pertinence
simple solutions for complex processes

www.pertinence.com

Challenges and Applications of Machine Learning to Manufacturing Problems

6th Mathias Seminar – Cannes

Oct 18th, 2006

Olivier Bousquet

> Agenda

- **What is Machine Learning?**
- **Current trends and challenges**
- **Applications**

What is Machine Learning

- **Building models from data (as opposed to ‘from first principles’)**
- **Can be used to model physical phenomena**
- **But also many other complex phenomena/systems**
  - Recognition problems (speech, handwritten text, images)
  - Web problems
  - Industrial processes
  - Bioinformatics
- **Three possible goals**
  - Analyze data to extract relevant information
  - Solve a complex problem without an explicit program
  - Perform prediction of complex phenomena

What is specific?

- **A specific way of processing data in order to make an inductive leap**
- **Induction is a gamble**
- **Based on your knowledge and observations you bet on the *future outcomes* or on the *underlying structure* (or regularities)**
- **Example**
  - after observing planets moving, Newton bet that the law would be of a certain form (it is correct as long as no observation contradicts it)
- **Two dual views**
  - Prediction: the extracted model is used to perform prediction of unobserved data
  - Compression of regularities: the extracted model is a summary of the observations

Supervised Learning

- **Most common situation**
  - Observations: $(x_1, y_1), \ldots , (x_n, y_n)$
  - Goal: construct a function **f** which
    1. **maps x's to y's**
    2. **agrees well with the observations**
    3. **makes correct predictions for unobserved x's**
- **Classification**
  - Y is in a finite set of "classes", Y={1,2,…,k}
  - Ex: recognition problems, spam filtering…
- **Regression**
  - Y is a real number
  - Ex: yield of a manufacturing process
- **Structured learning**
  - Y is a "structured object", such as a string, a text, an image, a graph…
  - Ex: protein secondary structure prediction

General Methodology

- **There are 3 main steps**
  1. Choice of a formalism for specifying the model (i.e. a mathematical/algorithmic form)
  2. Introduction of a preference over the elements of the model
  3. Choice of a way to trade-off between the preference and the agreement with the data
- **Examples**
  - Probabilistic modeling (Bayesian)
    1. **Multivariate normal distribution (on the X space), linear model from X to Y**
    2. **Prior distribution on the weight vector**
    3. **Bayes rule, Maximum Likelihood estimation**
  - Linear classification
    1. **Thresholded linear model**
    2. **l1-norm penalty for the weight vector**
    3. **Minimum of mean classification error plus norm (regularization)**

Agenda

- What is Machine Learning?
- Current trends and challenges
- Applications

Challenges

- **Growth of data**
  - Data storage / processing is cheaper and cheaper
  - Sensors are cheaper and cheaper
  - Produced data is more and more structured (no longer simple measurements but images, spectra, videos, structured text…)
- **Leads to very high dimensional problems**

- **Growth of the tool set**
  - Many communities are producing data analysis techniques (Statistics, Data Mining, Machine Learning…)
  - Many specialized techniques developed in separated fields (image processing, speech, text, DNA…)
- **Requires principled approaches**

Historical perspective (very sketchy)

- **Statistics**
  - parametric: finite-dimensional models, may not be consistent (no convergence even with infinite amount of data)
  - non-paramteric: usually relying on models that are not adapted to high dimensional X
- **Machine Learning**
  - Symbolic methods: Decision Trees, Rules, ILP… (lead to NP-hard problems – need for heuristics)
  - Non-symbolic: Perceptron (linear classifier), Neural networks…
- **First step: from combinatorial to smooth optimization**
- **Second step: from smooth to convex**
  - Makes optimization tractable
  - Allows to tackle large scale problems in a principled way

