Evolutionary Computation for Supervised Learning

Christian Gagné

Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique
Université Laval, Québec (Québec), Canada

christian.gagne@gel.ulaval.ca
http://vision.gel.ulaval.ca/~cgagne

GECCO'13 Companion, July 6-10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Evolutionary computation for supervised learning

- Supervised learning
  - Inferring a model from observational data
  - Main objective: to produce models that generalize
  - Two types: classification and regression
  - Wide range of applications
    - Pattern recognition, medical diagnosis, irregularity detection, forecasting (e.g. finance, weather), high-level control, etc.
- Evolutionary computation
  - Bio-inspired meta-heuristics
  - Black-box optimization
    - Derivative-free
    - Non-convex objectives
    - Non-conventional representations
- Supervised learning presents many challenges that can be solved through optimization
  - How can evolutionary computation be useful to improve supervised learning?

Aim and scope

- Questions tackled in this tutorial
  - What is supervised learning and what are its main issues?
  - Where is EC successful for doing supervised learning?
- This tutorial is:
  - A short presentation of relevant notions related to supervised learning
  - A selection of various approaches for evolutionary supervised learning
  - A proposal on how EC can successfully achieve or support supervised learning
- This tutorial is **not**:
  - An exhaustive survey on the application of EC to supervised learning
  - On how to improve EC with machine learning techniques (e.g. surrogate models)

Outline

- Overview of supervised learning
  - Presentation of supervised learning
  - Classification and regression
  - Model selection and generalization
- Applying EC to supervised learning
  - Feature selection and construction
  - Model optimization
  - Ensemble methods
  - Learning methodologies
- Perspectives and concluding remarks

Part I

Supervised Learning Overview

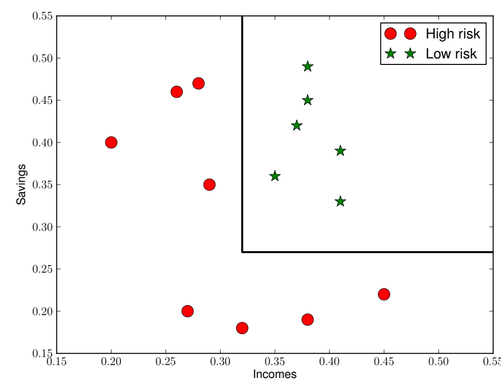
Why machine learning?

- Machine learning consists in using computers for **optimizing** an **information processing model** according to some **performance criteria** based on **observations**, be it data examples or past experiences
- When we know the good processing model to use, there is no need to do learning!
- Machine learning can be useful when:
  - We do not have expertise on the problem (e.g. rover on Mars)
  - We have an expertise, but cannot explain it (e.g. face recognition)
  - Solutions to the problem are changing over time (e.g. packet routing)
  - Solutions must be personalized (e.g. biometric identification)

Example

- A credit company wants to estimate automatically the risk level of its clients
- Available measures : client incomes (x_1) and client savings (x_2)
- Database of clients tagged as high risk (red) or low risk (green)

Example



If $x_1 > 0.32$ **and** $x_2 > 0.27$ **then** *low risk* **else** *high risk*

Model and observations

- Goal: to infer a **general processing model** from **specific observations**
  - The model must be a correct and useful approximation of the observations
- Observations are cheap and often available in high volume; knowledge is rare and expensive
- Example in data mining: link customers transactions to their buying behaviours
  - Suggestion of similar items on Amazon (books, musics), Netflix (movies), etc.

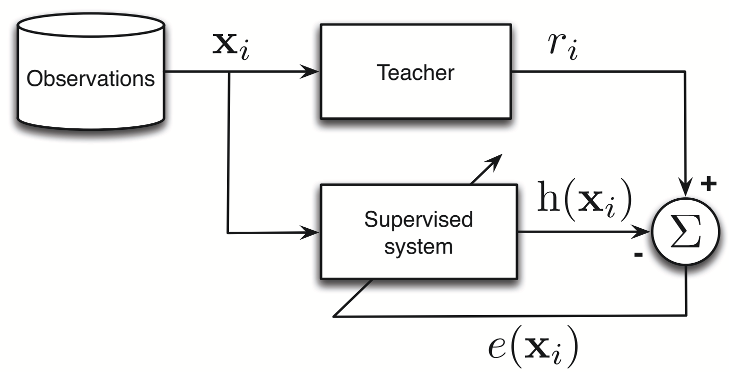
Views of machine learning

- To optimize a model from observations according to a performance criterion
- **Statistical view**: to infer from samples
- **Computing view**: to build algorithms and representations efficient at generating and evaluating the models
- **Engineering view**: to solve problems without having to specify or customize manually the processing models

Supervised learning

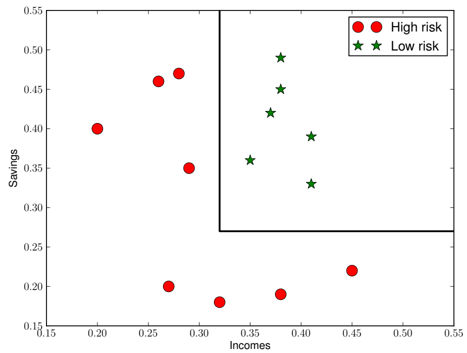
- Supervised learning
  - Goal: to learn a projection between observations X as input and associated values Y as output
- Mathematical model
  - $y = \mathrm{h}(\mathbf{x}|\theta)$
  - $\mathrm{h}(\cdot)$: general model function
  - θ: model parameters

Supervised learning diagram



Classification

- Y is discrete and corresponds to class labels
- $\mathrm{h}(\cdot)$ is a discrimination function



Applications of classification

- Pattern recognition
  - Face recognition: to recognize peoples notwithstanding the variations (pose, lighting, glasses, make-up, hairs)
  - Handwritten character recognition: to recognize characters notwithstanding the different writing styles
  - Speech recognition: temporal dependencies, use dictionaries of valid words/structures
- Decision support in health: to propose diagnosis from the symptoms
- Knowledge extraction and compression: to explain large databases with simple rules
- Irregularity detection: to identify frauds, intrusions, etc.

