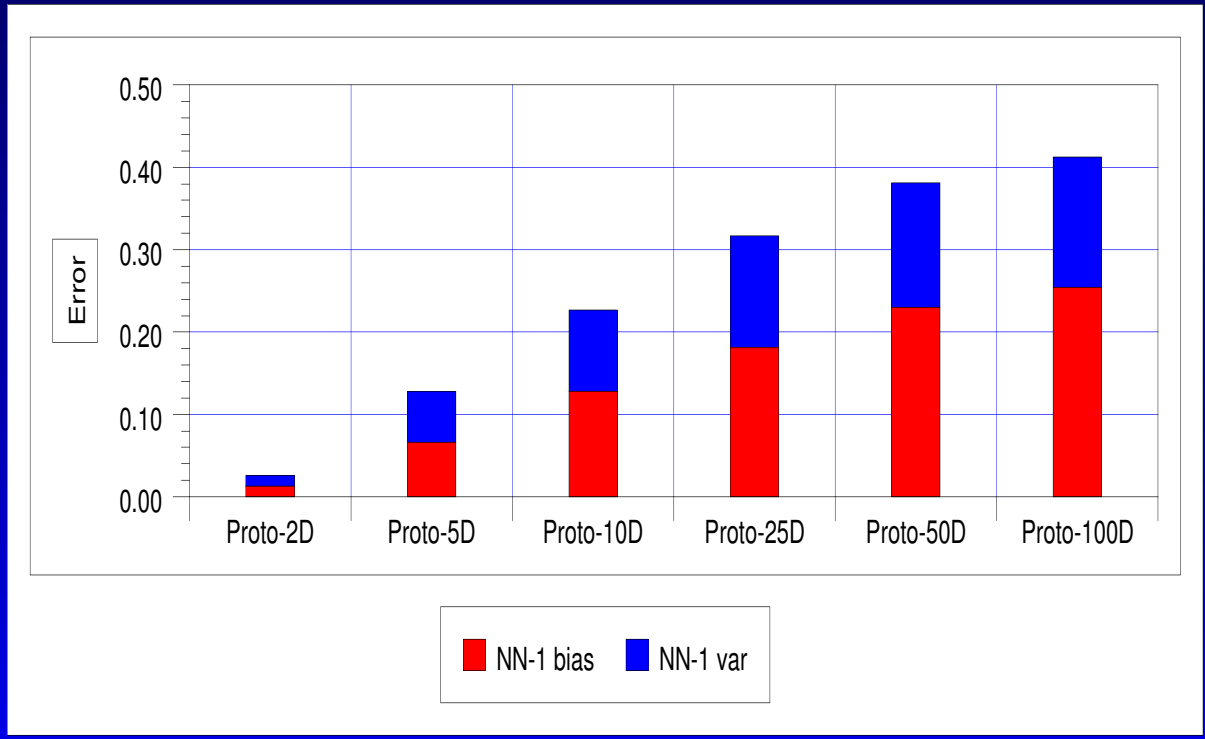
The Sante Fe Institute
(now at IBM, Almaden)

Curse of Dimensionality



20 dimensional unit hyper–cube.
100,000 instances uniformly distributed.
What is the expected distance of an instance to its closest neighbor?

- *0.1*
- *0.5*
- *0.7*
- *0.9*
- *0.99*
- *1.1*
- *1.5*
- *3.0*



Introduction

- Recent results in machine learning claim to show *surprising* results (no improvement, or worse performance) when obvious brute force *enhancements* to algorithms are attempted.
- Given the increase in CPU power, lookahead approaches and extensive searches become feasible, yet we constantly see negative results showing that the efforts fail.
- Researchers are sometimes at a loss explaining these results and call them pathologies, counter–intuitive, and oversearching.

Recent Examples

- Murphy and Pazzani (1994) found the smallest decision trees consistent with the data. The smallest trees typically had lower predictive accuracy than slightly larger trees.
- Quinlan and Cameron–Jones (1995) wrote
  More extensive search for rules is shown to lead to lower predictive accuracy.
- Murthy and Salzberg (1995) observed
  many instances of pathology, i.e., the lookahead produced trees that were both larger and less accurate.

Simple is Better

- Other researchers have shown that *simple* classifiers perform well:
- Holte (1993) showed that for many [small] datasets, looking at one feature is enough.
- Langley (1994), Kohavi & Sommerfield (1995) showed that Naive–Bayes, which assumes features are conditionally independent, outperforms C4.5 in some domains.
- Doughrety, Kohavi, Sahami (1995) showed that global discretization of each feature separately speeds up induction with no loss in accuracy.