Modern approaches

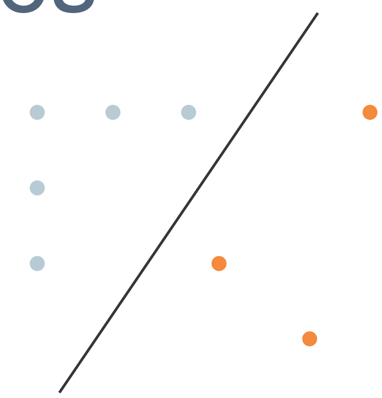
- **Current trends: solving high dimensional structured problems**
- **Two extreme cases to be considered**
  - Small number of observations (when testing is expensive, e.g. clinical studies, batch manufacturing,…)
  - Large number of observations (data streams, internet data…)
- **Trends**
  - Formulate problems as convex optimization ones
  - Use/Develop methods for large scale optimization problems
  - Regularized linear methods often used
  - Develop tools for dealing with structured data

Dealing with high dimensions

- **Two directions**
  - Reducing the dimension
  - Regularization
- **Both turn out to be very similar**

- **Reducing the dimension**
  - Feature selection: select a subset of the variables
  - Feature construction (e.g. boosting): create a small number of combined variables
  - Manifold learning: extract a low-dimensional manifold underlying the data distribution

Margin-based approaches



- **Idea: create linear model**

$$f(x) = sign\left(\sum_{i=1}^{d} w_i x^i\right)$$

- **But in high dimensional space**
- **Need regularization**

•It is easy to fit the training data (when dimension is larger than the sample size)

•This leads to overfitting (no generalization, error is 0 on the training set but large on validation set)

•To overcome this, we need to restrict the weight vector

- **L_2 regularization**

•inner product structure, kernels

$$w = \arg\min \sum_{i=1}^{n} \ell\left(\operatorname{sgn}\langle w | x_i \rangle, y_i\right) + \lambda \|w\|_2^2$$

- **L_1 regularization**

•sparsity

$$w = \arg\min \sum_{i=1}^{n} \ell\left(\operatorname{sgn}\langle w | x_i \rangle, y_i\right) + \lambda \|w\|_1$$

Regularization

- **Two main approaches**

•Tikhonov (e.g. SVM, Kernel Ridge Regression)

$$w = \arg\min \sum_{i=1}^{n} \ell\left(\operatorname{sgn}\langle w | x_i \rangle, y_i\right) + \lambda \|w\|_2^2$$

•Iterations (e.g. AdaBoost, PLS), update the weight vector so as to minimally increase the norm (projection methods)

- **In both cases, need to choose the parameter (lambda or number of iterations) adequately, usually by cross-validation**

L_2: Support Vector Machines

- **Using the Euclidean structure and duality of the optimization problem**
  - Solution of the problem expressed only in terms of inner products
  - Representer theorem $$\langle w | x \rangle = \sum_{i=1}^{n} \alpha_i \langle x_i | x \rangle$$
- **With an appropriate loss function, solution is sparse: only few non-zero coefficients**

Kernels: definition

- **Inner products can be replaced by other "similarity" measures**

  •Requirement for convexity of the optimization problem: behave like an inner product in some abstract space

  •Notion of positive definite kernel

$$\sum_{i=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

- **Corresponds to an implicit mapping to a higher dimensional space**

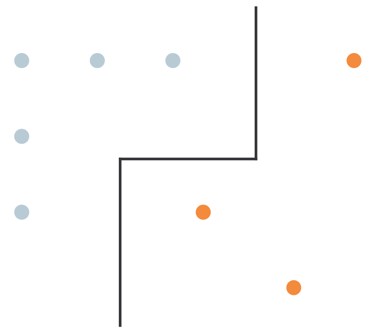
$$k(x, x') = \langle \phi(x) | \phi(x') \rangle$$

- **Known and used for a long time in geostatistics (kridging), but their power really comes in high dimensions**