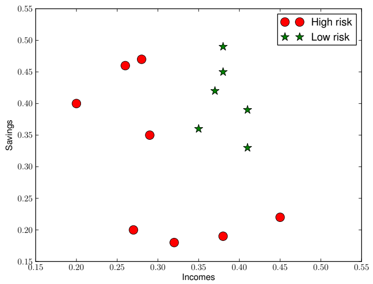
Face recognition

ORL database from AT&T Laboratories Cambridge:
`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

Handwritten character recognition

MNIST database of handwritten characters from Y. LeCun and C. Cortes: `http://yann.lecun.com/exdb/mnist/`

Learning from examples



- Observations:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

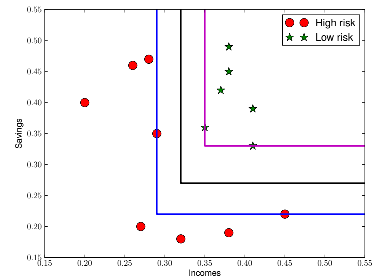
- Class labels:

$$r = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is high risk} \\ 0 & \text{if } \mathbf{x} \text{ is low risk} \end{cases}$$

- Set of N observations:

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

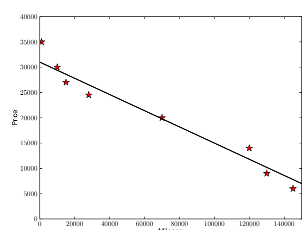
Classification hypotheses



- $\mathrm{h}(\mathbf{x}|\theta)$: parametric classification function
- θ: specific parametrization to the function
- $\theta = \theta_s$: most specific hypothesis (blue)
- $\theta = \theta_g$: most general hypothesis (magenta)

Regression

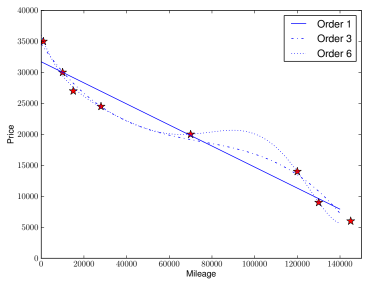
- Y is a real value
- $\mathrm{h}(\cdot)$ is the regression function
- Example: to forecast sale price of used car according to its mileage
  - Observations: mileage (x)
  - Forecast: sale price (y)
- Applications to forecasting
  - Finance
  - Weather
- Applications to high-level control
  - Steering wheel of an autonomous car (CMU NavLab)
  - Joints of a robotic arm



Model complexity and noise

- Noise in the data
  - Lack of precision
  - Labelling errors
  - Latent measures
- At equal performances, prefer the simplest model
  - Easier to use and to train (time and space complexity)
  - Easier to explain (intelligibility)
  - Generalize better (Occam's razor)

Polynomial regression



- First order with one variable:

$$\mathrm{h}(x|w_1,w_0) = w_1 x + w_0$$

- Solution with partial derivatives on empirical error
- Solutions with 1st, 3rd, and 6th order polynomial
  - 6th order is almost "perfect", but generalize badly
  - 3rd order capture better the data than 1st order

Models selection

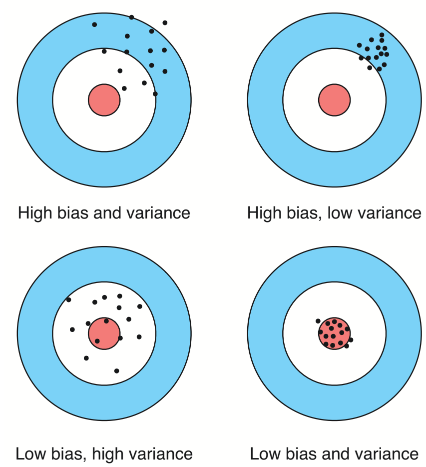
- Supervised learning is an *ill-posed* problem
  - The observations are not sufficient to provide an unique solution
- We thus need an *inductive bias*, by making assumptions on the space of hypothesis (function $\mathrm{h}(x|\theta)$ to use)
- Main objective: **generalization**
  - We need a model that perform well on new data
  - Overfitting: hypotheses $\mathrm{h}(\mathbf{x}|\theta)$ are too complex given the data
  - Underfitting: hypotheses $\mathrm{h}(\mathbf{x}|\theta)$ are too simple given the data
- Regularization: include a model complexity penalty in the optimization objective

Supervised learning trade-offs

- A trade-off must be made between three elements:
  - Hypotheses complexity, C
  - Training dataset size, N
  - Generalization error (on new observations), E
- When N increases, then E decreases
- When C increases, then E decreases for a while, and then increases
- Bias-variance trade-off
  - High bias: model often off target (too simple)
  - High variance: unstable model, does not capture the underneath phenomenon (too complex)
  - Reducing bias usually increases variance, and vice-versa
  - Mean square error is a composition of bias and variance

$$\mathbb{E}\left[(r-\mathrm{h})^2\right] = \underbrace{(r-\mathbb{E}[\mathrm{h}])^2}_{\text{bias}^2} + \mathrm{Var}(\mathrm{h})$$

Bias-variance trade-off



Empirical validation

- To estimate generalization error, we need data unused during training
- Classical approach, partition the dataset
  - Training set (50%)
  - Validation set (25%)
  - Test set (25%)
- Usual procedure
  1. Generate hypotheses $\mathrm{h}(\mathbf{x}|\theta)$ from the training set
  2. Evaluate generalization error of these hypotheses on the validation set and return the one that minimizes it
  3. Report as final performance the results on the test set
- With small datasets, there are other approaches
  - Partition dataset in K folds
  - Use $K-1$ folds for training and the remaining fold for validation
  - Repeat K times with all possible combinations and report the average validation error
  - Extreme case: K is equal to the dataset size (one training per data)

