ENTOOL - A Matlab Toolbox for Regression, Classification and Active Learning

C. Merkwirth[1]

J.D. Wichard[2]

M. Ogorzałek[1]

ChristianMerkwirth@web.de JoergWichard@web.de

[1]Jagiellonian University, Kraków, Poland

[2]FMP, Berlin, Germany

Learning a Dependency from Data

Given:	A sample of input-output-pairs $(\vec{x}^{\mu}, y^{\mu})$ with $\mu = 1, \ldots, N$
	A functional dependence $y(\vec{x})$ (maybe corrupted by noisy)
Aim:	Choosing a model (function) $\hat{f}$ out of hypothesis space $\mathcal{H}$
	close to true dependency f as possible
Classification	$f : \mathbf{R}^D \mapsto \{0, 1, 2, ...\}$ discrete classes
Regression	$f : \mathbf{R}^D \mapsto \mathbf{R}$ continuous output

Implementation usually via solution of an appropriate optimization problem:

- Matrix inversion in case of linear regression
- Minimization of a loss function on the training data
- Quadratic programming problem for SVMs

Model Types used in Statistical Learning

- Global Models
  - Linear Models
  - Polynomial Models
  - Neural Networks (MLP)
  - Support Vector Machines
- Semi–global Models
  - Radial Basis Functions
  - Multivariate Adaptive Regression Splines (MARS)
  - Decision Trees (C4.5, CART)
- Local Models
  - k–Nearest–Neighbors
- Hybrid Models
  - Projection Based Radial Basis Functions Network (PRBFN)

Validation and Model Selection

- Generalization error: How does the model perform on unseen data (samples) ?
- Exact generalization error is not accessible since we have only limited number of observations !
- Training on small data set tends to overfit, causing generalization error to be significantly higher than training error
- Consequence of mismatch between the capacity of the hypothesis space $\mathcal{H}$ (VC-Dimension) and the number of training observations
- Validation: Estimating the generalization error using just the given data set
  - Needed for choosing optimal model structure or learning parameters (step sizes etc.)
- Model Selection: Selecting the model with lowest (estimated) generalization error
- But estimation of generalization error is very unreliable on small data sets
- One often ends up with a bad performing model

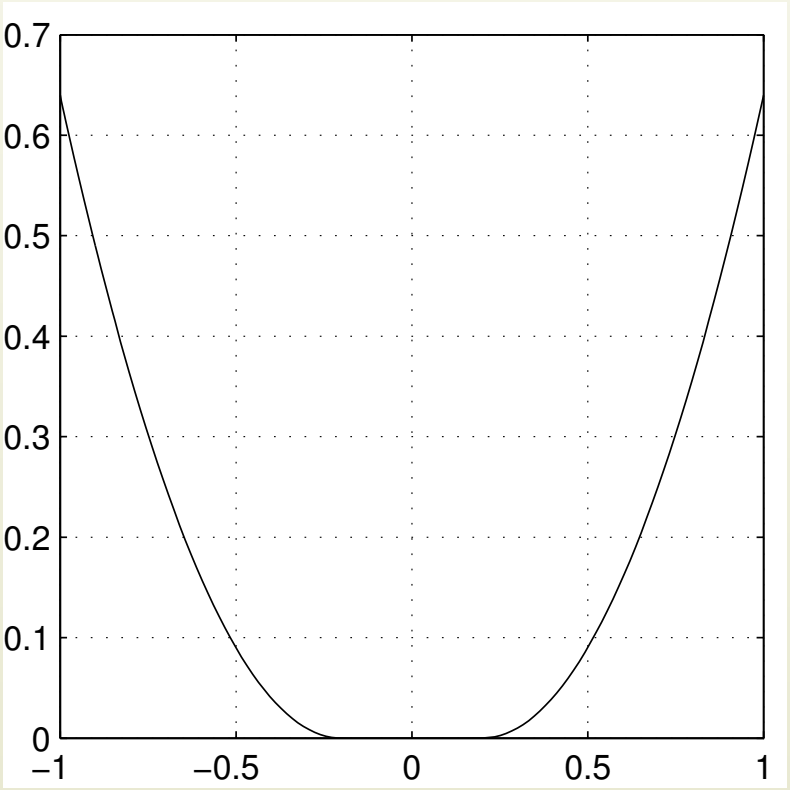
Improving Generalization for Single Models