Kernels: applications

- **Dimensionality of the optimization problem is equal to the number of data points (irrespective of the dimension of the space)**
  - Allows to deal with high-dimensional problems
  - Allows to deal with structured inputs (just need to compute a similarity). Examples: kernels for strings, graph elements, graphs, images, groups of points, distributions…
- **Allows to generate non-linear decision boundaries**
  - Without increasing the complexity of the problem to solve
  - Allows to incorporate implicit knowledge (sometimes similarity is easier to define that explicit features, e.g. for sequences)
  - Many standard techniques can be non-linearized (PCA, PLS, ICA…)

L_1: Boosting Methods



- ## Idea

  •L_1 regularization leads to sparse solutions (only a few non-zero coefficients) in the weight vector

  •So one can deal with large dimensions and produce (relatively) simple solutions

  •One can even create additional dimensions by using simple learning algorithms

- ## Principle

  •Choose a set of possible basis decision functions H

  •Create a regularized linear combination of them (e.g. convex)

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i h_i(x) \right)$$

Boosting: infinite dimensional optimization

- **Usually hypothesis spaces are infinite**
  - No direct way to deal with such problems
  - Solution: using an iterative optimization approach
  - Trick: sparsity ensures that this will converge fast

- **Approach**
  - Maintain a set of weights over the examples
  - At each step, look for hypothesis making the smallest weighted error
  - Add this hypothesis to the combination
  - Update the weights (typically increasing that of misclassified examples)

Computational Aspects

- ## Kernels
  - Effectively work in high (possibly infinite) dimensional spaces
  - No need to explicitly compute the feature representation
  - Computational effort in the kernel computation (e.g. dynamic programming)

- ## Boosting
  - Effectively works in infinite spaces
  - Computational effort in the search for the best hypothesis (e.g. dynamic programming)

Loss functions

- **Another distinctive feature: adapted loss functions for various problems**
- **Principles**
  - No need to estimate parameters $\|\theta - \theta^*\|$
  - No need to find the right model $\|f - f^*\|$
  - Accuracy is required only in regions of high probability $E(f(X) - Y)^2$
  - Consider the end goal directly $E\ell(f(X), Y)$
  - Focus on the right problem: no need to estimate P(X,Y) but only P(X|Y)
- **Examples**
  - Classification: hinge loss $(1-x)_+$ instead of square loss
  - Ranking: number of wrongly ordered pairs
  - Optimization: not position, but value close to maximum
  - Control setting: end quality, not model quality

Agenda

- What is Machine Learning?
- Current trends and challenges
- Applications

Manufacturing Processes Optimization

- **Consider a manufacturing process**
- **For each produced batch or product data are collected**
  - Raw material characteristics
  - Process parameters
  - Inline sensors
  - Final quality information
- **Goal: based on on-going production data, improve the process**
  - Understand sources of quality variability
  - Monitor the process and identify failures early
  - Suggest ways to improve the parameters settings
  - Provide a model for dynamical control of the process

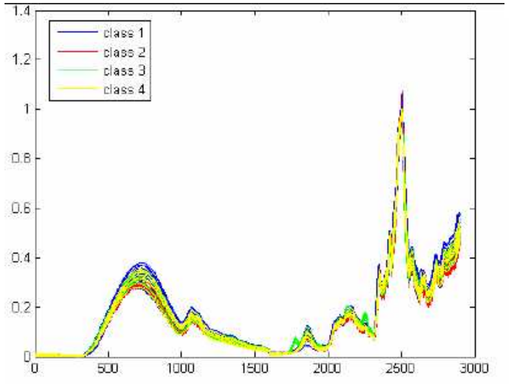
What is needed?

