Towards Autonomous Machine Learning

Christian Igel
igel@diku.dk

Department of Computer Science
University of Copenhagen

March 2, 2011
(handout version)
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Machine Learning

Learning is goal-directed changing of behaviour based on experience.

Supervised machine learning automates the process of *inductive inference*:

1. Observe a phenomenon
2. Construct a model of that phenomenon
3. Make predictions using this model
Binary classification

two classes: \{apples, pears\}, \{-1, 1\}
Supervised learning

The goal of supervised learning is to uncover an unknown relation between

- input space $\mathcal{X}$, e.g., real-valued vector describing colour and shape of fruit, and
- output space $\mathcal{Y}$, e.g., $\{\text{apples, pears}\}$, based on sample input-output data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$. 
The goal of learning

Assumption: Each pattern is drawn independently from a (unknown) distribution $p$ over $(\mathcal{X}, \mathcal{Y})$.

Given sample data, we want to find a hypothesis predicting output from input showing good average performance on patterns $(x_i, y_i) \sim p$. 
Hypotheses and learning machines

- A hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ maps inputs to outputs; a hypothesis class $\mathcal{H}$ is a set of those functions.
- Applying a learning algorithm means coming up with a hypothesis given sample data.
Decision functions

For binary classification, we usually consider hypothesis

\[ h(x) = \begin{cases} 
  1 & \text{if } d(x) > 0 \\
  -1 & \text{otherwise}
\end{cases} \]

based on decision functions

\[ d : \mathcal{X} \rightarrow \mathbb{R} \]

with decision boundary \( \{ x | x \in \mathcal{X} \wedge d(x) = 0 \} \).
Affine decision functions

Affine linear decision functions

\[ d(x) = \langle w, x \rangle + b \]

with \( \mathcal{X} = \mathbb{R}^n, w, x \in \mathbb{R}^n, b \in \mathbb{R} \) lead to hypothesis with (affine) linear decision boundaries:

\[ x \mapsto \begin{cases} 
1 & \text{if } \langle w, x \rangle + b > 0 \\
-1 & \text{otherwise}
\end{cases} \]
Goal of learning

Prediction quality is quantified by a task-dependent loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$ with $\forall y \in \mathcal{Y} : L(y, y) = 0$.

The typical loss for classification is the 0-1 loss:

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

The goal of supervised learning is to find $h$ minimising the risk

$$\mathcal{R}_p(h) = \int L(y, h(x)) dp(x, y) .$$
Risk and empirical risk

- The risk

\[ R_p(h) = \int L(y, h(x))d\mathcal{P}(x, y) \]

is approximated by the empirical risk of \( h \) on sample data \( S \) (\( |S| = \ell \)):

\[ R_S(h) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, h(x_i)) \]

- Under 0-1 loss, the risk of a hypothesis \( h \) is the probability of error and the empirical risk the fraction of errors on the training data.
Learning by heart is not enough

- Minimising empirical risk does not imply good *generalisation* (i.e., good performance on patterns not seen during training).

- **Overfitting**: The hypothesis faithfully reflects idiosyncrasies of the training data rather than the underlying distribution.
No free lunch...

If there is no a priori restriction on the possible phenomena that are expected, it is impossible to generalise.

Generalisation = Data + Knowledge

Olivier Bousquet

Bousquet, Boucheron, Lugosi. Introduction to Statistical Learning Theory. LNAI 3176, 2004
Prior knowledge

But in real-world applications we can make assumptions about the expected phenomena!

Prior knowledge can be incorporated by

- a proper restriction of the hypothesis space,
- a proper definition of neighbourhood/similarity (e.g., by defining a proper scalar product inducing a metric), and/or
- penalising the “complexity” of classifiers (i.e., preferring “simple” solutions).
Regularised risk minimisation

- The challenge is to find a trade-off between empirical risk and plausibility of the classifier, e.g., measured by its complexity.

- Regularised risk minimisation restricted to hypothesis class $\mathcal{H}$:

  $$h_S = \arg\min_{h \in \mathcal{H}} [\mathcal{R}_S(h) + \text{penalty} (\text{complexity}(h))]$$

  penalty($\cdot$) : $\mathbb{R} \to \mathbb{R}$, non-decreasing function
  complexity($h$) : $\mathcal{H} \to \mathbb{R}$, complexity measure

- Reasonable complexity measures should allow to derive bounds on the expected risk: “The lower the complexity, the higher the probability that the empirical risk is close to the risk.”
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Shortcomings of ML

- Machine learning (ML) is an integral part of today’s computing systems.

- Highly specialised solutions for restricted task reached superhuman performance, usually for problems requiring extensive computation.

- Still, in general ML does not come close to human learning abilities.

- Most ML algorithms
  - lack *autonomy* and
  - suffer from *scalability* problems.
Autonomy and scalability

Autonomous learning
Technical systems should learn autonomously, e.g., not requiring an expert to select
- learning algorithm and hyperparameters,
- appropriate data representation, etc.

