Machine learning methodology: Overfitting, regularization, and all that

CS194-10 Fall 2011

Outline

- ♢ Measuring learning performance
- ♢ Overfitting
- ♢ Regularization
- ♢ Cross-validation
- ♢ Feature selection

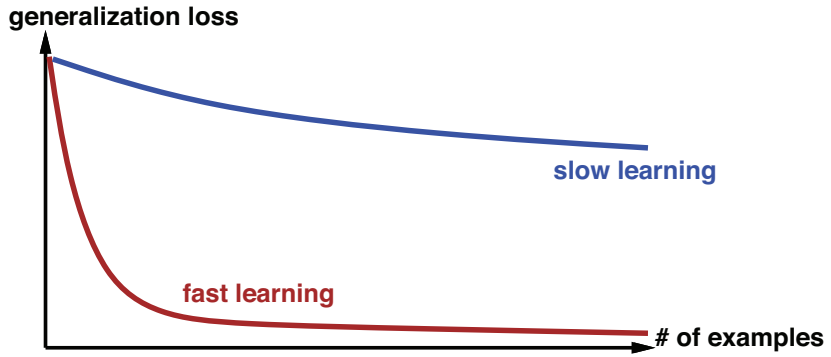
Performance measurement

We care about how well the learned function h generalizes to new data:

$$GenLoss_L(h) = E_{x,y}L(x, y, h(x))$$

Estimate using a test set of examples drawn from
same distribution over example space as training set

Learning curve = loss on test set as a function of training set size
(often averaged over many trials)

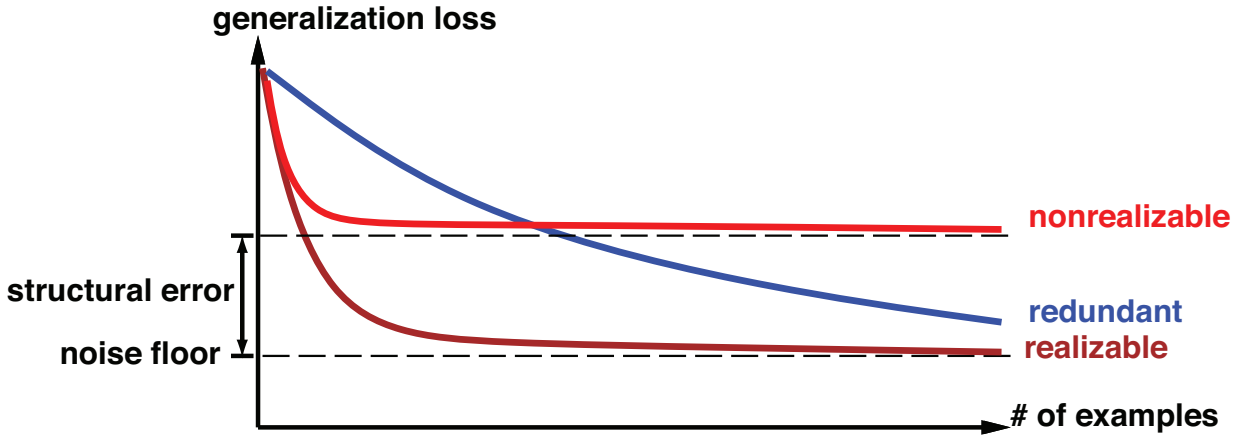


This is a way of evaluating learning *algorithms*

Performance measurement contd.

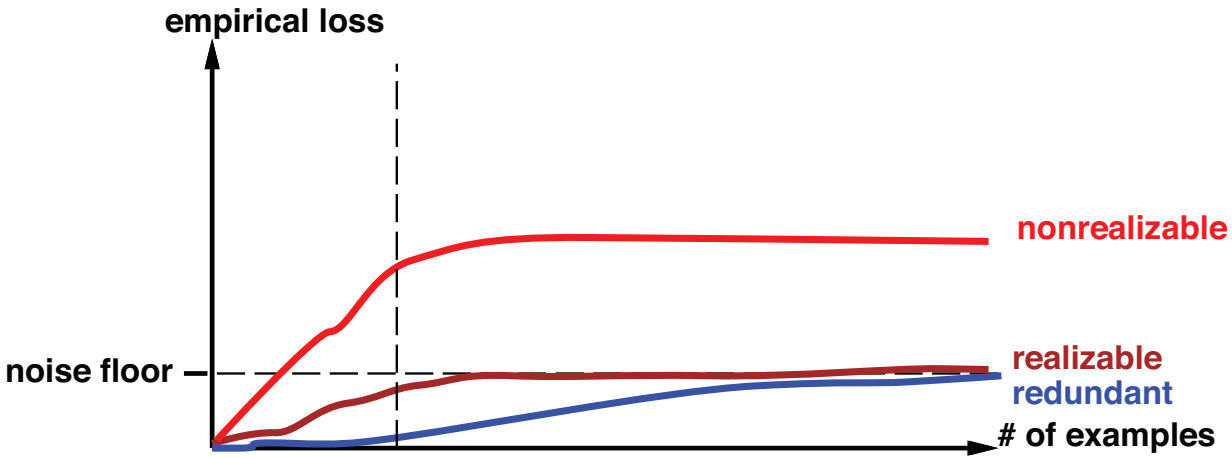
E.g., suppose data generated by quadratic + noise:

- Quadratic h is the realizable case (can express true f up to noise); learns quickly, reaches noise floor
- Linear h is the non-realizable case (restricted H or missing inputs); suffers additional structural error
- High-degree polynomial h: realizable but redundant; learns slowly



Training error

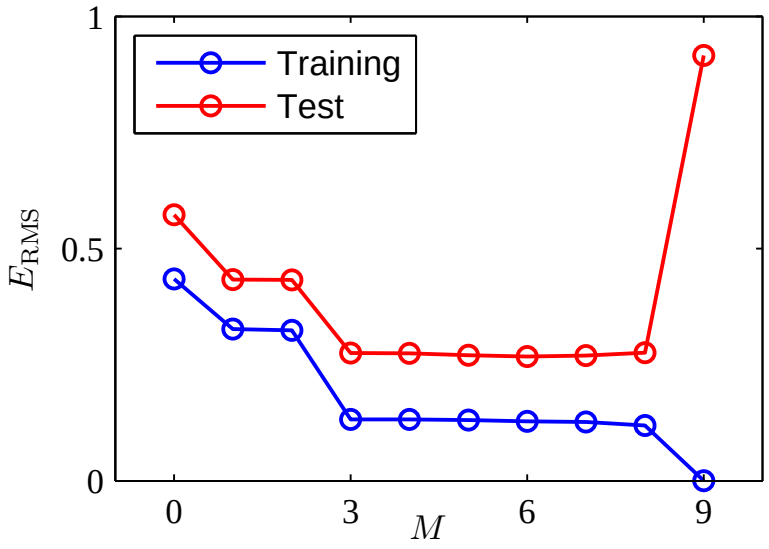
During learning, we have access to training error (empirical loss); things may look quite different given fixed training set:



Overfitting

Fix the training set size, vary H complexity (e.g., degree of polynomials)

Example from Bishop, Figure 1.5



For any given N, some h of sufficient complexity fits the data
but may have very bad generalization error!!

Regularization

Reduces overfitting by adding a complexity penalty to the loss function

L_2 regularization: complexity = sum of squares of weights
Combine with L_2 loss to get ridge regression:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

where $\lambda \geq 0$ is a fixed multiplier and $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{D} w_j^2$

w_0 **not** penalized, otherwise regularization effect depends on y-origin

L_2 Regularization Solution

First "center" the data:

- Fix $w_0 = \overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$
- Drop dummy x_0 from data matrix $\mathbf{X}$ and set $x'_{ij} = x_{ij} - \overline{x_j}$

Now can write

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

Derivative with respect to $\mathbf{w}$ is

$$\underbrace{-2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}}_{\text{as before}} + 2\lambda\mathbf{w}$$

and setting this to zero gives

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

MAP (maximum a posteriori) interpretation

General MAP learning:

$$\begin{aligned}\hat{h} &= \arg\max_h P(\text{data} \mid h) P(h) \\ &= \arg\min_h \underbrace{-\log P(\text{data} \mid h)}_{\text{as in MLE}} \underbrace{-\log P(h)}_{\text{complexity penalty}}\end{aligned}$$

For regression, suppose we think weights are *a priori* independent and (except for w_0) probably small:

$$P(h_{\mathbf{w}}) = \prod_{j=1}^{D} N(w_j \mid 0, \rho^2) = \alpha_\rho^D e^{-\sum_j w_j^2/2\rho^2} = \alpha_\rho^D e^{\mathbf{w}^T\mathbf{w}/2\rho^2}$$

MAP interpretation contd.