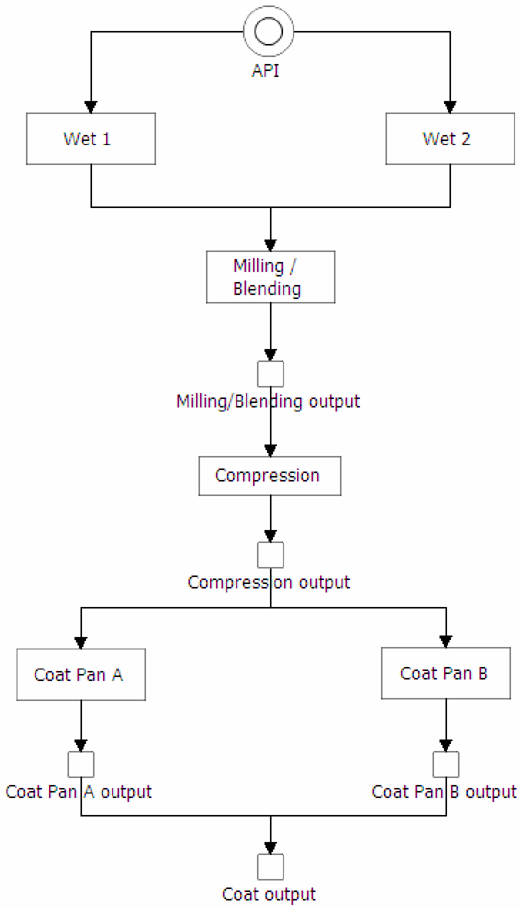
- **Efficient and versatile data analysis**
  - Ability to cope with real-world data (missing values, noise, mixed data types…)
  - Ability to take into account operational constraints
- **Interaction with experts (with varied levels of data analysis expertise)**
- **Guidance towards operational actions**
  - Focus is not on good predictions but on correctly identifying the best parameter zone

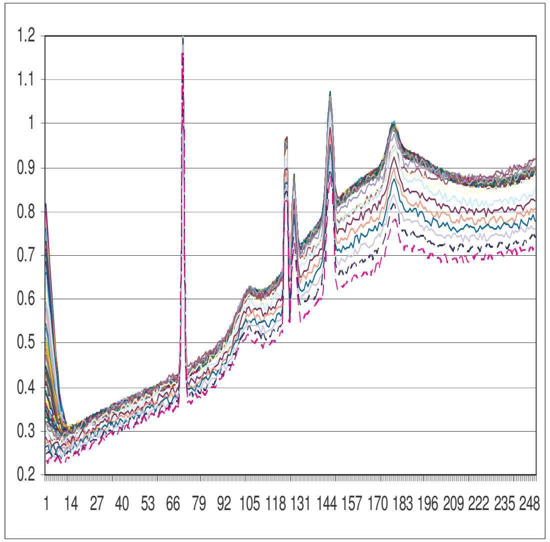
Example

- **Example from Pharmaceutical Manufacturing**
  - Tablet manufacturing
  - Mixing various ingredients with measured physical properties
  - Goal is to obtain good quality tablets (fast dissolution, non-brittle…)
- **Process control in two steps**
  - Estimate properties of raw material (excipient and active ingredients) from NIR spectra
  - Determine appropriate process parameters (that adapt to the raw material quality)
- **Requires to mix two kinds of data**
  - Advanced sensors (NIR spectra)
  - Process parameters

Tablet manufacturing







- Raw data contains 104 static variables (input and process parameters)
- 2 sets of spectra

Data

- **Historical data (44 batches)**
  - Spectrograms
  - API/Excipients characteristics
  - Process parameters

