Applied Machine Learning in Biomedicine

Enrico Grisan
enrico.grisan@dei.unipd.it
Algorithm’s objective cost

Formal objective for algorithms:
- minimize a cost function
- maximize an objective function

Proving convergence:
- does objective monotonically improve?

Considering alternatives:
- does another algorithm score better?
Loss function

We want to specify the objective of an algorithm

One idea: consider a loss function \( L(\hat{y}(x_*); y_*) \)

Would like to minimize loss at test time

Minimizing empirical loss might be a reasonable proxy:

\[
\sum_i L(\hat{y}(x_i); y_i)
\]
Choosing a loss function

• **Motivated by the application**
  – 0-1 error, achieving a tolerance, business cost

• **Computational convenience:**
  – Differentiability, convexity

• **Beware of loss dominated by artifacts:**
  – Outliers
  – Unbalanced classes
A step into linear regression

Find a linear function

\[ \hat{y} = w^T x \]

That approximates the mapping:

\[ x \rightarrow y \]
A step into linear regression

Find a linear function

\[ \hat{y} = \mathbf{w}^T \mathbf{x} \]

That minimizes the sum of squared residuals from \( y \):

\[
RSS(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
\]

\[
= \sum_{i=1}^{N} \left( y_i - b - \sum_{j=1}^{p} x_{ij}w_j \right)^2
\]
Vector form for RSS

\[ x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix} \]

\[ x_i = \begin{bmatrix} 1 \\ \text{age} \\ \text{BMI} \\ \vdots \\ \text{glycemia} \end{bmatrix} \]

\[ X = \begin{bmatrix}
1 & x_{11} & \cdots & x_{1D} \\
1 & x_{21} & \cdots & x_{2D} \\
1 & x_{31} & \cdots & x_{3D} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_{N1} & \cdots & x_{ND}
\end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \\ \vdots \\ x_N^T \end{bmatrix} \]

\[ w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \]
Least squares estimation

$$RSS(w) = (y - Xw)^T(y - Xw)$$

$$\frac{\partial RSS}{\partial w} = -2X^T(y - Xw)$$

$$\frac{\partial^2 RSS}{\partial w \partial w^T} = 2X^TX$$

$$-2X^T(y - Xw) = 0$$

$$\hat{w} = (X^TX)^{-1}X^Ty$$
Geometry of least squares

\[ \hat{y} = \hat{w}^T x = x^T (X^T X)^{-1} X^T y \]

The columns of $X$ span a subspace of $\mathbb{R}^{D+1}$

The closest point to $y$ in this subspace is its orthogonal projection

The orthogonal projection is given by the dot product

\[ y \approx \hat{y} = X (X^T X)^{-1} X^T y \]
Least square estimation

\[ Y = Xw \]

% \( w = D \times 1 \) weights
% \( X = N \times D \) test cases
% \( Y = N \times 1 \)

\( w = X \backslash Y \);
Least square estimation (2)

If we want to minimize the RSS

$$L(w^T x; y) = (Xw - Y)^T (Xw - Y)$$

we can use the iterative scheme with Newton update:

$$w^{\tau+1} = w^\tau - \eta \nabla L(w^T x; y)$$

$$\nabla L(w^T x; y) = \sum_{i=1}^{N} (w^T x_i - y_i)x_i = X^TXw - X^TY$$
Least square estimation (2)

1. Initialize $w^0$

2. Update
   
   $w^{\tau+1} = w^\tau - \eta X^T(Xw^\tau - Y) = w^\tau - \eta X^T(\hat{Y} - Y)$

2. Check termination condition
   
   a) $w^{\tau+1} = w^\tau$
   b) $|w^{\tau+1} - w^\tau| < \varepsilon$
   c) $\tau > T$
   d) $\max(\nabla L) < \varepsilon$
The importance of the step
Least squares classifier

Why not using linear least squares to fit regressors on binary targets?

```matlab
% fit yy = ww*xx
% ww = Dx1 weights
% xx  = NxD test cases
% yy  = Nx1

ww = xx\yy;
```
Least squares classifier

\[ y < 0 \quad y > 0 \]
Least squares classifier
Least squares classifier

Why not using linear least squares to fit regressors on binary targets?

\[ \text{fit } yy = ww^*xx \]
\[ ww = D \times 1 \text{ weights} \]
\[ xx = N \times D \text{ test cases} \]
\[ yy = N \times 1 \]

\[ ww = xx \backslash yy; \]
Least squares in practice (1)

Prostate cancer study (Stamey, 1989)

Mapping clinical measure with PSA marker
1) Normalize all data

\[ x_i = \begin{bmatrix} 1 \\ \log \text{ca vol} \\ \log \text{weight} \\ \text{age} \\ \log \text{ph} \\ \text{svi} \\ \log \text{cp} \\ \text{gleason} \\ \text{pgg45} \end{bmatrix} \rightarrow x'_i = \begin{bmatrix} 1 \\ \log \text{ca vol}/\sigma_1 \\ \log \text{weight}/\sigma_2 \\ \text{age}/\sigma_3 \\ \log \text{ph}/\sigma_4 \\ \text{svi}/\sigma_5 \\ \log \text{cp}/\sigma_6 \\ \text{gleason}/\sigma_7 \\ \text{pgg45}/\sigma_8 \end{bmatrix} \]
Least squares in practice (3)