- Remedies:
  - Manipulating training algorithm (e.g. early stopping)
  - Regularization by adding a penalty to the loss function
  - Using algorithms with built-in capacity control (e.g. SVM)
  - Rely an criteria like BIC, AIC, GCV or Cross Validation to select optimal model complextiy
  - Reformulate the loss function : ϵ-insensitive loss

ϵ-insensitive loss

- ϵ-insensitive quadratic loss is combination of quadratic loss (L_2 norm) with ϵ-insensitivity
- Mitigates the function estimation problem
- Resembles the ϵ-margin of SVM regression
- Samples are approximated correctly as soon as they fall into the ϵ-margin
- Mitigates overfitting
- Makes optimization problem non-unique

- Slightly improves tolerance against outliers in training set



ϵ-insensitive quadratic loss for regression with $\epsilon = 0.2$

Question

- Are there any other methods to improve generalization error ?
- Yes, by combining several individual models!

Ensemble Methods

Error decomposition:

$$
\begin{aligned}
e(\vec{x}) &= (y(\vec{x}) - \bar{f}(\vec{x}))^2 \\
\bar{\epsilon}(\vec{x}) &= \frac{1}{K}\sum_{k=1}^{K}(y(\vec{x}) - f_k(\vec{x}))^2 \\
\bar{a}(\vec{x}) &= \frac{1}{K}\sum_{k=1}^{K}(f_k(\vec{x}) - \bar{f}(\vec{x}))^2 \\
e(\vec{x}) &= \bar{\epsilon}(\vec{x}) - \bar{a}(\vec{x})
\end{aligned}
$$

Integrating over input space:

$$\mathbf{E} = \bar{\mathbf{E}} - \bar{\mathbf{A}}$$

Decorrelating Models

$$\mathbf{E} = \bar{\mathbf{E}} - \bar{\mathbf{A}}$$

How can we obtain models that have low generalization error (small $\bar{E}$), but are mutually uncorrelated (large $\bar{A}$)?

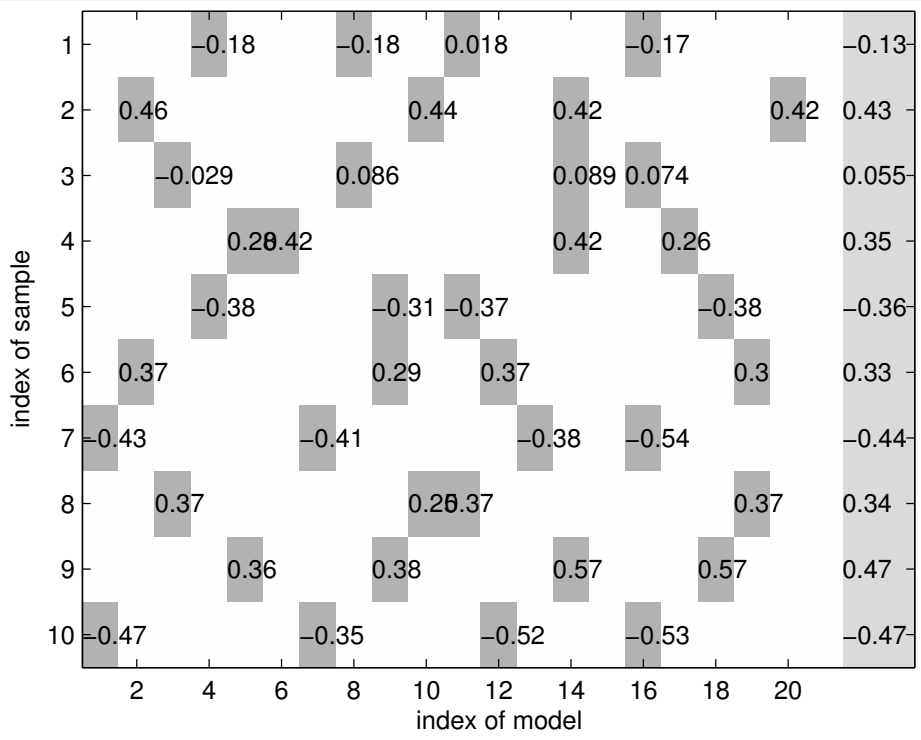
- Varying model structure (e.g. topology)
- Exploiting the disadvantage of getting stuck in local minima:
  - Varying initial conditions
  - Varying parameters of the training procedure
  - Using ϵ-insensitive loss function
- Choosing out of a much bigger population of trained models
- Applying resampling or sequencing techniques:

Crosstraining – Constructing Ensembles

- Finesse: Efficiently reuse samples by combining training, validation and selection of models
- Additional benefit of reduced correlation between models
- Repeatedly partition data set randomly into two sample classes
  - Training set, used for training and stopping criteria
  - Test set, used only for accessing generalization error after model has been trained
- Train population of (heterogenous) models, select best ones according to error on test set
- Repartition data set, taking care that test sets are mutually disjunct
- Combine best models of all partitionings to ensemble
- Optionally weight models according to the estimated generalization error on the total data set

Out-of-Train – CV for Ensembles

- Out-of-Train Calculation is combination of Cross Validation and Ensemble Averaging
- Inspired by Breiman's Out-Of-Bag technique
- Construct ensemble of models on numerous train/test partitionings
- Don't use test samples of each partition for model selection etc.
- OOT output for one sample of data set is average of all models where this sample was not in training set (out-of-train)



- Similar traditional CV, OOT tends to overestimate generalization error!
- Accounts for ensemble gain

Sensitivity Analysis

- Motivation: Determine variable importance with respect to prediction accuracy
- Might help uncovering causal relationships of underlying process
- Problem: Ensemble of heterogenous (nonlinear) models is even more difficult to analyze than single models
- Idea: Combine surrogate data method with OOT calculation
- No retraining of the ensemble necessary
- Uncovers linear and nonlinear relationships

- To determine importance of n-th variable:
  - Create surrogate/replicate of the original input data set where values of n-th variable are permuted randomly to destroy information content
  - Calculate OOT output for surrogate data set
  - Compare errors of OOT output of surrogate and original data set
  - If OOT error increases significantly, the n-th variable is important!
  - Average importance over several surrogate data sets for same variable to smooth out noise

Pros and Cons

Ensemble Methods

- Advantages
  - Straightforward extension of existing modeling algorithms
  - Almost fool-proof minimization of generalization error
  - Makes no assumptions on the structure of the underlying models
  - Alleviates the problem of model selection
- Disadvantages
  - Increased computational effort
  - Interpretation of ensemble is even harder than drawing conclusions from a single model

Combining Heterogenous Models

- Advantages
  - Often one model type performs superior on the given data set
  - Probability of using an unsuited model type decreases
  - Inherent decorrelation even without manipulating data set or training parameters
- Disadvantages
  - Ranking the generalization performance of heterogenous models is even more difficult than for models of same type

Why do Ensembling?

- You have Stat. Learning technique and know perfectly how to create optimal model m_{opt} with lowest expected generalization error $E(G)$
- By chance, you will do for some realization better, for some worse than $E(G)$, but in average you will get $E(G)$
- If your technique to produce m_{opt} gives you always a unique solution, you are fine.
- If not, the expected ambiguity of m_{opt} will be positive $E(A) > 0$
- If you ensemble several of the non-unique solutions, your expected generalization error will never get worse : $E(\bar{G}) = E(G) - E(A)$
- If you are not sure which model type will give optimal generalization error, you can ensemble over different model types (e.g. protein folding meta server, superpositions of FlexX solutions).
- This can be seen as a uniform prior probability over the model types.
- You can only outperform this when you have reliable estimates of the expected generalization errors of the model types (i.e. you can reliably rank model types/methods).