- **To be combined with**
  - Expertise
  - Regulatory / Process constraints

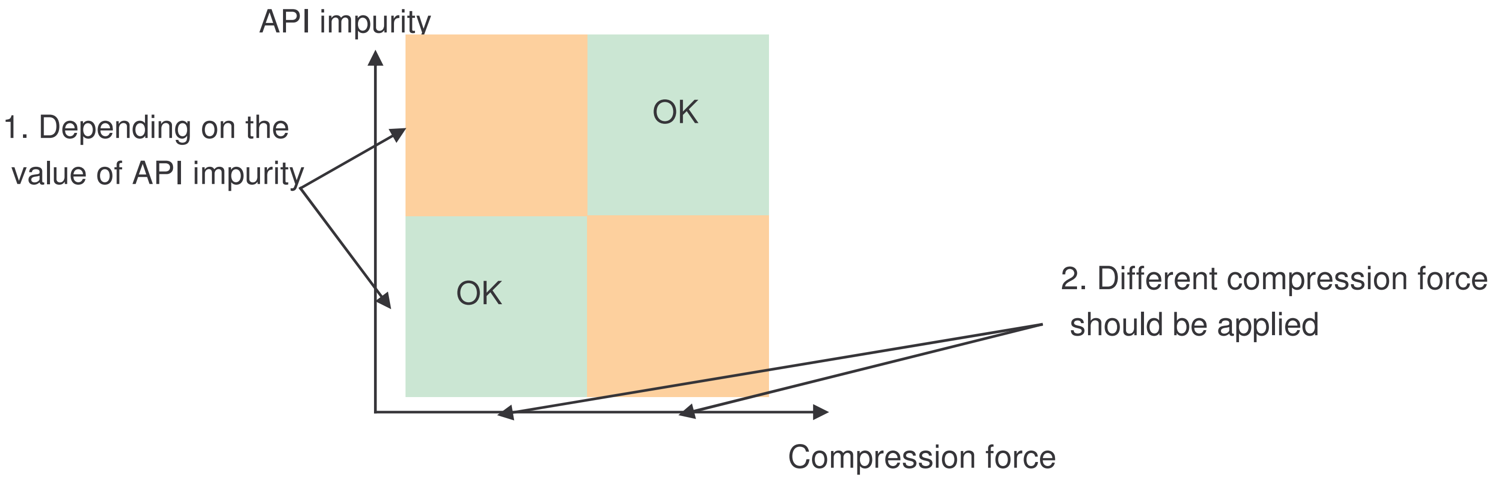
Analysis

- **Spectral calibration**
  - Wavelet transform (extract meaningful variability)
  - Combination of algorithms
    1. **Build a model with each of the algorithms (e.g. SVM, PLS, Boosting)**
    2. **Combine the models to get maximum predictive accuracy**
  - Determine the excipient type (perform prediction from the spectrum)
  - Determine the level of API purity (perform prediction from the spectrum)

- **Generate rule-based model**
  - Extract a set of explanatory rules from batch data

Rule-based model

- **Model is composed of several rules**
  - Rule = hyper-rectangle in parameter space
  - These rules map the parameter space
  - Model is piecewise constant
- **Why do we need more than one rule?**
  - Different situations to be covered
  - Different settings needed as a function of the raw material characteristics



Build an explanatory model

- **Such a model can be**
  - Automatically generated from the data
  - Easily edited by the user
- **Enhancing the rules with human expertise and practical constraints**
  - Reviewing the effect of adding parameters
  - Reviewing the effect of constraining the range of parameters
  - Iterative process guided by rule indicators
- **Goal of this process**
  - Build understanding
    1. **Identify influence of API/excipient characteristics on the end quality**
    2. **Determine conjoint influence of the parameters on the end quality**
  - Based this information, determine the best settings for process parameters

Other applications

- Speed-up complex optimization problems (when evaluating the function is costly), cf David's talk
- Building control models in dynamically changing environments (Reinforcement Learning)
- Estimating parameters of differential equations (work in progress)
- Generally speaking: a more direct approach, no need to model exactly the problems, focus on the end goal

Conclusion

- **The diversity and size of problems to be addressed is growing dramatically**
- **Versatile techniques are being developed**
  - to cope with high-dimensional problems,
  - based on large-scale convex optimization problems
  - with a philosophy of focusing on the final goal rather than the accuracy of the model
- **Choosing the right kernel/basis hypotheses is still an art rather than a science**
- **Applications to industrial problems**
  - raise interesting new questions
  - require combination with more “explicit” methods
  - direction of research: combining empirical with first principles models