HEP data: Finding structure in the noise

Tim Salimans

Algoritmica

November 14, 2014

Typical types of data

Statistics / Econometrics

Example: forecasting GDP

- weak relationships
- simple structure
- high noise level

Machine Learning / AI

Example: image recognition

- strong relationships
- deep/complex structure
- low noise level

Typical types of data

Statistics / Econometrics

Example: forecasting GDP

- weak relationships
- simple structure
- high noise level

Machine Learning / AI

Example: image recognition

- strong relationships
- deep/complex structure
- low noise level

HEP data has both high noise & deep structure

Sources of noise

- imperfect detectors
  - measurement errors
  - undetected particles (e.g. neutrinos)

Sources of noise

- imperfect detectors
  - measurement errors
  - undetected particles (e.g. neutrinos)
- imperfect simulators

Sources of noise

- imperfect detectors
  - measurement errors
  - undetected particles (e.g. neutrinos)
- imperfect simulators
- $H \rightarrow \tau\tau$ signal quite rare
  - high variance in importance weights
  - small effective sample size

Complex structure

Complex mapping from primitives to signal class

- Particle momenta individually have low correlation with signal class
- Relationship between particle momenta is complex
- Derived variables help some
- Variables like MMC are hard to calculate

Complex structure

Complex mapping from primitives to signal class

- Particle momenta individually have low correlation with signal class
- Relationship between particle momenta is complex
- Derived variables help some
- Variables like MMC are hard to calculate

Good model search needed to find correct relationship

- Hard to find correct model by greedy search
- Standard boosted decision trees (e.g. GBM in R) may perform poorly
- XGBoost / **RGF** are better at model search
- Neural nets excellent at discovering *deep* relationships

Statistical efficiency

Directly maximizing AMS not a good idea: much too noisy. Better to estimate the odds ratio of signal s to background b and then apply a cutoff.

Statistical efficiency

Directly maximizing AMS not a good idea: much too noisy. Better to estimate the odds ratio of signal s to background b and then apply a cutoff.

Goal: build model for $R = \frac{\mathbb{E}[w\mathbb{I}(\text{label}=s)|x]}{\mathbb{E}[w\mathbb{I}(\text{label}=b)|x]}$, with w the importance weights, s, b the signal and background identifiers, and x the measured particle momenta. Expectation is taken w.r.t. the *simulator distribution*.

Statistical efficiency

Directly maximizing AMS not a good idea: much too noisy. Better to estimate the odds ratio of signal s to background b and then apply a cutoff.

Goal: build model for $R = \frac{\mathbb{E}[w\mathbb{I}(\text{label}=s)|x]}{\mathbb{E}[w\mathbb{I}(\text{label}=b)|x]}$, with w the importance weights, s, b the signal and background identifiers, and x the measured particle momenta. Expectation is taken w.r.t. the *simulator distribution*.

Problem: The importance weights w are highly variable: small effective sample size.

Statistical efficiency

Decompose the problem to improve efficiency:

$$R = \frac{\mathbb{E}[w\mathbb{I}(\text{label} = s)|x]}{\mathbb{E}[w\mathbb{I}(\text{label} = b)|x]} = R_1 R_2,$$

with

$$\begin{aligned} R_1 &= \frac{\mathrm{P}(\text{label} = s|x)}{\mathrm{P}(\text{label} = b|x)} \\ R_2 &= \frac{\mathbb{E}[w|x, \text{label} = s]}{\mathbb{E}[w|x, \text{label} = b]} \end{aligned}$$

Statistical efficiency

Decompose the problem to improve efficiency:

$$R = \frac{\mathbb{E}[w\mathbb{I}(\text{label} = s)|x]}{\mathbb{E}[w\mathbb{I}(\text{label} = b)|x]} = R_1 R_2,$$

with

$$\begin{aligned} R_1 &= \frac{\mathrm{P}(\text{label} = s|x)}{\mathrm{P}(\text{label} = b|x)} \\ R_2 &= \frac{\mathbb{E}[w|x, \text{label} = s]}{\mathbb{E}[w|x, \text{label} = b]} \end{aligned}$$

Advantage: The subproblems are easier (larger effective sample size)

