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

C. Gagné (U. Laval)  
EC for Supervised Learning  
GECCO 2013 Tutorial 5 / 68

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).

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 behaviour
    - 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
  - $y = h(x; \theta)$
  - $h(\cdot)$: general model function
  - $\theta$: model parameters

Supervised learning diagram

```
Observations  X_i
              Teacher  T_i
          h(X_i)
Supervised system
      +
     e(X_i)
```

C. Gagné (U. Laval)  EC for Supervised Learning  GECCO 2013 Tutorial 9 / 68

C. Gagné (U. Laval)  EC for Supervised Learning  GECCO 2013 Tutorial 10 / 68

C. Gagné (U. Laval)  EC for Supervised Learning  GECCO 2013 Tutorial 11 / 68

C. Gagné (U. Laval)  EC for Supervised Learning  GECCO 2013 Tutorial 12 / 68
Classification

- $Y$ is discrete and corresponds to class labels
- $h(\cdot)$ 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

Learning from examples

- Observations:
  \[ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]
- Class labels:
  \[ r = \begin{cases} 1 & \text{if } x \text{ is high risk} \\ 0 & \text{if } x \text{ is low risk} \end{cases} \]
- Set of \( N \) observations:
  \[ \mathcal{X} = \{x^t, r^t\}_{t=1}^N \]

Classification hypotheses

- \( h(x; \theta) \): parametric classification function
- \( \theta \): 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
- \( h(\cdot) \) 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:
  \[ h(x|w_1, w_0) = w_1 x + w_0 \]
- 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 \( h(x|\theta) \) to use)
- Main objective: **generalization**
  - We need a model that perform well on new data
  - Overfitting: hypotheses \( h(x|\theta) \) are too complex given the data
  - Underfitting: hypotheses \( h(x|\theta) \) 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

\[
\mathbb{E}[(r - \hat{h})^2] = (r - \mathbb{E}[h])^2 + \text{Var}(h)
\]

\( \text{bias}^2 \)
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
  - Generate hypotheses $h(x|\theta)$ from the training set
  - Evaluate generalization error of these hypotheses on the validation set and return the one that minimizes it
  - 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: $h(x|\theta)$
  - Instances, hyperplanes, decision trees, rules sets, neural networks, graphical models, etc.
- Evaluation
  - Empirical error: $E(\theta|X) = \frac{1}{N} \sum_{t=1}^{N} \ell(r_t, h(x_t|\theta))$
  - Recognition rate, precision, recall, square error, likelihood, posterior probability, information gain, margin, cost, etc.
- Optimization
  - Procedure: $\theta^* = \arg\min_{\theta} E(\theta|X)$
  - Combinatorial optimization, gradient descent, linear/quadratic programming, etc.

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

Segmentation → Feature extraction → Classification / regression → Decision / combining

Where EC can intervene

Segmentation → Feature extraction → Classification / regression → Decision / combining

Feature selection

• Curse of dimensionality
  ▶ Adding one dimension increases exponentially the input space
  ▶ 100 equidistant data in 1D ⇒ $10^{20}$ 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: \[ \binom{D}{K} \]
    \[ \binom{10}{5} = 252, \quad \binom{50}{10} \approx 10^{10}, \quad \binom{100}{20} \approx 10^{20} \]
  ▶ 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 $x$ 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)

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. +, -, *, /, 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 \( d(x, y) \) 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

Bias-variance trade-off with ensembles

- \( h_j \) are i.i.d., with expectation \( E[h_j] \) and variance \( \text{Var}(h_j) \)

\[
E[h] = E\left[ \sum_{j=1}^{L} \frac{1}{L} h_j \right] = \frac{1}{L} E[h_j] = E[h_j]
\]

\[
\text{Var}(\tilde{h}) = \text{Var}\left( \sum_{j=1}^{L} \frac{1}{L} h_j \right) = \frac{1}{L^2} L \text{Var}(h_j) = \frac{1}{L} \text{Var}(h_j)
\]

- 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

\[
E \left[ (r - h)^2 \right] = (r - E[h])^2 + \text{Var}(h)
\]

Where EC can intervene (bis)

- Segmentation
- Feature extraction
- Classification / regression
- Decision / combining

Cross-cutting elements:
- Learning methodologies
- Coevolution

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
Diversity and negative correlation

- Ensemble variance, general case
  \[
  \text{Var} (\bar{h}) = \frac{1}{L^2} \text{Var} \left( \sum_j h_j \right) = \frac{1}{L^2} \left[ \sum_j \text{Var} (h_j) + 2 \sum_{i<j} \text{Cov}(h_i, h_j) \right]
  \]
- 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 an 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
    - \textit{Off-EEL}: select the members from the final generation
    - \textit{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 dataset ⇒ 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
  - 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 predictions 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, ≈ 25% chances one charlatan will pass the test, for n = 30, ≈ 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)
  - Use as much data as possible, to improve training and fitness evaluation reliability
    - When relevant, use a distinct dataset from the training set for evaluating the fitness (use an evaluation set)
    - If possible, renew evaluation dataset at each generation
    - Generalization performance must be evaluated on data not used for computing the fitness (use a validation set)
    - Number of evaluations before oversearching should be evaluated, which is dependent of the amount of data available
    - 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

C. Gagné (U. Laval) EC for Supervised Learning GECCO 2013 Tutorial 57 / 68

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 $\sigma$ of Gaussian SVMs)
    - Non-convex, non-differentiable performance measure (e.g. AUC-ROC)
    - Intelligible models (e.g. symbolic regression)

C. Gagné (U. Laval) EC for Supervised Learning GECCO 2013 Tutorial 58 / 68

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)

C. Gagné (U. Laval) EC for Supervised Learning GECCO 2013 Tutorial 59 / 68

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 terms 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.

C. Gagné (U. Laval) EC for Supervised Learning GECCO 2013 Tutorial 60 / 68
### 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 stochastic 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 (sometimes 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 III


References IV


References V


References VI