Three dimensions of supervised learning

- Representations
  - Parametrized hypotheses: $\mathrm{h}(\mathbf{x}|\theta)$
  - Instances, hyperplanes, decision trees, rules sets, neural networks, graphical models, etc.
- Evaluation
  - Empirical error: $E(\theta|\mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\ell(r^t,\mathrm{h}(\mathbf{x}^t|\theta))$
  - Recognition rate, precision, recall, square error, likelihood, posterior probability, information gain, margin, cost, etc.
- Optimization
  - Procedure : $\theta^* = \mathrm{argmin}_{\forall\theta}\, E(\theta|\mathcal{X})$
  - Combinatorial optimization, gradient descent, linear/quadratic programming, etc.

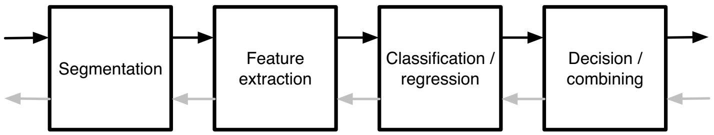
Part II

Evolutionary Computation for Supervised Learning

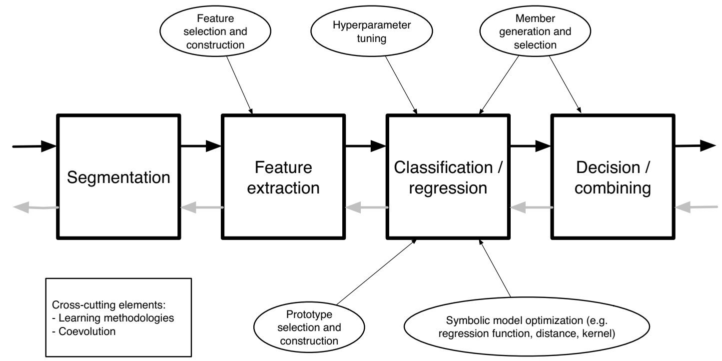
Using EC for supervised learning

- Combinatorial optimization (bit strings and permutations)
  - Data selection (e.g. prototypes)
  - Feature selection
  - Members selection in ensembles
- Real-valued optimization
  - Hyperparameter tuning
  - Unconventional performance measure
  - Prototype construction
- Genetic programming
  - Symbolic regression
  - Feature and classifier model
  - Distance measure and kernel function
- General approaches
  - Member production for ensemble
  - Dynamic evaluation data selection (e.g. competitive coevolution)
  - Learning methodologies and data handling

Pattern recognition pipeline



Where EC can intervene



Feature selection

- Curse of dimensionality
  - Adding one dimension increases exponentially the input space
  - 100 equidistant data in 1D $\Rightarrow 10^{20}$ data in 10D for the same sampling density
  - High dimensionality: increased time and space complexity
- Feature selection (Guyon and Elisseeff, 2003)
  - Objective: to find a subset of K input variables among the D original variables (features) while limiting the impact on performance
  - Number of possible subsets: $\binom{D}{K}$

$$\binom{10}{5} = 252, \ \binom{50}{10} \approx 10^{10}, \ \binom{100}{20} \approx 10^{20}$$

  - Combinatorial optimization problem

Filter vs wrapper

- Filter approach for feature selection
  - Use a statistical measure to evaluate the link between the features and the labels (e.g. mutual information)
  - Usually very fast as the statistical measure is cheap to compute
  - The statistical measure may have little to do with the learning method used
- Wrapper approach for feature selection
  - Train a model for every feature subset candidates
  - Expensive, as a complete training is done for each fitness evaluation
  - Will capture all complex interactions between the features and the method used

Feature selection with EC

- Feature selection has been tackled with EC since a long time (Siedlecki and Sklansky, 1989)
- Multiobjective bit string GA is obvious for that (Emmanouilidis, Hunter, and MacIntyre, 2000; Oliveira et al., 2003)
  - Each bit represents whether a feature is selected
  - Evaluation often done following a wrapper approach
  - Optimizing the performance (e.g. minimizing error rate) while minimizing the number of features selected
- Many have used EC-based feature selection for producing classifiers
  - Acting on the features is algorithm-independent and may influence the classifiers generated
  - Particularly useful for generating a diverse pool of classifiers (see later)

Instance-based classification

- k-Nearest Neighbour (k-NN) classification
  - Assign class label according to the majority label of the k nearest instances
  - Classical approach: select nearest instances in the training set
  - No training required, testing complexity of $N \times M$ (N: train set size, M: test set size)
- Reducing the instance pool size by prototype selection
  - Removing redundant and noisy instances
  - Reduce testing time and space complexity
  - A variety of heuristics has been proposed (Garcia et al., 2012; Wilson and Martinez, 2000)
- Another combinatorial optimization problem!

