Classification in BCI

J.Farquhar

Donders Institute for Brain, Cognition and Behaviour, Radboud University Nijmegen

Signal Processing for Brain Computer Interfaces,
Nijmegen, Jan 2011

Outline

Why we need Classifiers

Introduction to classification
Linear Models

Linear Regularized Least-Squares Regression

Least-Squares Regression
Regularization
Learning Objective Function

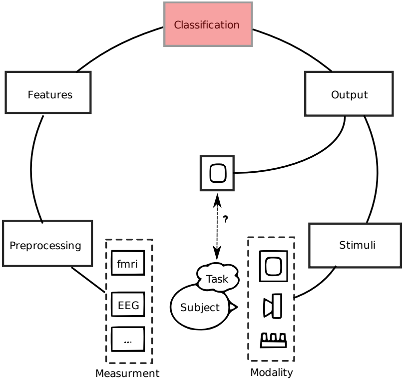
Linear Classification

Linear Classification

Classification Tips & Tricks

Data Hygiene
Advanced Methods
Summary

Where we are in the BCI cycle



The need for classifiers

Just to re-iterate

Goal of BCI

Develop systems which completely paralyzed people (such as late-stage sufferers of Amyotropic Lateral Sclerosis, ALS) could use to communicate,

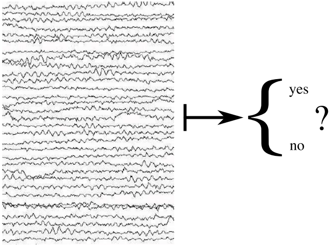
Note the emphasis on the brain!
That means we can't use:

- muscles
- peripheral nerves, e.g. motor neurons
- machine artifacts

but

Instead we have to use the recorded neural activity.

Decoding the neural code



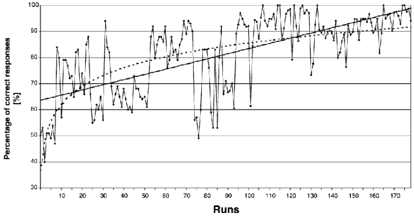
Unfortunately,

- In general, we don't know how the users intentions are encoded in the neural signal
- we don't know how the signal may be encoded for this individual

Decoding Options

Thus we have two options:

- train the user to generate a strong known signal, e.g. SCP, μERDS
- train the machine to extract this users signal..



We use machine learning and in particular classification to automatically decode the neural code for each new subject.

The utility of Machine Learning in BCI

Get results quickly

- Shift the burden of learning off the patient and onto the computer.
- Minutes/Hours to recognize relevant features, rather than weeks/months training a patient to modulate pre-specified features.

Let the system run itself.

- No intervention from experts.

(Your) Learning Goals

- Explain the terms: model-type, feature space, loss-function, regularization function
- Understand the peril of over-fitting and ways of avoiding it.
- Be able to write down a generic classifiers objective function and describe the purpose of each term in it
- Discuss the differences between regression and classification and their implications.
- Explain the necessity for cross-validation
- Outline the basic classifier training methodology and its main steps

Warning

(simple?) Equations and Linear-Algebra Imminent

Outline

Why we need Classifiers
- Introduction to classification
- Linear Models

Linear Regularized Least-Squares Regression
- Least-Squares Regression
- Regularization
- Learning Objective Function

Linear Classification
- Linear Classification

Classification Tips & Tricks
- Data Hygiene
- Advanced Methods
- Summary

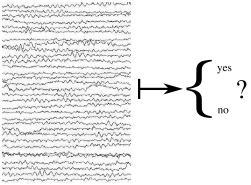
Overview of the lecture

- Only talk about binary-linear classification
- Start with linear-regression
- Discuss loss-functions for regression
- Discuss the problems of outlier-sensitivity and two methods for addressing it
- Discuss various interpretations of the purpose/effect of adding regularization to the objective function
- Discuss the implications of moving from regression to classification
- Move from regularized-least-squares-regression (rLSR) to regularised-least-squares-classification (rLSC)
- Move from rLSC to regularized logistic regression (rLR)

The Prediction Problem

Prediction Problem (I)

Given x what is the best guess for y?



- x is the independent variable, (what we measure)
- y is the dependent variable, (what we want to predict)

Solve this problem learning a decision function, f which maps from x to y. i.e.

$$f : x \mapsto y \implies f(x) = \hat{y}$$

Prediction Problem (II)

Given a training set consisting of pairs of $\{x_1, x_2, ...\}$ and $\{y_1, y_2, ...\}$, how do we learn a decision function f which best predicts y from x.