Assuming Gaussian regression model with variance σ^2, MAP formula is

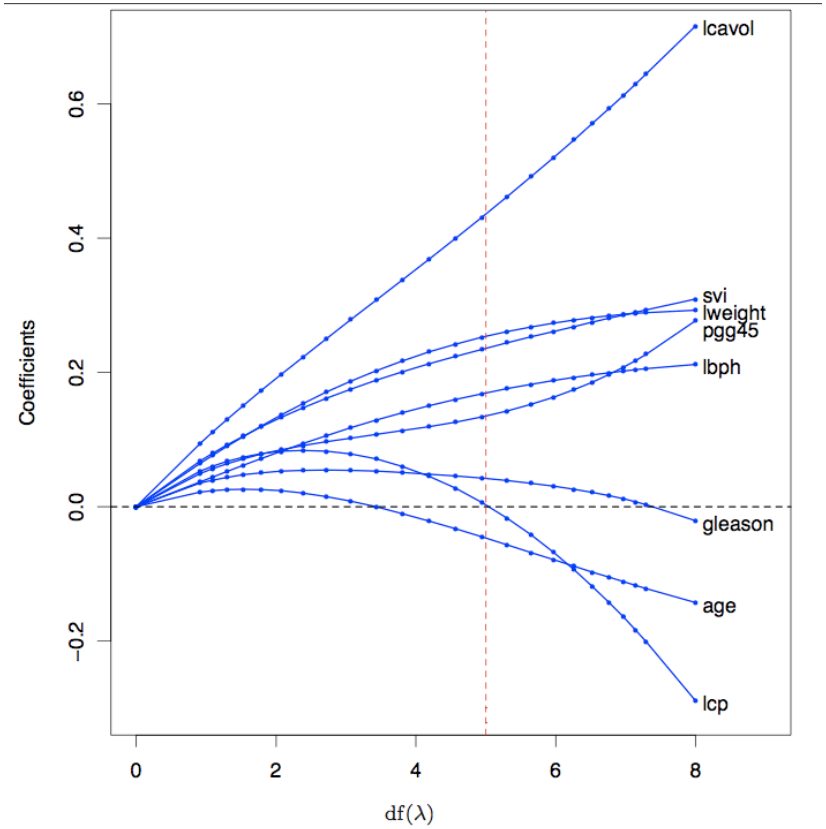
$$\begin{aligned}\hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \left(\frac{1}{2\sigma^2}(\mathbf{Y}-\mathbf{X}\mathbf{w})^T(\mathbf{Y}-\mathbf{X}\mathbf{w}) - N\alpha_\sigma \right) + \left(\frac{1}{\rho^2}\mathbf{w}^T\mathbf{w} - D\alpha_\rho \right) \\ &= \arg\min_{\mathbf{w}} (\mathbf{Y}-\mathbf{X}\mathbf{w})^T(\mathbf{Y}-\mathbf{X}\mathbf{w}) + \frac{\sigma^2}{\rho^2}\mathbf{w}^T\mathbf{w}\end{aligned}$$

which is identical to L_2 regularization with $\lambda = \sigma^2/\rho^2$

Effect of L_2 regularization

As λ **increases**, $\mathbf{w}^T\mathbf{w}$ **decreases**

Example from Hastie, Fig 3.8 (scale is inverse of λ):



L_1 regularization (LASSO)

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

where $\lambda \geq 0$ and $\|\mathbf{w}\|_1 = \sum_{j=1}^{D} |w_j|$

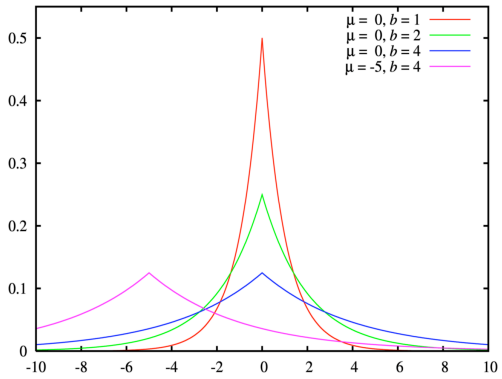
Looks like a small tweak, but makes a big difference!

1) No more closed-form solution
   - use **quadratic programming**
     $\min_{\mathbf{w}} (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w})$ s.t. $\|\mathbf{w}\|_1 \leq s$
   - convex problem, polytime (but expensive) solution