Scalability in adaptive systems
We need learning algorithms that are able to
- handle large amounts of data as well as to
- generalise from few training examples.
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Large-margin classifiers and kernel trick

\[ k(x, z) = \langle \Phi(x), \Phi(z) \rangle \]
Binary SVMs

prior knowledge:
- kernel function
- regularisation parameter

My SVM research

- How to choose the right kernel function (i.e., feature space)?
- How to compute the large margin separation efficiently?
- How to realise online and active learning?
- How to keep the final solution fast & small?
- How to realise fast multi-class learning?
- How to build deep kernel machines?
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Bayesian model selection

Given sample data $S$ let’s consider different models $\mathcal{M}$ (e.g., SVM kernel and regularisation parameters) and maximise

$$p(\mathcal{M} | S) = \frac{p(S | \mathcal{M}) p(\mathcal{M})}{p(S)}$$

a posteriori $\propto$ likelihood $\times$ prior

- $p(\mathcal{M} | S)$ probability of the model (parameters) given the data
- $p(\mathcal{M})$ prior of the model (parameters)
- $p(S | \mathcal{M})$ likelihood of the model (parameters)

*How likely are some parameters “in general”?
*How likely is the data given the model?
*Let’s maximise it!*
Maximum likelihood SVM model selection

- SVMs learn a decision function $d : \mathcal{X} \rightarrow \mathbb{R}$ and classify $x \in \mathcal{X}$ according to:

$$x \text{ belongs to class } \begin{cases} 1 & \text{if } d(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- After training, we use a heuristic and fit sigmoid functions $\sigma$ to model class probabilities:

$$p(y_i = 1) = \sigma(d(x_i))$$

Platt, *Advances in Large Margin Classifiers*, pp. 61–74, 1999

- The likelihood is differentiable w.r.t. kernel parameters and regularisation parameter – and we do efficient gradient ascent.
Results

Experiments:

- Multi-variate Gaussian kernels (i.e., the most important kernel family)
- 28 small and moderate size datasets (i.e., when model selection is of utmost importance)
- Comparison with smoothed cross-validation, span-bound, leave-one-out error, and cross-validation

Maximum likelihood model selection for SVMs efficiently and robustly finds models having better generalisation performance than those obtained by alternative methods.

Glasmachers, Igel: Maximum Likelihood Model Selection for 1-Norm Soft Margin SVMs with Multiple Parameters, *IEEE Transactions on Pattern Recognition and Machine Intelligence* 32, 2010
Example: Hydroacoustic signal classification

- Verification of the comprehensive nuclear-test-ban treaty (CTBT)
- Data from hydroacoustic network

- SVMs detects explosive events (4.3 % error)

Tuma, Igel, Prior: Hydroacoustic Signal Classification Using Kernel Functions for Variable Feature Sets. *ICPR 2010*
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Covariance matrix adaptation evolution strategy

- Randomised, derivative free, iterative optimisation algorithm
- Search distribution is updated to make recent beneficial steps more likely
- Invariant under rotation, translation, flipping of search space, and order preserving transformations of objective function
- Both fast and robust, allows for multi-objective optimisation


Example: Line Motion Illusion
Spatio-temporal modeling: Cortical processes

\[ \tau_u \frac{\partial u(x, t)}{\partial t} = -u(x, t) + h_u + g_{uu} \int w_{uu}(\|x - x'\|)f[u(x', t)]dx' - g_{vu}f[v(x, t)] + S(x, t) \]

\[ \tau_v \frac{\partial v(x, t)}{\partial t} = -v(x, t) + h_v + g_{uv} \int w_{vu}(\|x - x'\|)f[u(x', t)]dx' \]

- \( u(x, t), v(x, t) \):
  - membrane potentials
- \( h_u, h_v \):
  - resting potentials
- \( S(x, t) = \int S'(x', t)w_{\text{LGN}}(\|x - x'\|)dx' \):
  - afferent input
- \( w_{uu}(\cdot), w_{vu}(\cdot) \):
  - synaptic weight functions (Gaussian)
- \( f(u) = (1 + e^{-\beta(u-u_0)})^{-1} \):
  - mean firing rate

Results of evolutionary model identification

- **A**: Moving square
- **B**: Flashed square
- **C**: Flashed bar
- **D**: LM cortex (mm)

The image shows the results of a model identification process for different stimuli, comparing simulated and actual data. The x-axis represents time (ms), and the y-axis represents ΔF/F × 10^-4.
Outline

1. Supervised Machine Learning
2. Autonomy and Scalability
3. Support Vector Machines
4. Model Selection for Support Vector Machines
5. Robust Direct Optimisation
6. When Theory and Practice Meet
Shark

http://shark-project.sourceforge.net


Summary

SHARK is a modular C++ library for the design and optimization of adaptive systems. It provides methods for linear and nonlinear optimization, in particular evolutionary and gradient-based algorithms, kernel-based learning algorithms and neural networks, and various other machine learning techniques. SHARK serves as a toolbox to support real world applications as well as research in different domains of computational intelligence and machine learning. The sources are compatible with the following platforms: Windows, Solaris, MacOS X, and Linux.
When theory and practice meet . . .

- Roth, Igel, Handmann: *IJCIA* 4, 2004

- Winter et al.: *IEEE TEC* 12, 2008
  Winter et al.: *UMB* 35, 2009


- Pellecchia et al.: *IEEE Intelligent Sys* 20, 2005

- Sutterp, Igel: *Multi-objective Machine Learning* Ch. 9, Springer, 2006

- Mayr et al.: *Analytical Chemistry* 75, 2003


- Mersch et al.: *IJNS* 17, 2007
When theory and practice meet . . .

Making machine learning systems applicable by increasing their autonomy and scalability using concepts from computational and optimisation theory and from biological information processing.

http://image.diku.dk/igel

Pellecchia et al.: *IEEE Intelligent Sys* 20, 2005
Suttrop, Igel: *Multi-objective Machine Learning* Ch. 9, Springer, 2006

Igel et al.: *IEEE/ACM TCBB* 4, 2007
Mersch et al.: *IJNS* 17, 2007