Generalization

- at one level the prediction problem is trivial
- just store the training set, but..
- ..then what do we predict for a new unseen x?

We also require our decision function to generalize to predict well on new unseen data not in the training set

Prediction Problem (III)

Given a training set $\{(x_1, y_1), (x_2, y_2), ...\}$ learn a decision function f which best predicts y from x on a new unseen testing set.

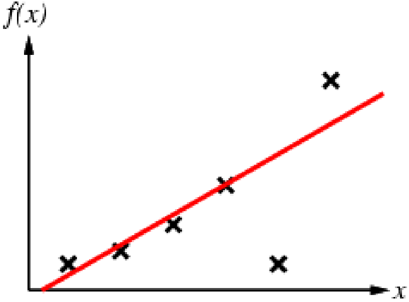
- because f represents the important parts of the data we say f models the data

Model-Types

- for any data-set there are an infinite number of possible models which "fit" the data
- some are very simple, (constant,linear)
- some are less so, (quadratic,cubic)
- some are very complex, ("Neural-Network")

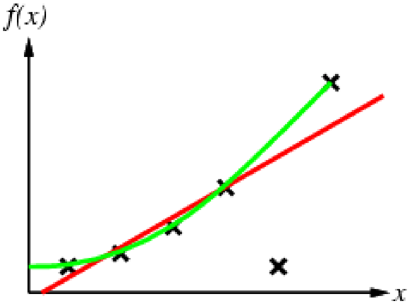
Model-Types

- for any data-set there are an infinite number of possible models which “fit” the data
- some are very simple, (constant,linear)
- some are less so, (quadratic,cubic)
- some are very complex, (“Neural-Network”)



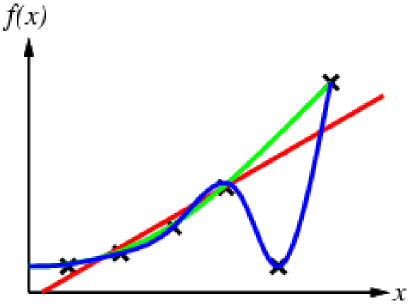
Model-Types

- for any data-set there are an infinite number of possible models which "fit" the data
- some are very simple, (constant,linear)
- some are less so, (quadratic,cubic)
- some are very complex, ("Neural-Network")



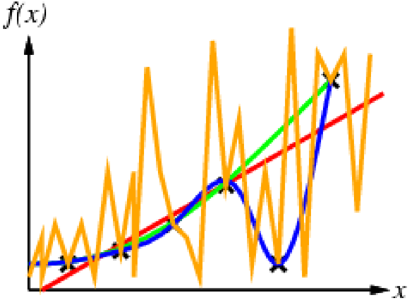
Model-Types

- for any data-set there are an infinite number of possible models which “fit” the data
- some are very simple, (constant,linear)
- some are less so, (quadratic,cubic)
- some are very complex, (“Neural-Network”)



Model-Types

- for any data-set there are an infinite number of possible models which “fit” the data
- some are very simple, (constant,linear)
- some are less so, (quadratic,cubic)
- some are very complex, (“Neural-Network”)



Model-Selection Problem

Part of the classification problem is to pick which type of model best suits the data

Types of Model

It turns out that:

Many complex models are equivalent to linear models in a transformed feature space.

For Example,

- Quadratic model $\Leftrightarrow$ linear model in feature space of all pairwise and linear products of features.

Thus

Only discuss linear models here

Types of Prediction problem

There are many types of prediction problem, depending on the type of y we must predict:

- Vector: $y \in \mathbb{R}^N$, can be a point in N dimensional space, e.g. age+weight. Commonly called multiple-regression.
- Numeric: $y \in \mathbb{R}$, can be any real number, e.g. height, weight. Commonly called regression
- Categorical: $y \in \{A, B, C...\}$, one of a set of possible values. e.g. LH, RH, FT. Commonly called multi-class classification
- Binary: $y \in \{-1, +1\}$, one of two possible values, e.g. LH, RH. Commonly called binary classification.

As any multi-class can be turned into a set of binary problems:

We only discuss Linear Binary classification here.

Outline

Why we need Classifiers

- Introduction to classification
- Linear Models

Linear Regularized Least-Squares Regression

- Least-Squares Regression
- Regularization
- Learning Objective Function

Linear Classification

- Linear Classification

Classification Tips & Tricks

- Data Hygiene
- Advanced Methods
- Summary

The Linear Model

1-d version

$$y = mx + c \qquad (1)$$

n-d version

$$f(x) = w^\top x + b \qquad (2)$$

Note

The decision function f is uniquely given by the numbers, $\mathbf{w}, b$. These numbers are the parameters of the model which we must find.