2) LASSO = MAP learning with Laplacian prior

$$P(w_j) = \frac{1}{2b} e^{-\frac{|w_j|}{b}}$$

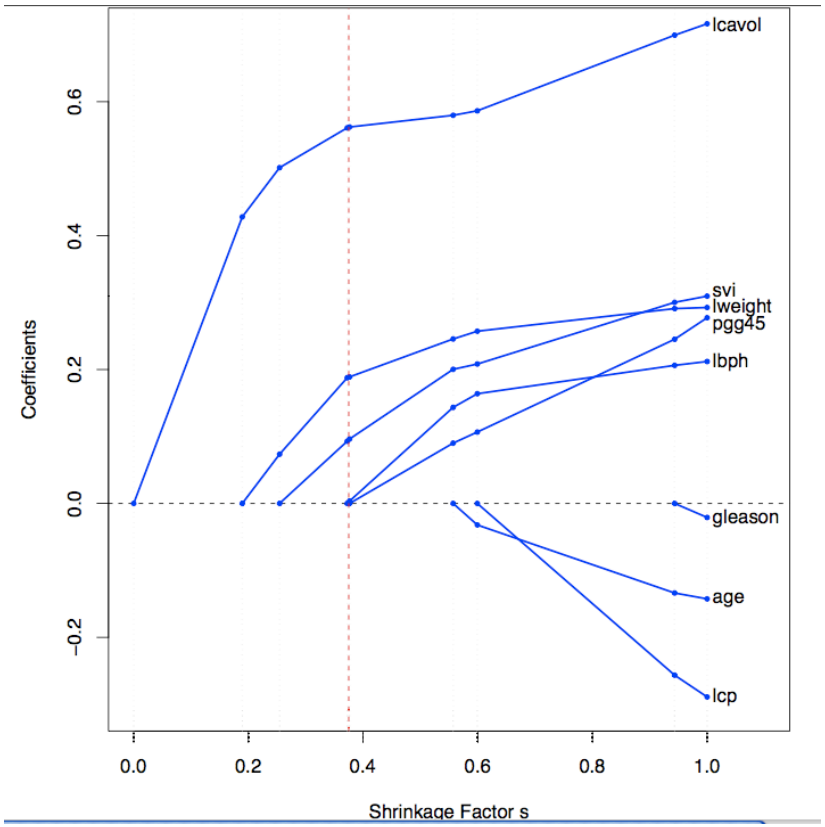
where b is the scale



Effect of L_1 regularization

Laplace prior encourages sparsity, i.e., mostly zero weights

Example from Hastie *et al.*, Fig 3.10:



Cross-validation

Regularization helps but still need to pick λ.
Want to minimize test-set error, but we have no test set!

Idea: make one (a validation set) by pretending we can't see the labels

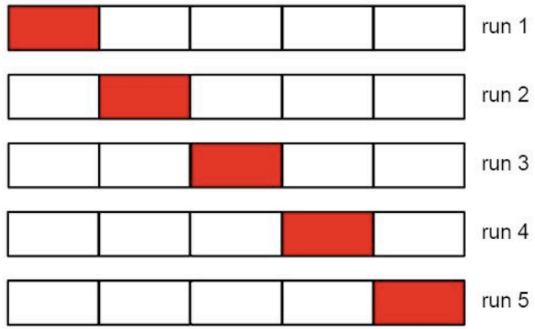
Try different values of λ, learn $\hat{h}^\lambda$ on rest of data,
test $\hat{h}^\lambda$ on validation set, pick best λ, train on all

Problem: small validation set $\Rightarrow$ large error in estimated loss
large validation set $\Rightarrow$ small training set $\Rightarrow$ bad $\hat{h}^\lambda$

Cross-validation contd.

K-fold cross-validation: divide data into K blocks
for $k = 1$ to k
    train on blocks except kth block, test on kth block
average the results, choose best λ

Common cases: $K = 5, 10$ or $K = N$ (LOOCV)



High computation cost: K folds $\times$ many choices of model or λ

Feature selection

Another way to get a sparse predictor: pick out a small set of the most relevant features

A set of features $F \subseteq \{X_1, \ldots, X_D\}$ is minimally relevant
if there is some h definable using F such that
1) no h' defined on a superset of F has lower generalization loss
2) any h'' defined on a subset of F has higher generalization loss

Any feature not in a minimally relevant set is irrelevant

Problems in choosing a minimally relevant set:
– inaccurate estimate of generalization loss
$\Rightarrow$ some features *appear* relevant when they're not
– NP-hard to find a set even with perfect estimates

Forward selection: greedily add feature that decreases CV error most
Backward selection: greedily delete feature that decreases CV error most

Summary

Learning performance = prediction accuracy measured on test set

Trading off complexity and degree of fit is hard

Regularization penalizes hypothesis complexity
- L_2 regularization leads to small weights
- L_1 regularization leads to many zero weights (sparsity)

Feature selection tries to discard irrelevant features

Cross-validation enables selection of feature sets or regularization penalties by estimating test-set error on parts of the training set