Prototype selection

- As with feature selection, bit string GA is good for prototype selection (Derrac, García, and Herrera, 2010)
  - Each bit identify whether an instance is used as prototype
  - Kuncheva and Bezdek (1998) used a single objective with a weighted sum of performance and number of prototypes
  - Require however to select from a relatively small pool of instances (when representing a selection as a bit string)
- Simultaneous prototype and feature selection (Kuncheva and Jain, 1999)

Prototype construction

- Prototype selection: select instances from a pool
  - Why not creating new prototypes from scratch!
  - Prototype construction might produce smaller but more representative set of prototypes
- Common approaches for prototype construction
  - Clustering the data set (e.g. K-means)
  - Learning vector quantization (a kind of supervised K-means)
- Evolutionary prototype construction (Derrac, García, and Herrera, 2010; Kuncheva and Bezdek, 1998)
  - Used real-valued algorithm to evolve $\mathbf{x}$ values of a given number of prototypes
  - Another approach: sequential optimization, where each run evolves a bunch of prototypes with Particle Swarm Optimization (PSO) (Nanni and Lumini, 2009)
  - Michigan-style PSO for prototype construction (Cervantes, Galván, and Isasi, 2009)

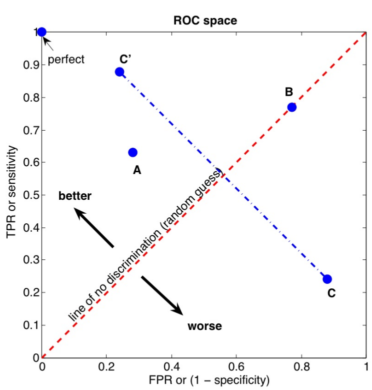
Real-valued EC for supervised learning?

- Should we optimize the real-valued parameters with EC?
  - Optimization in learning often solved through convex optimization procedure
    - SVM: quadratic programming
    - Neural networks: gradient descent (backpropagation)
    - Variants of Boosting (e.g. LPBoost)
  - When convex optimization works well, do not try to beat it with EC
    - Convex optimization techniques are well-known, converge usually faster and/or to better solutions (with guarantees)
- However, real-valued EC has its niches
  - Prototype construction
  - Hyperparameter tuning
  - Unconventional optimization objectives (e.g. non-convex, non-differentiable)
  - Multiobjective optimization

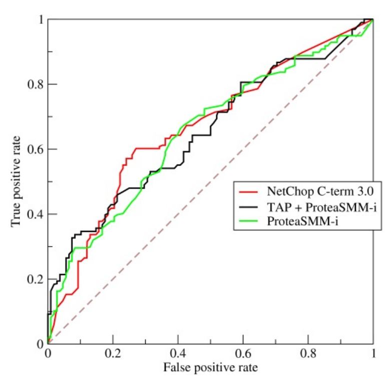
AUC-ROC

- ROC curves (Fawcett, 2006)
  - x-axis: false positive rate
  - y-axis: true positive rate
  - Given a real-valued output, position on the curve correspond to a threshold
  - Allow evaluating performance for different types of errors or varying class balance
- Area under the ROC curve (AUC-ROC)
  - Evaluate the capacity to discriminate two classes for all threshold values
  - Independent of the class balance
  - Strong links with the Wilcoxon–Mann–Whitney statistical test and Gini coefficient
  - Hard to handle by convex optimization methods
- Evolving classifiers using the AUC-ROC as fitness measure (Sebag, Azé, and Lucas, 2004)

ROC Curves



http://commons.wikimedia.org/wiki/File:ROC_space.png



http://en.wikipedia.org/wiki/File:Roccurves.png

Hyperparameter tuning

- Hyperparameters: parameters of the learning algorithm
  - Learning rate and regularization coefficient
  - Number of hidden layers and neurons
  - Number of neighbours
  - Parametrization of kernel functions
- Sensitivity to these values varies
  - Sometime, ballpark figures are good enough
  - In other cases, fine tuning of hyperparameters is required
  - For some algorithms, there are complex interactions between hyperparameters
- Grid search
  - Testing all combinations of hyperparameter values
  - Efficient for 1 to 3 parameters, using relatively coarse set of values
- Evolutionary algorithms for hyperparameters
  - Tuning regularization coefficient (C) and Gaussian kernel covariance matrix of SVMs with CMA-ES (Friedrichs and Igel, 2005)
  - Tuning SVMs with multiobjective GA (TP, FP, and #SV) (Suttorp and Igel, 2006)

Neuroevolution

- Artificial neural networks often used for classification and regression
  - Classical network: Multilayer Perceptron (MLP)
  - New trend: deep networks
- Optimizing neural network topologies
  - Hyperparameter tuning: optimizing the number of layers and neurons of MLPs
- Neuroevolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002)
  - Evolve both the weights and topology of the network
  - Try to find a balance between fitness and speciation
  - Start with simple topologies and develop them incrementally
- In general, neuroevolution has not appeared particularly fit for supervised learning
  - Much better at control/reinforcement learning tasks

Genetic programming

- Genetic Programming (GP) is a natural approach for supervised learning
  - Classification/regression model can be seen as a computer program
  - Specifying the GP configuration for evolving the model is straightforward in many cases
- Evolve variable-length model
  - Allow to produce models of varying complexity
  - Bloat problem can be fought through regularization, much like what is done in supervised learning (Amil et al., 2009)
  - Models produced are symbolic and intelligible
- Applications of GP to classification (Espejo, Ventura, and Herrera, 2010)
  - Feature construction
  - Decision trees
  - Rule-based systems
  - Discriminant functions

Symbolic regression

- Introductory example for GP (Koza, 1992)
  - Infer an equation in its analytical form from a set of test cases
  - Arithmetic operators as branches (e.g. $+, -, \times, \div \sin, \cos, \exp, \log$)
  - Variables of the problem (i.e. $x_1, \ldots, x_D$) and constants (e.g. $0, 1, \pi, ERC$) as terminals
- Still relatively efficient for doing regression
  - Particularly interesting when symbolic equations are requested
  - Does an implicit feature selection
- See the GECCO workshop on symbolic regression and modelling