Properties of the Linear Model

The Linear model has some interesting properties to note:

- Predictions lie on a straight line
- like any vector w can be decomposed into two parts:
  1. its direction, $\tilde{w}$
  2. its magnitude, $|w|$
- thus we have: $f(x) = |w|(\tilde{w}^\top x) + b$
- so $f(x)$ maps from x to predicted y's in two steps
  1. $\tilde{w}^\top x$ maps from x to the closest point on the (1-D) line w,
  2. like the 1-D case, $|w|(\tilde{w}^\top x) + b$ scales and shifts this point such that the distance from the origin gives the prediction
- whole sets of points get given the same prediction ...
- ... these sets are perpendicular to the line w,
- ... and form, lines (1-D), planes (2-D), ... hyper-planes (n-d)
- Thus, this model ignores any properties of x orthogonal to the line w

Visualizing linear models

- linear-model has 1 weight value per feature value, thus
- ... we can visualize it as if it was a normal data sample, x
- magnitude of the weight tells us about that features importance
- sign of the weight tells us (roughly) if the feature is correlated/anti-correlated with y

Always

Visualize your models just to sanity check them

Outline

Least-Squares Regression

- Regression Problem: given an n-d input, x, predict a real valued y
- Linear Model: decision function model is: $f(x; w, b) = w^\top x + b$

Parameter Estimation Problem

How do we find the parameters of the model w, b given the dataset $\{(x_1, y_1), (x_2, y_2), ...\}$?

Intuitively,

- want the decision function which "fits" the data as well possible, i.e. with the least error

To formalize this we need to define exactly what we mean by least error.
Do this by defining a loss-function, $\mathcal{L}$

Loss functions

Loss Function

Given a true value y and a prediction $\hat{y}$ the loss function gives a number which tells us how much the error costs, i.e. $\mathcal{L} : (\hat{y}, y) \mapsto \mathbb{R}$

Parameter Estimation

Find the best decision function, $f^*(x)$, by minimizing the summed loss on the training set, $\{(x_1, y_1), (x_2, y_2), ...\}$:

$$f^*(x) = \min_{w,b} J(w, b) = \min_{w,b} \sum_i \mathcal{L}(y_i, f(x_i; w, b)) \tag{3}$$

- $J(w, b)$ is called the learning objective function.
- Regression loss functions are 3-d (2-inputs, 1 output) so we can visualize them this way.

Loss functions

Loss Function

Given a true value y and a prediction $\hat{y}$ the loss function gives a number which tells us how much the error costs, i.e. $\mathcal{L} : (\hat{y}, y) \mapsto \mathbb{R}$

Parameter Estimation

Find the best decision function, $f^*(x)$, by minimizing the summed loss on the training set, $\{(x_1, y_1), (x_2, y_2), ...\}$:

$$f^*(x) = \min_{w,b} J(w, b) = \min_{w,b} \sum_i \mathcal{L}(y_i, f(x_i; w, b)) \tag{3}$$

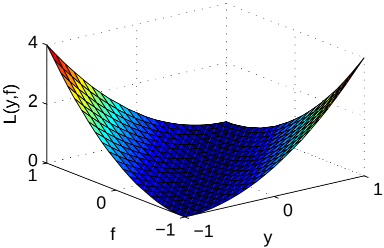
- $J(w, b)$ is called the learning objective function.
- Regression loss functions are 3-d (2-inputs, 1 output) so we can visualize them this way.

Least-Squares Loss Function

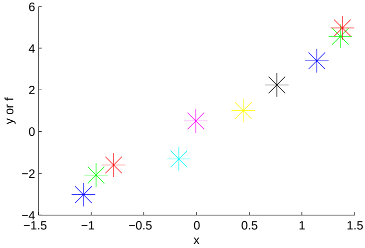
The most common loss function for regression problems is

Squared error loss

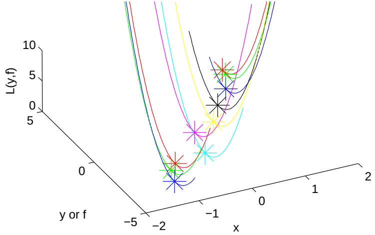
$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$$



1-d toy problem



1-d Toy Problem



Adding the Quadratic Loss

Least-Squares Loss Function

The main reason for its popularity is because, when combined with a linear model, the optimal solution is easy to compute, i.e.

$$\min_{w,b} J_{\mathrm{LS}}(w,b) = \min_{w,b} \sum_i (y_i - (w^\top x_i + b))^2 \tag{4}$$

is

$$w^* = [XX^\top]^{-1} XY^\top = X^\dagger Y \tag{5}$$

as can be seen by; taking derivatives of J_{LS}, setting equal to zero and solving the resulting system of equations
(In Matlab you can fit a LSR model using: `> w=X\Y`)

Outlier Sensitivity

Despite its popularity LS loss function has a major problem:

Outlier Sensitivity

The parameter estimates are determined by a few points distant from the majority of the data, because they generate most of the loss.