2) Split the data into train and test set
3) Fit the regression on the training set
4) Estimate results on the test set

\[
\begin{align*}
X_{train} &= \begin{bmatrix}
x'_1^T \\
x'_2^T \\
x'_3^T \\
\vdots \\
x'_N^T 
\end{bmatrix} \\
\hat{\mathbf{w}} &= (X_{train}^T X_{train})^{-1} X_{train}^T \mathbf{y} \\
\hat{Y}_{test} &= X_{test} \hat{\mathbf{w}}
\end{align*}
\]
5) Evaluate the mean error

\[ RSS_{test} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \]
6) Evaluate the features importance

\[
\Sigma_w = Var(w) = (X^T X)^{-1} \sigma^2
\]

\[
\hat{\sigma}^2 = \frac{1}{N - (D + 1) - 1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
\]
Least squares in practice (6b)

Features’ z-score

\[ z_j = \frac{\hat{w}_j}{\hat{\sigma} \sqrt{\Sigma_{w,jj}}} \]
Multiclass linear classifier

One versus all

K class, K>2

Building K-1 binary classifiers
Multiclass linear classifier

One versus one

K class, K>2

Building $K(K-1)/2$ binary classifiers
Multiclass linear classifier

Least squares approach

$Y$ is an $N \times K$ indicator label matrix

$W$ is a $D \times K$ weight matrix

$X$ is a $N \times D$ data matrix

\[
\hat{W} = (X^T X)^{-1} X^T Y
\]

\[
y(x) = \hat{W}^T x
\]

\[
\sum_{k=1}^{K} y_k (x) = 1
\]
Linear regression (with features)

\[ X = \left[ \text{ones}(N,1), \ xx, \ xx.^2, \ xx.^3, \ xx.^4, \ xx.^5, \ xx.^6 \right]; \]
\[ W = X \backslash \text{yy}; \]

\text{FunctionInterpolation}
\[ X_{\text{new}} = \left[ \text{ones}(N,1), \ x_{\text{new}}, \ x_{\text{new}}.^2, \ x_{\text{new}}.^3, \ x_{\text{new}}.^4, \ x_{\text{new}}.^5, \ x_{\text{new}}.^6 \right]; \]
\[ Y_{\text{new}} = X_{\text{new}} \times W; \]
Neighbour-based regression

Take height from the nearest input
Kernel smoothing

Weight points in proportion to a \textit{kernel}

\[ \hat{y}_* = \sum_i w_i x_i \]

\[ w_i = \frac{k(x_*, x_i)}{\sum_j k(x_*, x_j)} \]
Kernel smoothing
Overfitting

We can make the empirical loss zero:
Generalization

Want to do well on future, unknown data:
Expected error

Training error

\[ \overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) \]

Generalization error

\[ Err_T = E[L(Y, \hat{f}(X))|\mathcal{T}] \]

Expected test error

\[ Err = E[L(Y, \hat{f}(X))] = E[Err_T] \]
Validation set

\[ x' = [1 \ x \ x^2 \ \cdots \ x^p] \]

\[ y = w^T x' \]
Learning curves

Mean squared error vs. polynomial order

- Validation
- Train
Using validation set

Validation set used to estimate test error for fitted model

Can overfit the validation set

Tracking a validation set is also used during fitting of a single model
  - ad hoc
  - depends on optimizer
  - sometimes fast
  - sometimes can work annoyingly well
Model selection and assessment

**Model selection:**
Estimating the performance of different models in order to choose the best one

**Model assessment:**
having chosen a final model, estimating its prediction error on new data
Cross validation

We want a procedure for estimating at least the average generalization error

\[ Err = E[L(Y, \hat{f}(X))] = E[Err_T] \]

even when the data is scarce and we cannot set aside a validation set

Data

<table>
<thead>
<tr>
<th>Fold 1 - Train</th>
<th>Fold 2 - Train</th>
<th>Fold 3 - Train</th>
<th>Fold 4 - Train</th>
<th>Fold 5 - Test</th>
<th>Fold 1 - Train</th>
<th>Fold 2 - Train</th>
<th>Fold 3 - Test</th>
<th>Fold 4 - Train</th>
</tr>
</thead>
</table>

K-Fold Cross Validation

1) Split the data into K parts
2) Set aside the $k^{th}$ part and fit the model to the other K-1 parts of the data
3) Repeat for each $k = 1, \ldots, K$
4) Combine the K estimates of prediction error

$$Err_{CV} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^k(i)(x_i))$$
Choosing K

K=N  leave one out
Unbiased estimator for the prediction error
High variance (training set are very similar)

Small-sized training set may overestimate the prediction error

Rule of thumb:
K=5 or K=10
Cross validation scenario

Large number of predictors $D \gg N$

Typical strategy
1) Screen the predictors: find a subset of promising predictors $D' \ll D$ showing some correlation with the class labels

2) Using the subset, building a classifier $y = \hat{f}(x')$

3) Use cross validation to estimate tuning parameters and the prediction error $Err_{CV}$
Cross validation scenario

Large number of predictors $D \gg N$

Correct strategy
1) Divide the data into $K$ folds

2) For each fold:
   a) Screen the predictors using all samples except those in fold $k$
   b) Using the subset, building a classifier $y = \hat{f}(x')$ using all samples except those in fold $k$
   c) Predict the class labels using the samples in fold $k$

3) Estimate the prediction error $Err_{CV}$
P1. Cleveland Heart Disease

Data from V.A. Medical Center, Long Beach and Cleveland Clinic Foundation, 1988.

297 patients, 14 attributes per patient

Predict heart disease (possibly in a scale 1-4)

$$1 - \overline{err} = 0.85$$
P2. HIV cleavage site

Knowledge of the mechanism of HIV protease cleavage specificity is critical to the design of specific and effective HIV inhibitors. Searching for an accurate, robust, and rapid method to correctly predict the cleavage sites in proteins is crucial when searching for possible HIV inhibitors.

Scope is to predict if a sequence of aminoacids will constitute a cleavage site.