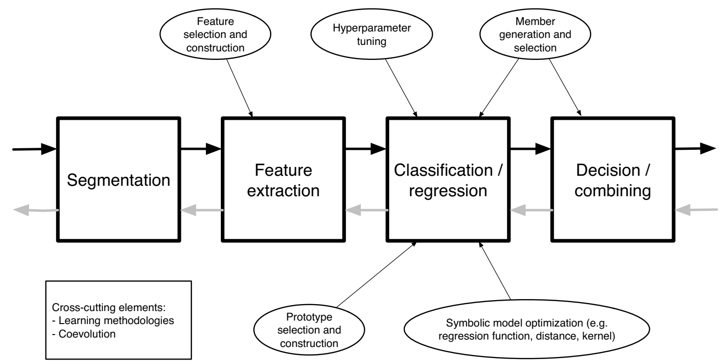
Feature construction

- Feature construction
  - Creating new features from the existing ones
  - Usually allow to reduce the input size of the model
  - Particularly interesting when done through some non-linear mapping
  - Wrapper and filter methods can be used
- Domain knowledge is usually difficult to obtain
  - Building automatically features should help to extract useful information and use the good representation
- Feature construction with GP
  - Make use of symbolic regression to construct features
  - Evolve all features at the time (Sherrah, Bogner, and Bouzerdoum, 1997) or one feature constructed at the time (Bot, 2001)
  - Multiobjective feature construction with GP (Zhang and Rockett, 2009)

Evolving distance measure or kernel function

- Distance measure: evaluate how dissimilar are two values
  - Central component of instance-based classifiers (e.g. k-NN)
  - Most common is Euclidean distance, but others are possible
  - Using GP to evolve the distance measure of classifiers (Gagné and Parizeau, 2007)
    - Evolve a $\mathrm{d}(\mathbf{x},\mathbf{y})$ with vector instructions (i.e. similar to Matlab)
- Kernel function: measure similarity of two data
  - Central in SVM and other kernel methods
  - Allow mapping the input space in an higher dimension one, without working explicitly in it (kernel trick)
  - Kernels can be a composition of other kernels
  - Evolving kernels with GP (Gagné et al., 2006; Sullivan and Luke, 2007)
    - Branches and terminals allows to define basic kernels that are combined through the evolution
    - Allow customization of the kernel function to the problem domain

Where EC can intervene (bis)



Ensemble methods

- Condorcet's jury theorem (1785)
  - Assuming a jury of independent voters who have a probability of $p > 1/2$ of making the correct decision
  - Jury reaches correct decision asymptotically (with probability of 1), as jury size increases
  - Votes assumed to be independent and identically distributed (i.i.d.)
  - Theoretical justification of democracy
- Making ensembles of classifiers/regression functions
  - Ensembles are usually more reliable than single classifiers
  - Eliminate noise of individual decisions
  - Require members to be diversified
- Weak members are sufficient to make ensembles
  - No need to obtain ultra high performances, better than 50% (better than random) is good enough
  - Often easier to generate diversity with weak algorithms

Bias-variance trade-off with ensembles

- Bias and variance with ensembles
  - h_j are i.i.d., with expectation $\mathbb{E}[\mathrm{h}_j]$ and variance $\mathrm{Var}(\mathrm{h}_j)$

$$\begin{aligned}
\mathbb{E}[\bar{\mathrm{h}}] &= \mathbb{E}\left[\sum_{j=1}^{L}\frac{1}{L}\mathrm{h}_j\right] = \frac{1}{L}L\,\mathbb{E}[\mathrm{h}_j] = \mathbb{E}[\mathrm{h}_j] \\
\mathrm{Var}(\bar{\mathrm{h}}) &= \mathrm{Var}\left(\sum_{j=1}^{L}\frac{1}{L}\mathrm{h}_j\right) = \frac{1}{L^2}L\,\mathrm{Var}(\mathrm{h}_j) = \frac{1}{L}\mathrm{Var}(\mathrm{h}_j)
\end{aligned}$$

- Variance decreases as the number of members (L) increases
  - With ensembles, we can reduce variance without altering bias
  - And so is reduced the mean square error

$$\mathbb{E}\left[(r-\mathrm{h})^2\right] = \underbrace{(r-\mathbb{E}[\mathrm{h}])^2}_{\text{bias}^2} + \mathrm{Var}(\mathrm{h})$$

Diversity and negative correlation

- Ensemble variance, general case

$$\mathrm{Var}(\bar{\mathrm{h}}) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j \mathrm{h}_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}\left(\mathrm{h}_j\right) + 2\sum_j\sum_{i>j}\mathrm{Cov}(\mathrm{h}_j,\mathrm{h}_i)\right]$$

  - Reduce further variance with negatively correlated members
  - Square error can be reduced, as far as negative correlation does not alter bias
- Diversity of responses in ensembles
  - Goal when creating ensembles: members are not making mistakes on the same data
  - Extreme case without diversity: L copies of the same member
- Evolutionary ensembles with negative correlation learning (Liu, Yao, and Higuchi, 2000)
  - Make ensemble of neural networks for regression
  - Individual networks trained with backpropagation + negative correlation
  - Using EC to generate the members of the ensemble

Overproduce and select

- Overproduce: generate a varied pool of classifiers
- Select: choose a subset of classifiers from the pool, maximizing a given measure (performance and/or diversity)
  - Feature selection techniques transpose well to member selection
- EC is good for overproduction
  - Diversity in the population is a already a desired property of EC
  - Diversity measures are often hard to use with convex optimization
  - Population of solutions = pool of classifiers
  - Generating a diverse pool through evolutionary feature selection (Oliveira, Morita, and Sabourin, 2006)
- Evolutionary member selection
  - Dynamic selection of members at runtime with NSGA-II, according to the data to classify (Dos Santos, Sabourin, and Maupin, 2008)
  - Overfitting cautious member selection methodology relying on multiobjective GA (Dos Santos, Sabourin, and Maupin, 2009)