Definitions

1. **X** is the input space.
2. **Y** is the output space.
3. The target **f** is $P(Y_F = y_F \mid x)$.
4. The hypothesis **h** is $P(Y_H = y_H \mid x)$.
5. The training set **d** is a set of **m** pairs of x–y values.
6. The expected misclassification rate (cost) is

$$E(C) = 1 - \sum_{y \in Y} P(y_H = y_F = y) \ .$$

We will drop the random variables in the rest of the talk.

The Decomposition

$$E(C) = \sum_x P(x) \left(\text{bias}_x^2 + \text{variance}_x + \sigma_x^2 \right) \tag{1}$$

where

$$\text{bias}_x^2 \equiv \frac{1}{2} \sum_{y \in Y} [P(Y_F = y \mid x) - P(Y_H = y \mid x)]^2 \tag{2}$$

$$\text{variance}_x \equiv \frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_H = y \mid x)^2 \right) \tag{3}$$

$$\sigma_x^2 \equiv \frac{1}{2} \left(1 - \sum_{y \in Y} P(Y_F = y \mid x)^2 \right) \ . \tag{4}$$

$$\tag{5}$$

f and m in the conditioning events are implicit.

Tree Pruning / Overfitting

- Consider splitting a node in a decision tree. Assume that the label is independent of the attributes, yet we are able to split and form pure nodes.

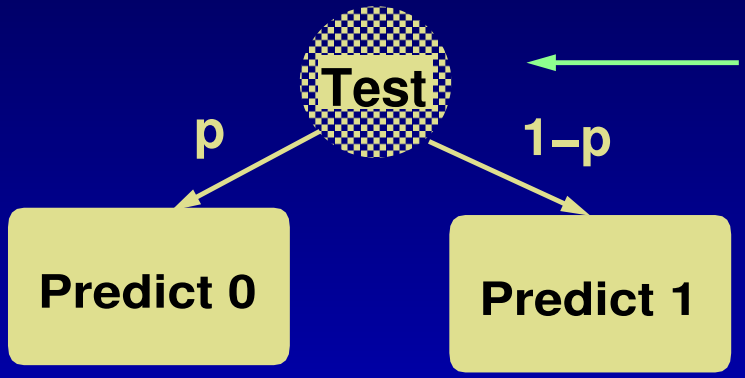
The node is not pure yet we stop and predict majority

Assume p<0.5 (predicting 1)

$Bias^2 = p^2$

Var = 0

Bias + Var = p^2



We split the node and get two pure nodes with the "right" probabilities

$Bias^2 = 0$

Var = p(1–p)

Bias + Var = p(1–p)

$p^2 < p(1-p)$ if p< 0.5, which we assumed. In this case, it is better not to split.

Relation to Quadratic Bias+Var

If the output space Y is turned into a vector of Indicator variables (1 for the class, 0 for others), then we show that the standard quadratic–loss bias+variance decomposition is exactly twice ours.

This leads to a simple relation to quadratic loss:

$$E\left[\left(\vec{Y_F} - \vec{Y_H}\right)^2 \mid f, m, x\right] = 2\,P(y_H \neq y_F \mid f, m, x) \quad (1)$$

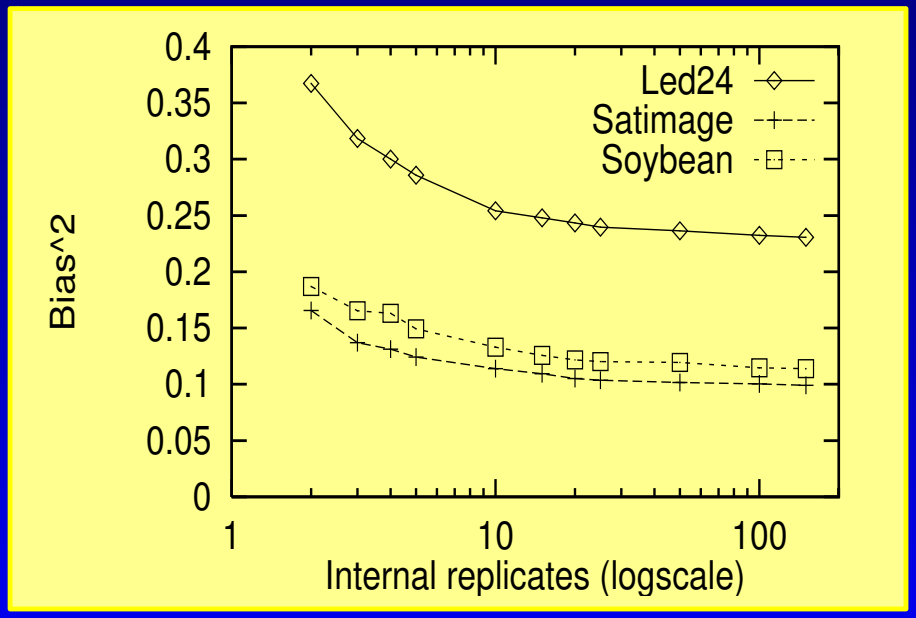
$$= 2\,E(C_{\text{0-1 loss}} \mid f, m, x)\,. \quad (2)$$