The ENTOOL Toolbox for Statistical Learning

- The ENTOOL toolbox for statistical learning is designed to make state-of-the-art machine learning algorithms available under a common interface
- Allows construction of single models or ensembles of (heterogenous) models
- Supports decorrelation of models by offering resampling techniques
- Though primarily designed for regression, it is possible to construct ensembles of classifiers with ENTOOL

- Requirements:
  - Matlab (TM)
- Operating systems:
  - Windows
  - Linux
  - Solaris (limited)

ENTOOL Software Architecture

- Each model type is implemented as separate class
- All model classes share common interface
- Exchange model types by exchanging constructor call
- Automatic generation of ensembles of models
- Models are divided into two brands:
  1. Primary models like linear models, neural networks, SVMs etc.
  2. Secondary models that rely on primary models to calculate output. All ensemble models are secondary models.

- Lifecycle of a model can be divided into three phases:
  1. During construction, topology of the model is specified. The model can't be used yet.
  2. Model has now to be trained on some training data set $(\vec{x}_i, y_i)$
  3. After training, the model can be evaluated on new/unseen inputs $(\vec{x}_n)$
- Constructors should assign random default topologies in order to create uncorrelated models
- It is possible to construct ensembles of ensembles

Syntax

- Constructor syntax:
  `model = perceptron;` will create a MLP model with default topology
  `model = perceptron(12);` MLP model with 12 hidden layer neurons
  `model = linear;` will create a linear model
- Training syntax:
  `model = train(model, x, y, [], [], 0.05);`
  trains model with ϵ-insensitive loss of 0.05 on data set $(\vec{x}_i, y_i)$
- Evaluation syntax:
  `y_new = calc(model, x_new)` evaluates the model on new inputs
- How to build an ensemble of models:
  `ens = crosstrainensemble;` will create an empty ensemble object
  `ens = train(ens, x, y, [], [], 0.05);` calls training routines for several primary models and joins them into ensemble object
- Ensemble evaluation:
  `y_new = calc(ens, x_new)` evaluates the ensemble on new inputs

Adjusting class specific training parameters

- 5th argument when calling `train` specifies training parameters
- Except topology, often training parameters have to be specified:

```
tp = get(perceptron, 'trainparams')
error_loss_margin:  0.0100
decay:  0.0010
rounds:  500
mrate_init:  0.0100
max_weight:  10
mrate_grow:  1.2000
mrate_shrink:  0.5000
```

- Assign new value: `tp.decay = 0.05`
- And give training parameters while training:

```
model = train(perceptron, x, y, [], tp, 0.05);
```

Specifying which Model Types to Ensemble

- Ensemble constructor will train several models on dataset:

```
tp = get(crosstrainensemble, 'trainparams')
nr_cv_partitions:  8
frac_test:  0.2000
minimum_testsamples:  5
remove_worst:  0.3300
use_models:  0.8000
weight_models:  0
modelclasses:  6x3 cell
scaledata:  1
```

- Assign new value:

```
tp.modelclasses = {'perceptron', [], {}; ...
{'lssvm', [], {'function', 'RBF_kernel', 100, 2}}
```

- And give training parameters while training:

```
ens = train(crosstrainensemble, x, y, [], tp, 0.05);
```

Primary Models Types

ares Adaption of Friedman's MARS algorithm

linear Linear model with optional ridge regression

perceptron Multilayer perceptron with iRPROP+ training

perceptron2 Magnus Nørgaard's single layer perceptron, trained with Levenberg-Marquart

prbfn Shimon Cohen's projection based radial basis function network

rbf Mark Orr's radial basis function code

vicinal k-nearest-neighbor regression with adaptive metric

mpmr Thomas Strohmann's Mimimax Probability Machine Regression

lssvm Johan Suykens' least-square SVM toolbox

tree Adaption of Matlab's build-in regression/classification trees

osusvm SVM code based on Chih-Jen Lin's libSVM

vicinalclass k-nearest-neighbor classification

Ensemble Classes

ensemble Virtual parent class for all ensemble classes

crosstrainensemble Ensemble class that trains models according to crosstraining scheme. Creates ensembles of decorrelated models.