This problem is clearly because the LS loss counts large prediction errors much more strongly than small ones.

Over-fitting

A related issue to outlier-sensitivity is:

Over-fitting

Some parameters of the decision function become determined by a few points distant from the majority because only these points depend on these parameters.

Hard to show when models have few parameters...
But if we have many parameters then ...

Much bigger problem when have high-dimensional input, where there are likely to be many irrelevant parameters

Outlier Sensitivity/Over-fitting Solutions

Two main solutions to these problems:

1. Alternative loss functions – which are less sensitive to outliers
2. Regularization – addition cost-function which stops parameters being determined by only a few points

Alternative Loss functions

Many Possible loss functions,

- Squared error (L2, Quadratic), $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$
- Absolute (L1), $\mathcal{L}(y, \hat{y}) = |y - \hat{y}|$,
- Something Weird like, $\mathcal{L}(y, \hat{y}) = y^{2/3} - \hat{y}^{-3}$

As with the model-type picking a loss function depends on what suits the problem best.

And as we will see later

- appropriate loss function choice allows us to treat classification as regression

Outline

Regularization function

Regularization Function

Given a set of decision function parameters, w, the regularization function gives a number which tells us how much those parameters cost, i.e.

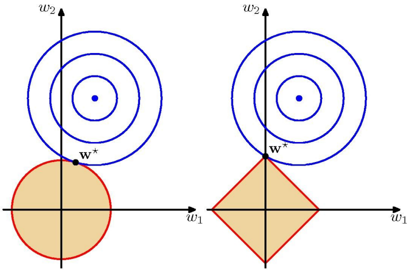
$$\mathcal{R} : w \mapsto \mathbb{R}$$

N.B. picking a regularization function (or its parameters) is another part of the model selection problem!
Few different interpretations of what a regularization function is doing:

- Occam's Razor: solutions should be as simple as possible
- Prior-over-parameters : expresses "preference" for some parameter sets
- Margin-Maximization: only for classification problems, see later.

Types of Regularization Function

- Quadratic: $\mathcal{R}(w) = w^\top w = |w|^2$ (also called L2 regularization)
- L1: $\mathcal{R}(w) = \sum_d |w_d| = |w|_1$ (tends to make sparse)
- L0: $\mathcal{R}(w) = \sum_d (|w_d| > 0)$ (very sparse)



- N.B. all these regularisers treat $w = 0$ as the "preferred" / minimum-complexity parameter values
- In some (rare!) cases we may know better and use a different prefer-ed parameter set.
- e.g. cross-subject learning (see next lecture)

Only talk about *L*2 regularization here.

Outline

Learning Objective Function (key-slide)

Learning Objective Function

Given a training dataset, $\{(x_1, y_1), (x_1, y_2), ...\}$, and a decision function, $f(x; w)$ parametrized by w, the machine learning objective function is;

$$J(w) = \lambda \mathcal{R}(w) + \sum_i \mathcal{L}(y_i, f(x_i; w)) \tag{6}$$

- $f(x; w)$ is the decision function with parameters w
- $\mathcal{L}(y, \hat{y})$ is the loss-function which tells you how much prediction errors cost
- $\mathcal{R}(w)$ is the regularization function which controls how much different parameter settings cost
- λ is the regularization hyper-parameter and controls how many data-points are needed for a parameter be changed by the data
- λ (and any other hyper-parameters) are part of the model-type and picking it is part of the model-selection problem

Linear Regularized Least Squares Regression

$$J_{\mathrm{rLS}}(w) = \lambda w^\top w + \sum_i (y_i - (w^\top x_i + b))^2 \tag{7}$$

- very commonly used
- also called ridge-regression
- can show the optima is given by: $w^* = [XX^\top + \lambda I]^{-1} XY^\top$
- Easy to use in Matlab:

```
> w = [XX' + eye(N)*lambda]^(-1) XY'
```

Outline

Classification vs. Regression

Regression

- prediction, $\hat{y}$, is a real valued number
- continuum of error – measured by how far apart, $\hat{y}$ and y, are

Classification

- $\hat{y}$ is either[1] T or F
- discrete error – its either right or its not!