Experimental Methodology

- Randomly divide each dataset into two parts, **D** (dataset), and **E** (evaluation).
  To evaluate learning on size **m**, we made D of size 2m and kept the rest for E.
- Generate **N** training sets of size m from D using uniform random sampling without replacement.
- Run the learning algorithm on each training set and estimate the bias and variance for each instance in the set E.

Estimating the Quantities

- In theory, we need to generate all possible training sets of size m. In practice, we generate only N training sets (Monte–Carlo estimation).
- Averaging over a finite number of training sets gives a biased estimators for bias and variance



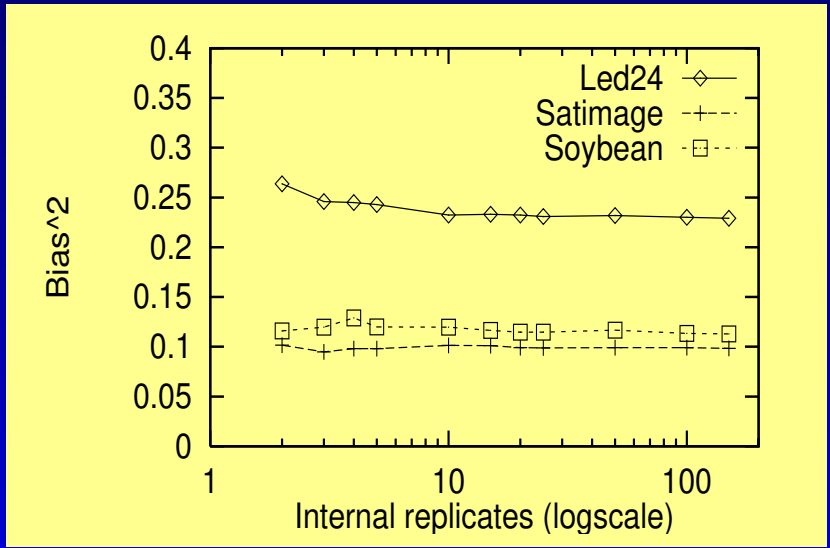
Note how the bias^2 decreases as N increases

Correcting the "Bias"

- The bias is a squared quantity, b^2. While our estimate for b is unbiased, our estimate for b^2 is biased because $E[b^2] >= (E[b])^2$ ← Jensen's inequality