Ensembles for free

- Evolving a population of classifiers
  - Why not making a ensemble of classifiers, using the population as a pool?
  - Diversity of the population = diversity of the pool?
- Ensemble learning for free with EC (Gagné et al., 2007)
  - Using EC to produce a population of classifiers
    - Fitness function enforcing diversity by assigning a fixed credit for each test case
  - The ensemble is build by selecting members from the population
    - *Off*-EEL: select the members from the final generation
    - *On*-EEL: build the ensemble during the evolution, incrementally
  - Somehow related to Michigan-style algorithms

Bagging and Boosting

- Bagging: generate passively varied classifiers through random resampling of training set
- Boosting: produce varied classifiers by modifying sampling weights of data according to their difficulty
- BagGP and BoostGP (Iba, 1999)
  - Split the population into subpopulations
  - Resample training set for each subpopulation, using Bagging or Boosting
  - Make ensemble with the best individual of each subpopulation
- GPboost: modify weighting of test cases of several sequential GP runs (Paris, Robilliard, and Fonlupt, 2002)

Dynamic subset selection

- Dataset size for evolutionary learning is a concern
  - Many individuals evaluated with a large datasets ⇒ expensive computation
  - Not all instances need to be used for evaluating all individuals at each generation
- Dynamic Subset Selection (DSS) (Gathercole and Ross, 1994)
  - Evaluate fitness with a training subset of "difficult" instances
  - Compute a weight for each training instance according to its age and difficulty
  - Assign a selection probability according to the normalized instance weight and target training subset size
  - Renew subset at each generation
- A variant of DSS has been successfully applied to train GP classifiers with a dataset of 500 000 instances (Song, Heywood, and Zincir-Heywood, 2005)

Competitive coevolution

- Competitive coevolution (Hillis, 1990)
  - Evolving species with antagonistic goals (i.e. parasite-host model)
  - Can reduce significantly the number of test cases for each individual
- Coevolutionary symbolic regression (methods for evolving robust programs) (Panait and Luke, 2003)
  - Host species: symbolic regression with GP
  - Parasite species: test cases evolved with real-valued GA
  - Good at improving generalization, by renewing test cases at each generation
- Coevolving nearest neighbour classifiers (Gagné and Parizeau, 2007)
  - Species 1: distance measure with GP
  - Species 2: prototype selection with multiobjective GA (cooperative)
  - Species 3: selection of evaluation data with GA (competitive)
  - Competitive coevolution limits greatly overfitting, with reduced distance measure and prototypes set size

Oversearching

- Discriminate charlatans from competent financial counsellors (Jensen and Cohen, 2000)
  - Ask counsellors to predict whether stock markets will go up or down on a day
  - Request to make prediction for 14 days, a candidate is deemed competent if he predicts correctly 11 days or more
    - A charlatan makes random guesses (50%/50%), so have 2.87% chances of passing the test
- Does not work for selecting a counsellor among n
  - Probability that a charlatan passes the test among n: $1-(1-0.0287)^n$
    - For $n=10$, $\approx$ 25% chances one charlatan will pass the test, for $n=30$, $\approx$ 58% chances
  - For high n, almost sure that charlatans will pass the test, even thought they are not doing better than random guesses
- Oversearching: searching for solutions in huge model spaces
  - By testing too many candidate solutions, may select one that fit well the training set, but does not generalize well
  - Common issue when doing supervised learning with EC

Learning methodologies

- Recommendations to avoid overfitting and oversearching (Igel, 2012)
  1. Use as much data as possible, to improve training and fitness evaluation reliability
  2. When relevant, use a distinct dataset from the training set for evaluating the fitness (use an evaluation set)
  3. If possible, renew evaluation dataset at each generation
  4. Generalization performance must be evaluated on data not used for computing the fitness (use a validation set)
  5. Number of evaluations before oversearching should be evaluated, which is dependent of the amount of data available
  6. Final results shall be reported on a distinct dataset (use a test set)
- Up to four datasets may be required in a proper methodology
  - Training set: to train classifiers
  - Evaluation set: to evaluate fitness of individual on new data
  - Validation set (a.k.a. final selection set): to select the individual to retain from an evolution and/or do early stopping
  - Test set: to evaluate generalization performances and compare different algorithms

Part III

Perspectives and Concluding Remarks

Where is EC useful for supervised learning?

- Optimizing classification/regression models with EC
  - Many state-of-the-art models rely on convex optimization methods (e.g. SVM)
    - EC not likely to figure well compared to these approaches
  - But EC can achieve excellent results in specific cases
    - Prototype selection/construction for instance-based learning
    - Hyperparameter tuning, when there is a complex relation among these (e.g. C and σ of Gaussian SVMs)
    - Non-convex, non-differentiable performance measure (e.g. AUC-ROC)
    - Intelligible models (e.g. symbolic regression)

Where is EC useful for supervised learning? (cont.)

- Building representations
  - Feature selection/construction
  - Distance measures and kernel functions
  - Segmentation level of the pattern recognition pipeline
- Building ensembles
  - Generating pool of diverse models
  - Selecting members for making the ensembles
  - Population of models = an ensemble!
- Many optimization challenges in supervised learning
  - EC can be very useful where other "classical" methods fail
  - Combinatorial optimization
  - Multiobjective optimization
  - Variable-length and symbolic representations (i.e. GP)

Methodological guidelines

- Dataset size trade-off of evolutionary learning
  - Avoid using small datasets
    - Learning has moved beyond the few hundreds instances found in most toy datasets
    - With small datasets further partitioning gets difficult
  - Big dataset implies long fitness evaluation
    - EC is expensive in term of number of candidate solutions evaluated
- Proper supervised learning with EC requires up to 4 datasets
  - Training set, evaluation set, validation set, and test set
- Oversearching issue
  - Large datasets are required to avoid good performances by chance
  - Selecting best-of-run with a validation set
  - Validation set good also for early stopping
- Renewing the evaluation set during the evolution
  - Competitive coevolution, dynamic subset selection, etc.