Statistical efficiency

Decompose the problem to improve efficiency:

$$R = \frac{\mathbb{E}[w\mathbb{I}(\text{label} = s)|x]}{\mathbb{E}[w\mathbb{I}(\text{label} = b)|x]} = R_1 R_2,$$

with

$$\begin{aligned} R_1 &= \frac{\mathrm{P}(\text{label} = s|x)}{\mathrm{P}(\text{label} = b|x)} \\ R_2 &= \frac{\mathbb{E}[w|x, \text{label} = s]}{\mathbb{E}[w|x, \text{label} = b]} \end{aligned}$$

Advantage: The subproblems are easier (larger effective sample size)
Disadvantage: Solving the subproblems might give a biased solution for the original problem

Gradient boosting machine

Friedman, 2001

Algorithm 1: Gradient Boosted Decision Tree (GBDT) [15]

$h_0(\mathbf{x}) \leftarrow \arg\min_\rho \mathcal{L}(\rho, Y)$
for $k = 1$ **to** K **do**
  $\tilde{Y}_k \leftarrow -\partial\mathcal{L}(h, Y)/\partial h|_{h=h_{k-1}(X)}$
  Build a J-leaf decision tree $T_k \leftarrow \mathcal{A}(X, \tilde{Y}_k)$ with leaf-nodes $\{b_{k,j}\}_{j=1}^J$
  for $j = 1$ **to** J **do** $\beta_{k,j} \leftarrow \arg\min_{\beta \in \mathbb{R}} \mathcal{L}(h_{k-1}(X) + \beta \cdot b_{k,j}(X), Y)$
  $h_k(\mathbf{x}) \leftarrow h_{k-1}(\mathbf{x}) + s \sum_{j=1}^J \beta_{k,j} \cdot b_{k,j}(\mathbf{x})$ // s is a shrinkage parameter
end
return $h(\mathbf{x}) = h_K(\mathbf{x})$

- *functional gradient descent*
- very general, no need to normalize covariates
- popular implementation in R works well for many applications
- greedy model search, does not work well for HEP data

Regularized greedy forest

Johnson & Zhang, 2014. Variation on gradient boosting that decouples structure search and optimization.

Algorithm 3: Regularized greedy forest framework

$\mathcal{F} \leftarrow \{\}$.
repeat
  $\mathcal{F} \leftarrow$ the optimum forest that minimizes $\mathcal{Q}(\mathcal{F})$ among all the forests that can be obtained by applying one step of structure-changing operation to the current forest $\mathcal{F}$.
  if *some criterion is met* **then** optimize the leaf weights in $\mathcal{F}$ to minimize loss $\mathcal{Q}(\mathcal{F})$.
until *some exit criterion is met*
Optimize the leaf weights in $\mathcal{F}$ to minimize loss $\mathcal{Q}(\mathcal{F})$.
return $h_{\mathcal{F}}(\mathbf{x})$

- L_2 regularization of leaf coefficients for noise control
- $Q()$ used in structure search can be different from $Q()$ used for optimization of leaf coefficients
- use less regularization in structure search to make the search less *greedy*, key to make this work for HEP

Model combination by stacking

- How to determine tuning parameters? (nr of leaves, regularization)
- Solve original problem, or decompose into subproblems?
- How to combine models for the subproblems?

Model combination by stacking

- How to determine tuning parameters? (nr of leaves, regularization)
- Solve original problem, or decompose into subproblems?
- How to combine models for the subproblems?

$\rightarrow$ Estimate all models and combine through *stacking*

Model combination by stacking

1. Estimate all models on k fold CV data

Model combination by stacking

1. Estimate all models on k fold CV data
2. Predict the data left out

Model combination by stacking

1. Estimate all models on k fold CV data
2. Predict the data left out
3. Concatenate the k predictions into vectors $\hat{y}_i$

Model combination by stacking

1. Estimate all models on k fold CV data
2. Predict the data left out
3. Concatenate the k predictions into vectors $\hat{y}_i$
4. Combine predictions linearly:

$$\min_w \mathcal{L}\left(\sum_{\text{models}} w_i \hat{y}_i\right) \quad \text{s.t. all } w_i \geq 0$$

Model combination by stacking

1. Estimate all models on k fold CV data
2. Predict the data left out
3. Concatenate the k predictions into vectors $\hat{y}_i$
4. Combine predictions linearly:

$$\min_{w} \mathcal{L}\Big(\sum_{\text{models}} w_i \hat{y}_i\Big) \quad \text{s.t. all } w_i \geq 0$$

5. Predict test set using models estimated on all training data

Model combination by stacking

1. Estimate all models on k fold CV data
2. Predict the data left out
3. Concatenate the k predictions into vectors $\hat{y}_i$
4. Combine predictions linearly:

$$\min_{w} \mathcal{L}\Big(\sum_{\text{models}} w_i \hat{y}_i \Big) \quad \text{s.t. all } w_i \geq 0$$

5. Predict test set using models estimated on all training data
6. Combine model predictions using optimal w

Result

Completed • $13,000 • 1,785 teams

Higgs Boson Machine Learning Challenge

Mon 12 May 2014 – Mon 15 Sep 2014 (58 days ago)

Dashboard

Private Leaderboard - Higgs Boson Machine Learning Challenge

This competition has completed. This leaderboard reflects the final standings.