Our estimate

What we want



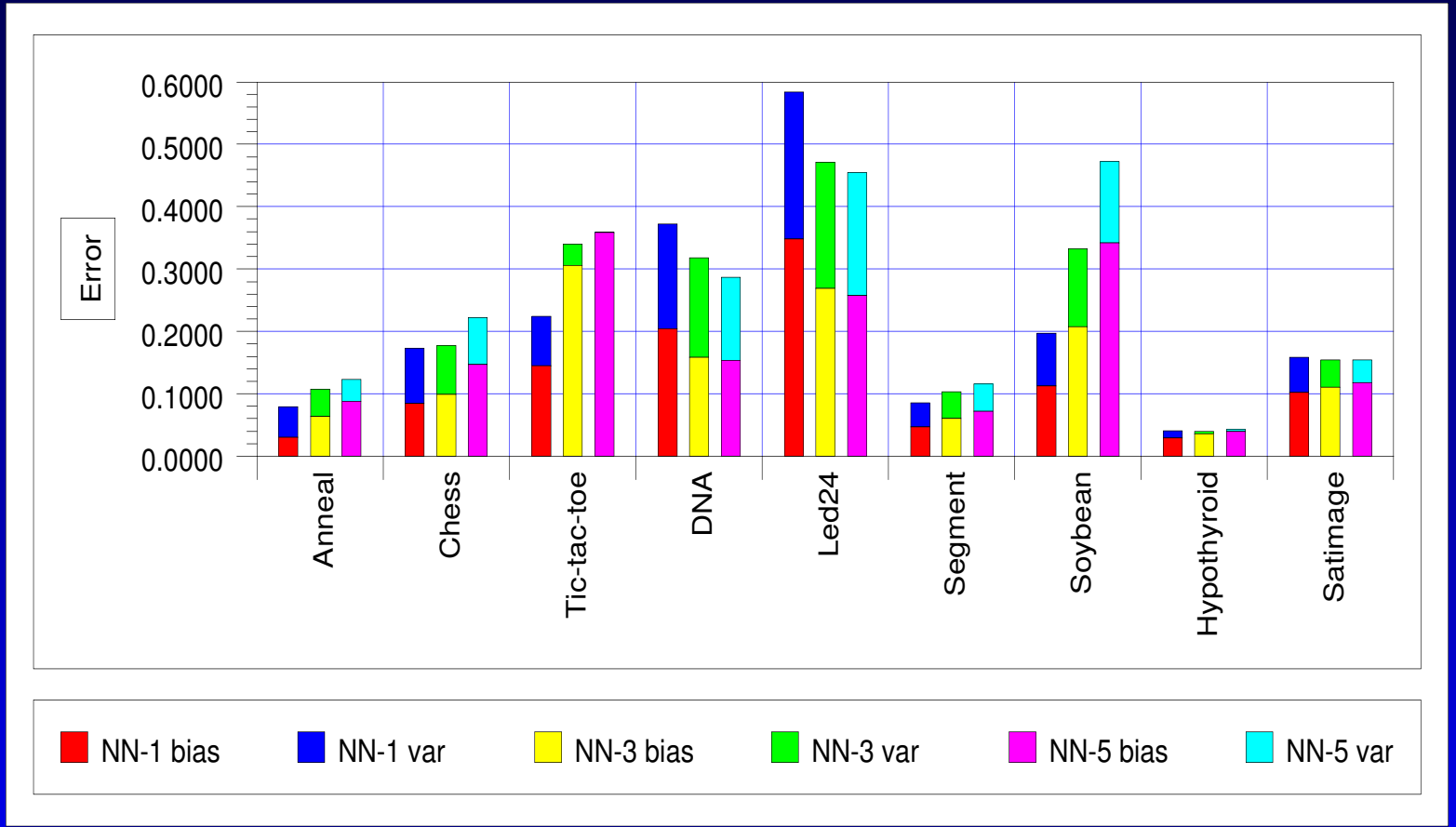
Note how the estimate is unbiased

Unbiased estimator of variance for binomial distribution it the correction term

$$b*(b-1)/(n-1) = var[b] = E[b^2] - (E[b])^2$$

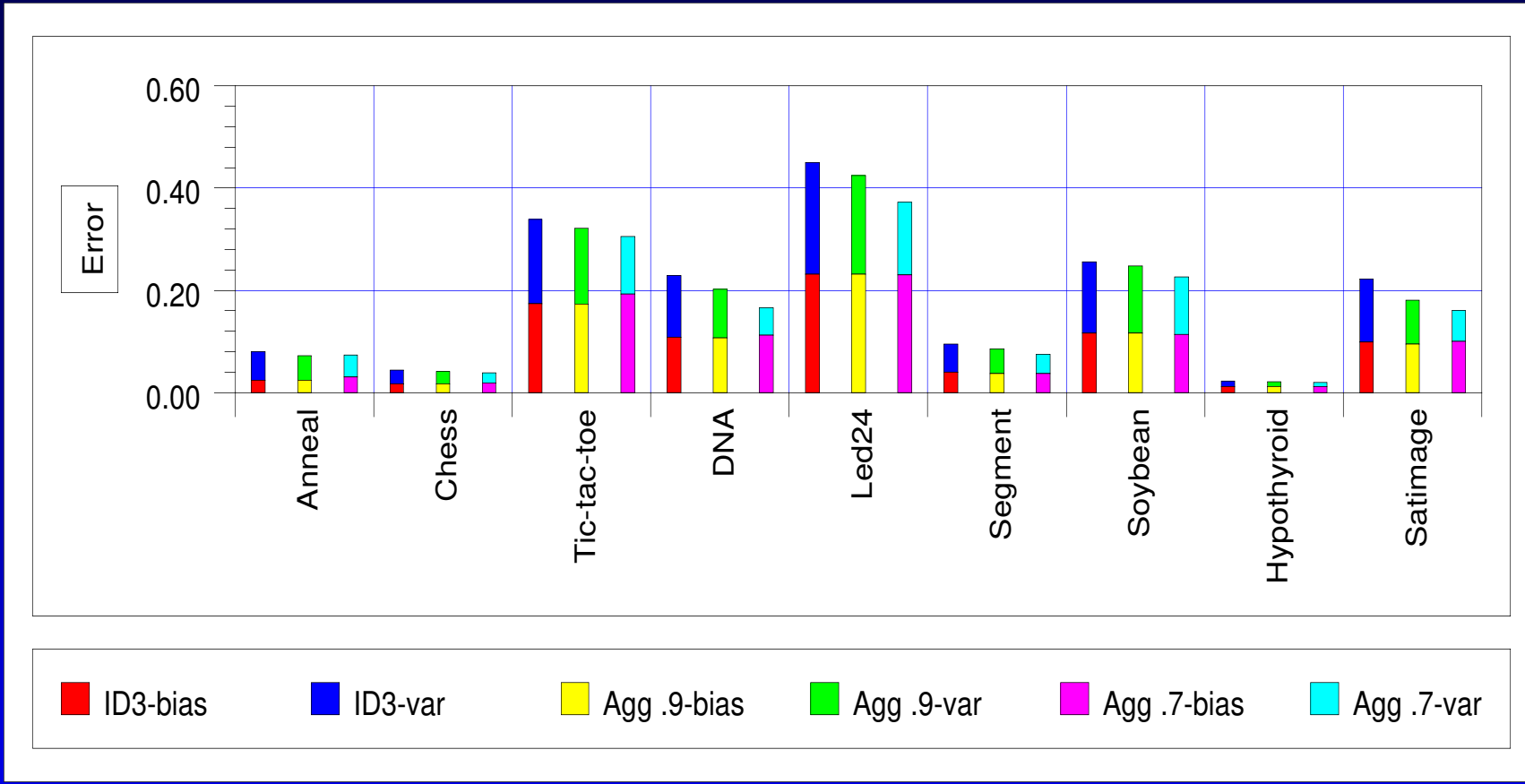
"Real" (UC Irvine) datasets

Guessing what's best for real datasets is hard. All combinations are possible.



Combining Classifiers

Combining classifiers by voting reduces variance (aggregation, ensembles, stacked classifiers).



Conclusions

- The bias–variance decomposition is useful in analyzing variants of algorithms and attributing changes to bias/variance or both.
- Obvious, natural, and exhaustive methods are not so useful because they increase variance.
- "Technical" problem in estimating the bias and variance required a binominal correction.
- The code for the decomposition exists in MLC++ and can be used with any induction algorithm implemented in MLC++.

The Basic Observations