[1] only binary considered here

Problem Transformation

Turn binary classification problem into regression by:

1. treating y as real valued by setting $T = 1$, $F = -1$
2. training a predictor
3. converting real valued predictions back to classes by thresholding at 0, i.e. $\hat{y} > 0 = T$, $\hat{y} < 0 = F$

- Note: only really works for binary problems, for more it imposes some idea of similarity of classes
- relaxes the discrete error into a real-valued one
- importance of different errors is dependent on the loss-function used

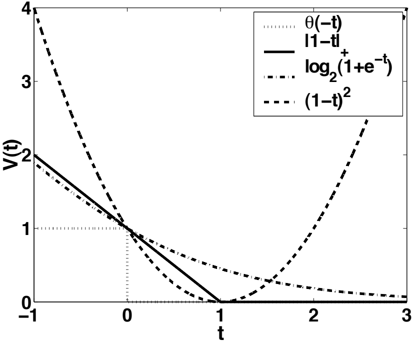
Linear Regularized Least Squares Classification

- Just rLSR applied to the transformed classification problem
- uses the LS loss – this is probably not what you want! (+100 vs. -1)?

Its quick and simple

Classification Loss functions

Better to use a loss function designed for classification problems



- (0-1) loss – $\mathcal{L}(y,\hat{y}) = |y - \text{sgn}(\hat{y})|$
- Hinge loss – $\mathcal{L}(y,\hat{y}) = \max\{1-\hat{y}y, 0\}$ (used in Support vector machines)
- Logistic Loss – $\mathcal{L}(y,\hat{y}) \propto \ln(1+e^{-y\hat{y}})$ (used in Logistic Regression)

Linear Regularised Logistic Regression Classification (rLRC)

- Linear Model: $f(x; w, b) = w^\top x + b$
- Quadratic Regularize: $\mathcal{R}(w) = w^\top w$
- Logistic Loss: $\mathcal{L}(y, \hat{y}) = \ln(1 + e^{-y\hat{y}})$

Gives:

$$J_{\mathrm{rLR}}(w) = \lambda w^\top w + \sum_i \ln(1 + e^{-y_i(w^\top x_i + b)}) \tag{8}$$

Has the nice property that:

- transformed decision function values are valid probabilities, i.e.

$$\mathrm{Pr}(y = T|x) = \left[1 + e^{-f(x)}\right]^{-1} \tag{9}$$

What

I use as my first choice BCI classifier

Outline

<rant on>

Data Hygiene

Don't leak test-set information to your classifier.

Golden Rule of Classification

Test-set(s) are only used for final performance assessment.

i.e. No parameter selection can be based on the test-set. This includes:

- Pre-processing parameters, e.g. mean signal
- Feature extraction parameters, e.g. spectral bands
- Classifier parameters, e.g. regularization parameters, model-type
- This includes post-hoc reporting of only the algorithm which performed best

<rant off>

Tips for using classifiers in BCI

1. Only give it the right features....
2. ... but, don't be too conservative. Its smarter than you think.
3. use the simplest model that works – generally for BCI linear is best.
4. use a regularized classifier – generally quadratic regularization is good enough
5. find the reg-parameter by cross-validation
6. visualize the classifiers parameters as a sanity check

Outline

Advanced Methods

- Non-linear classifiers – just use linear in a transformed feature space
- Multi-class – two (easy) ways to do it:
  1. 1 vs. 1 – train all possible binary sub-problems and vote
  2. 1 vs. Rest – pick the classifier which is “most sure”

Outline

Summary

- Classification is necessary to decode the neural signals into user intentions.
- Classifiers (and regressors) try to fit a model to the data
- One must pick the model-type to match the problem
- The model parameters are found by minimizing an objective function consisting of....
- a loss-function which penalizes mis-fitting of the data, and
- a regularization-function which penalizes over-complex models or over-sensitivity to the data
- Moving from regression (ordinal targets) to classification (categorical targets) requires new loss functions
- Performance assessment should be performed on a separate test set
- cross validation should be used to get robust performance estimates

Background Reading: Journal Article (s)

B Blankertz, G Dornhege, S Lemm, M. Krauledat, G. Curio, K-R Müller, The Berlin Brain-Computer Interface:Machine Learning Based Detection of User Specific Brain States Journal of Universal Computer Science, vol. 12, no. 6 (2006), 581-607