New horizons

- Deep learning (Bengio, 2009)
  - "The biggest data science breakthrough of the decade"
  - Techniques to train neural network with many layers (deep networks)
  - Several EC techniques can be tackled to develop better network (e.g. neuroevolution)
- Large-scale learning (Bottou and Bousquet, 2011)
  - Big data learning: how to apply *efficiently* (performance- and computation-wise) supervised learning to huge databases?
  - Implicit parallelism of EC can allow relatively fast processing on parallel machines, along with some clever data management
- Semi-supervised learning (Zhu, 2007)
  - Big databases, with only a small subset of data labelled
  - Learn structures from unlabelled data, tag then with labelled one

Conclusion

- Many researchers in machine learning have low esteem of EC
  - Just a bunch of *ad hoc* bio-inspired stochatic methods (not so *ad hoc*)
  - There is no theoretical proofs supporting the methods (that's not true!)
  - Very expensive computation required, close to brute force search (sometime true)
- Tackle the good problems, where classical learning fails
  - Some problems are ignored in machine learning, as they do not fit the tools they are used to
- Be audacious but humble
  - Learning community is hyperactive and so moving quickly
  - Before doing anything, understand what the community knows on the problem and the solutions proposed

References I

Amil, N. M., N. Bredeche, C. Gagné, S. Gelly, M. Schoenauer, and O. Teytaud (2009). "A statistical learning perspective of genetic programming". In: *Proc. of EuroGP*. Springer, pp. 327–338. Url: http://dx.doi.org/10.1007/978-3-642-01181-8_28.

Bengio, Y. (2009). "Learning deep architectures for AI". In: *Foundations and Trends in Machine Learning* 2.1, pp. 1–127. Url: http://dx.doi.org/10.1561/2200000006.

Bot, M. C. (2001). "Feature extraction for the k-nearest neighbour classifier with genetic programming". In: *Proc. of EuroGP*. Springer, pp. 256–267. Url: http://dx.doi.org/10.1007/3-540-45355-5_20.

Bottou, L. and O. Bousquet (2011). "The Tradeoffs of Large Scale Learning". In: *Optimization for Machine Learning*. Ed. by S. Sra, S. Nowozin, and S. J. Wright. MIT Press, pp. 351–368. Url: http://leon.bottou.org/papers/bottou-bousquet-2011.

Cervantes, A., I. M. Galván, and P. Isasi (2009). "AMPSO: a new particle swarm method for nearest neighborhood classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39.5, pp. 1082–1091. Url: http://dx.doi.org/10.1109/TSMCB.2008.2011816.

Derrac, J., S. García, and F. Herrera (2010). "A survey on evolutionary instance selection and generation". In: *International Journal of Applied Metaheuristic Computing (IJAMC)* 1.1, pp. 60–92. Url: http://sci2s.ugr.es/pr/pdf/2010-Derrac-IJAMC.pdf.

Dos Santos, E. M., R. Sabourin, and P. Maupin (2008). "A dynamic overproduce-and-choose strategy for the selection of classifier ensembles". In: *Pattern Recognition* 41.10, pp. 2993–3009. Url: http://dx.doi.org/10.1016/j.patcog.2008.03.027.

References II

Dos Santos, E. M., R. Sabourin, and P. Maupin (2009). "Overfitting cautious selection of classifier ensembles with genetic algorithms". In: *Information Fusion* 10.2, pp. 150–162. Url: http://dx.doi.org/10.1016/j.inffus.2008.11.003.

Emmanouilidis, C., A. Hunter, and J. MacIntyre (2000). "A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator". In: *Proc. of IEEE-CEC*, pp. 309–316. Url: http://dx.doi.org/10.1109/CEC.2000.870311.

Espejo, P. G., S. Ventura, and F. Herrera (2010). "A survey on the application of genetic programming to classification". In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 40.2, pp. 121–144. Url: http://dx.doi.org/10.1109/TSMCC.2009.2033566.

Fawcett, T. (2006). "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8, pp. 861–874. Url: http://dx.doi.org/10.1016/j.patrec.2005.10.010.

Friedrichs, F. and C. Igel (2005). "Evolutionary tuning of multiple SVM parameters". In: *Neurocomputing* 64, pp. 107–117. Url: http://dx.doi.org/10.1016/j.neucom.2004.11.022.

Gagné, C. and M. Parizeau (2007). "Coevolution of nearest neighbor classifiers". In: *International Journal of Pattern Recognition and Artificial Intelligence* 21.05, pp. 921–946. Url: http://dx.doi.org/10.1142/S0218001407005752.

Gagné, C., M. Schoenauer, M. Sebag, and M. Tomassini (2006). "Genetic programming for kernel-based learning with co-evolving subsets selection". In: *Proc. of Parallel problem solving from nature*. Springer, pp. 1008–1017. Url: http://dx.doi.org/10.1007/11844297_102.

References III

Gagné, C., M. Sebag, M. Schoenauer, and M. Tomassini (2007). "Ensemble learning for free with evolutionary algorithms?" In: *Proc. of the Genetic and evolutionary computation*. ACM, pp. 1782–1789. Url: http://dx.doi.org/10.1145/1276958.1277317.

Garcia, S., J. Derrac, J. R. Cano, and F. Herrera (2012). "Prototype selection for nearest neighbor classification: Taxonomy and empirical study". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3, pp. 417–435. Url: http://dx.doi.org/10.1109/TPAMI.2011.142.

Gathercole, C. and P. Ross (1994). "Dynamic training subset selection for supervised learning in genetic programming". In: *Proc. of Parallel Problem Solving from Nature*. Springer, pp. 312–321. Url: http://dx.doi.org/10.1007/3-540-58484-6_275.