$P(y_H \mid f, m, x)$ is an average over all possible datasets of size m:

$$P(y_H \mid f, m, x) = \sum_d P(d \mid f, m, x) P(y_H \mid d, f, m, x) \quad (1)$$

$$= \underbrace{\sum_d P(d \mid f, m)}_{\text{probability of a dataset}} \underbrace{P(y_H \mid d, x)}_{\text{classifier's prediction} = y_H?} \quad (2)$$

Proposition: Y_F and Y_H are conditionally independent given the target f and and a test point x.

$$P(y_F, y_H \mid f, x) = P(y_F \mid y_H, f, x) P(y_H \mid f, x) \quad (3)$$

$$= P(y_F \mid f, x) P(y_H \mid f, x) \ . \quad (4)$$

Relation to Quadratic B+V

1. Let $\vec{Y_F}$ be an $\mathbb{R}^{|Y|}$-valued random variable restricted so that exactly one of its components equals 1 and all others equal 0. The single 1-valued component is the one with index y_F.
2. Define $\vec{Y_H}$ similarly in terms of Y_H.
3. $\vec{Y_F}$ and $\vec{Y_H}$ are real-valued so we can define quadratic loss:

$$(\vec{y_F} - \vec{y_H})^2 = \begin{cases} 2 & \text{if } y_F \neq y_H \\ 0 & \text{if } y_F = y_H \end{cases}$$