cvensemble Ensemble class that trains models according to crossvalidation/out-of-training scheme. Can be used to access OOT error.

extendingsetensemble Boosting variant for regression.

subspaceensemble Creates an ensemble of models where each single model is trained on a random subspace of the input data set.

optimalsvm Wrapper that trains RBF osusvm/lssvm with optimal parameter settings (C and γ)

featureselector Does feature selection and trains model on selected subset

Summary

- Classification vs. Regression
- Types of models
  - Neural Networks
  - Support Vector Machines
  - Nearest–Neighbor Algorithms
- Methods for improving generalization
  - for single models
  - for ensembles of models

Outlook

- Write ensembles classes specialized on classification
- Make loss function exchangeable
- Add more model types
- Active Learning and DOE can be used in combination with any kind of predictive model (e.g. protein fold prediction, structure activity relationships etc.)

Literature

- Krogh, Vedelsby Neural Network Ensembles, Cross Validation and Active Learning Advances in Neural Information Processing Systems 7, MIT Press 1995
- Peronne, Cooper When networks disagree: Ensemble methods for neural networks Neural Networks for Speech and Image Processing, Chapman Hall 1993
- Hastie, Tibshirani, Friedman The Elements of Statistical Learning Springer 2001
- Vapnik The Nature of Statistical Learning Theory Springer 1999
- Chih-Chung Chang, Chih-Jen Lin LIBSVM : a library for support vector machines Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001
- Freund Short introduction to boosting 1999
- Sacks, Welch, Mitchell, Wynn Design and analysis of computer experiments. Statistical Science, 4(4):409-435, 1989
- Domingos A unified bias-variance decomposition for zero-one and squared loss 2000

Literature cont.

- McNames Innovations in Local Modeling for Time Series Prediction Ph.D. Thesis, Stanford University 1999
- Nørgaard Neural Network Based System Identification Toolbox, Tech. Report. 00-E-891, Department of Automation, Technical University of Denmark 2000
- Orr Matlab Functions for Radial Basis Function Networks 1999
- Cohen, Intrator A Hybrid Projection Based and Radial Basis Function Architecture, Lecture Notes in Computer Science, 2000
- Poland Model-Based and Evolutionary Optimization Algorithms for Motor Development.), PhD thesis 2002

Appendix : Backpropagation

- Method for calculating the gradient of the i-th observation loss $L(y_i, z_i)$ for of a layered function $z = f(\vec{x}_i, \vec{p})$ versus the parameters $\vec{p}$:

$$\vec{g}_i(\vec{p}) = \frac{\partial L(y_i, f(\vec{x}_i, \vec{p}))}{\partial \vec{p}}$$

- Consists of:
  - A forward pass in which the output z of the network is computed and intermediate results are stored
  - A backward pass in which the gradient versus the parameters $\vec{p}$ is computed
- Gradient $\vec{g}$ can be used for minimization of the total loss $E = \sum_i L(y_i, z_i)$

Neural Networks

- Consist of one or more layers of neurons where each neuron is connected to the output of all the neurons in the preceding layer
- Neuron computes weighted sum of its inputs and applies sigmoidal activation function $\Rightarrow$ activity
- Model is characterized by topology and weights
- Training consists of adapting weights until training error is minimized
- Many different training algorithms: gradient based (Backprop,Rprop), second order (Conjugate Gradients, Levenberg Marquardt), Stochastic Annealing etc.
- Large computational variance due to local minima
- Difficult to determine optimal network topology

k–Nearest–Neighbor Models

- Wanted: Value $f(\vec{x})$ at point $\vec{x}$ (query point)
- Determine k points from the training data set that are closest to the query point $\vec{x} \Rightarrow$ nearest–neighbors
- Distances are computed according to some metric (e.g. Manhattan, Euclidian etc.)
- Average over the ys belonging to the k Nearest–Neighbors, weighted by distance $\Rightarrow f(\vec{x})$
- Very low computational variance
- Good for interpolation, bias problems at boundaries of data set
- Brute force computation of neighbors is expensive