Guyon, I. and A. Elisseeff (2003). "An introduction to variable and feature selection". In: *Journal of Machine Learning Research* 3, pp. 1157–1182. Url: http://jmlr.csail.mit.edu/papers/v3/guyon03a.html.

Hillis, W. D. (1990). "Co-evolving parasites improve simulated evolution as an optimization procedure". In: *Physica D: Nonlinear Phenomena* 42.1, pp. 228–234. Url: http://dx.doi.org/10.1016/0167-2789(90)90076-2.

Iba, H. (1999). "Bagging, boosting, and bloating in genetic programming". In: *Proc. of the Genetic and evolutionary computation conference*. Vol. 2, pp. 1053–1060. Url: http://www.cs.bham.ac.uk/~wbl/biblio/gecco1999/GP-407.pdf.

Igel, C. (2012). "A Note on Generalization Loss When Evolving Adaptive Pattern Recognition Systems". In: *IEEE Transactions on Evolutionary Computation* PP. Url: http://dx.doi.org/10.1109/TEVC.2012.2197214.

References IV

Jensen, D. D. and P. R. Cohen (2000). "Multiple comparisons in induction algorithms". In: *Machine Learning* 38.3, pp. 309–338. Url: http://dx.doi.org/10.1023/A:1007631014630.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press. Url: http://books.google.com/books/about/Genetics_programming.html?id=Bhtxo60BV0EC.

Kuncheva, L. I. and J. C. Bezdek (1998). "Nearest prototype classification: Clustering, genetic algorithms, or random search?" In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 28.1, pp. 160–164. Url: http://dx.doi.org/10.1109/5326.661099.

Kuncheva, L. I. and L. C. Jain (1999). "Nearest neighbor classifier: simultaneous editing and feature selection". In: *Pattern Recognition Letters* 20.11, pp. 1149–1156. Url: http://dx.doi.org/10.1016/S0167-8655(99)00082-3.

Liu, Y., X. Yao, and T. Higuchi (2000). "Evolutionary ensembles with negative correlation learning". In: *IEEE Transactions on Evolutionary Computation* 4.4, pp. 380–387. Url: http://dx.doi.org/10.1109/4235.887237.

Nanni, L. and A. Lumini (2009). "Particle swarm optimization for prototype reduction". In: *Neurocomputing* 72.4, pp. 1092–1097. Url: http://dx.doi.org/10.1016/j.neucom.2008.03.008.

Oliveira, L. S., M. Morita, and R. Sabourin (2006). "Feature selection for ensembles using the multi-objective optimization approach". In: *Multi-Objective Machine Learning*. Springer, pp. 49–74. Url: http://dx.doi.org/10.1007/3-540-33019-4_3.

References V

Oliveira, L. S., R. Sabourin, F. Bortolozzi, and C. Y. Suen (2003). "A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition". In: *International Journal of Pattern Recognition and Artificial Intelligence* 17.6, pp. 903–929. Url: http://dx.doi.org/10.1142/S021800140300271X.

Panait, L. and S. Luke (2003). "Methods for evolving robust programs". In: *Proc. of the Genetic and evolutionary computation conference*. Springer, pp. 1740–1751. Url: http://dx.doi.org/10.1007/3-540-45110-2_66.

Paris, G., D. Robilliard, and C. Fonlupt (2002). "Applying boosting techniques to genetic programming". In: *Proc. of Artificial evolution*. Springer, pp. 267–278. Url: http://dx.doi.org/10.1007/3-540-46033-0_22.

Sebag, M., J. Azé, and N. Lucas (2004). "ROC-based evolutionary learning: Application to medical data mining". In: *Proc. of Artificial Evolution*. Springer, pp. 384–396. Url: http://dx.doi.org/10.1007/978-3-540-24621-3_31.

Sherrah, J. R., R. E. Bogner, and A. Bouzerdoum (1997). "The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming". In: *Proc. of the Genetic Programming conference*. Citeseer, pp. 304–312.

Siedlecki, W. and J. Sklansky (1989). "A note on genetic algorithms for large-scale feature selection". In: *Pattern Recognition Letters* 10.5, pp. 335–347. Url: http://dx.doi.org/10.1016/0167-8655(89)90037-8.

References VI

Song, D., M. I. Heywood, and A. N. Zincir-Heywood (2005). "Training genetic programming on half a million patterns: an example from anomaly detection". In: *IEEE Transactions on Evolutionary Computation* 9.3, pp. 225–239. Url: http://dx.doi.org/10.1109/TEVC.2004.841683.

Stanley, K. O. and R. Miikkulainen (2002). "Evolving neural networks through augmenting topologies". In: *Evolutionary computation* 10.2, pp. 99–127. Url: http://dx.doi.org/10.1162/106365602320169811.

Sullivan, K. M. and S. Luke (2007). "Evolving kernels for support vector machine classification". In: *Proc. of the Genetic and evolutionary computation conference*. ACM, pp. 1702–1707.

Suttorp, T. and C. Igel (2006). "Multi-objective optimization of support vector machines". In: *Multi-objective machine learning*. Springer, pp. 199–220. Url: http://dx.doi.org/10.1007/3-540-33019-4_9.

Wilson, D. R. and T. R. Martinez (2000). "Reduction techniques for instance-based learning algorithms". In: *Machine learning* 38.3, pp. 257–286. Url: http://dx.doi.org/10.1023/A:1007626913721.

Zhang, Y. and P. I. Rockett (2009). "A generic multi-dimensional feature extraction method using multiobjective genetic programming". In: *Evolutionary Computation* 17.1, pp. 89–115. Url: http://dx.doi.org/10.1162/evco.2009.17.1.89.

Zhu, X. (2007). *Semi-Supervised Learning Literature Survey*. Tech. rep. Computer Sciences TR 1530. University of Wisconsin – Madison. Url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.9681&rep=rep1&type=pdf.