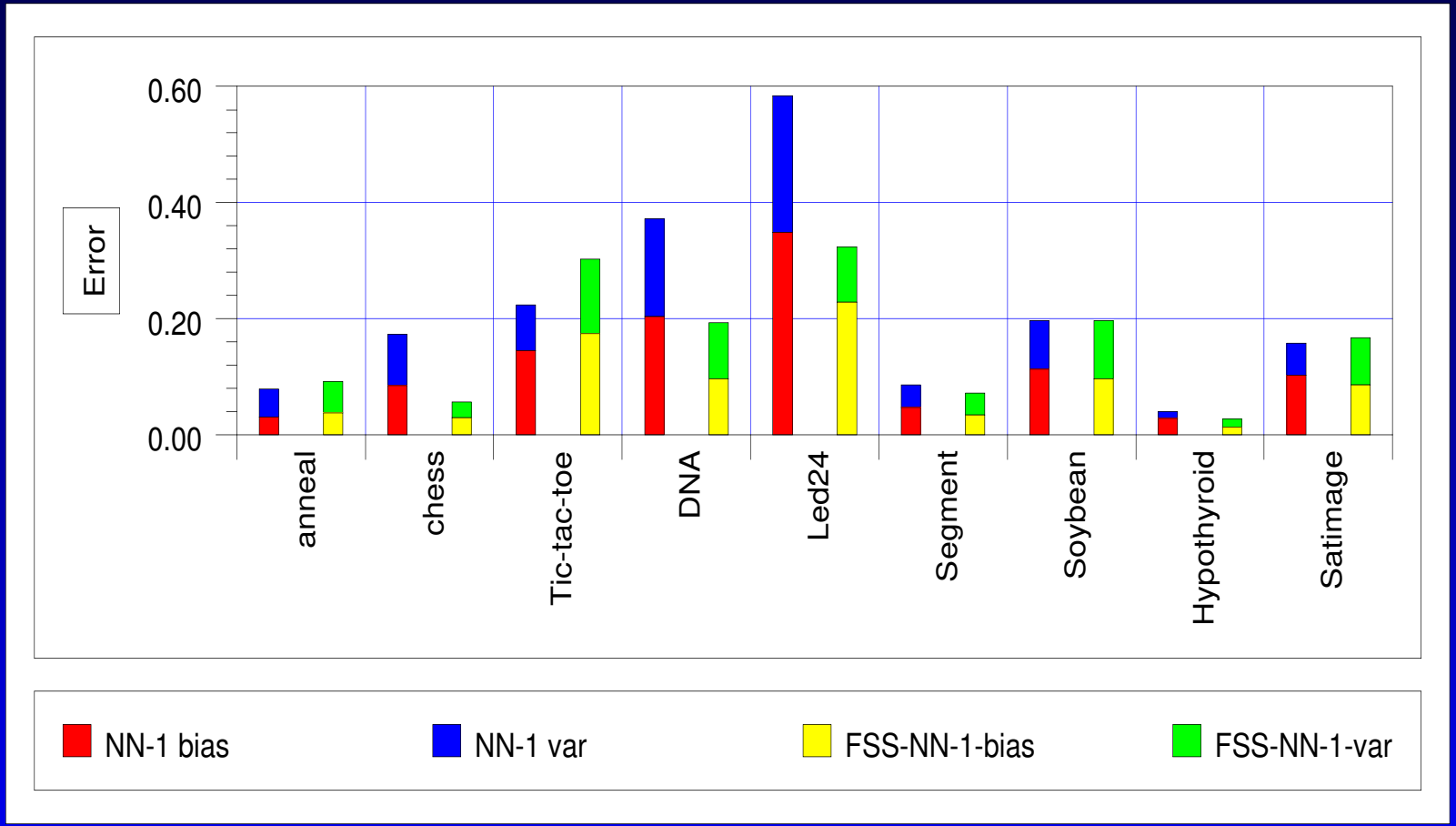
4. This leads to a simple relation to quadratic loss:

$$E\left[\left(\vec{Y_F} - \vec{Y_H}\right)^2 \mid f, m, x\right] = 2\,P(y_H \neq y_F \mid f, m, x) \quad (1)$$

$$= 2\,E(C_{\text{0-1 loss}} \mid f, m, x)\ . \quad (2)$$

Feature Subset Selection

FSS (using wrapper approach) can be a big win for both bias and Variance.



Multiple Weights

A natural extension to feature subset selection is weighting: allow multiple weights. But... this increases the variance