See someone using multiple accounts? Let us know.

#	Δ1w	Team Name ‡ model uploaded * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑4	Gábor Melis ‡ *	3.80581	110	Sun, 14 Sep 2014 09:10:04 (-0h)
2	↓1	**Tim Salimans ‡ ***	**3.78913**	**57**	**Mon, 15 Sep 2014 23:49:02 (-40.6d)**
3	—	nhlx5haze ‡ *	3.78682	254	Mon, 15 Sep 2014 16:50:01 (-76.3d)
4	↑55	ChoKo Team	3.77526	216	Mon, 15 Sep 2014 15:21:36 (-42.1h)
5	↑23	cheng chen	3.77384	21	Mon, 15 Sep 2014 23:29:29 (-0h)

Incorporating physics knowledge

- Physics knowledge important because of difficulty of model search, and limited sample sizes

Incorporating physics knowledge

- Physics knowledge important because of difficulty of model search, and limited sample sizes
- Variables like MMC and SVFIT suboptimal: We need

$$\int \mathrm{P}(\text{label} = s|m)p(m|x)dm$$

rather than $\arg\max_m p(m|x)$.

Incorporating physics knowledge

- Physics knowledge important because of difficulty of model search, and limited sample sizes
- Variables like MMC and SVFIT suboptimal: We need

$$\int \mathrm{P}(\text{label} = s|m)p(m|x)dm$$

rather than $\arg\max_m p(m|x)$.
- Physics knowledge also important to ensure generalization to real data.

"CAKE" variable

- Team CAKE (Thomas Gillam, Christopher Lester, Damien George) came up with a variable modelling

$$C = \frac{p(x|H \to \tau\tau)}{p(x|Z \to \tau\tau)}$$

which is almost the desired likelihood ratio $p(x|\text{label} = s)/p(x|\text{label} = b)$

"CAKE" variable

- Team CAKE (Thomas Gillam, Christopher Lester, Damien George) came up with a variable modelling

$$C = \frac{p(x|H \to \tau\tau)}{p(x|Z \to \tau\tau)}$$

which is almost the desired likelihood ratio $p(x|\text{label} = s)/p(x|\text{label} = b)$

- Variable improved result for many teams, but also led to model instability
  - it made my public score worse
  - would have given me winning result on private leaderboard

“CAKE” variable

- Team CAKE (Thomas Gillam, Christopher Lester, Damien George) came up with a variable modelling

$$C = \frac{p(x|H \to \tau\tau)}{p(x|Z \to \tau\tau)}$$

  which is almost the desired likelihood ratio $p(x|\text{label} = s)/p(x|\text{label} = b)$
- Variable improved result for many teams, but also led to model instability
  - it made my public score worse
  - would have given me winning result on private leaderboard
- Reason for instability: variable C is a model in itself, should restrict $\partial p(\text{label} = s|x, C)/\partial C$ to be positive as in stacking

"CAKE" variable

- Team CAKE (Thomas Gillam, Christopher Lester, Damien George) came up with a variable modelling

$$C = \frac{p(x|H \to \tau\tau)}{p(x|Z \to \tau\tau)}$$

which is almost the desired likelihood ratio $p(x|\text{label} = s)/p(x|\text{label} = b)$

- Variable improved result for many teams, but also led to model instability
    - it made my public score worse
    - would have given me winning result on private leaderboard
- Reason for instability: variable C is a model in itself, should restrict $\partial p(\text{label} = s|x, C)/\partial C$ to be positive as in stacking
- Alternative: make CAKE model more flexible and optimize parameters on the data

Real data

- Simulated data $\neq$ reality
- Is our model picking up on errors in the simulation?
- Will results generalize to real data?

Real data

- Simulated data $\neq$ reality
- Is our model picking up on errors in the simulation?
- Will results generalize to real data?

- Need physics knowledge to restrict model structure (a la CAKE)

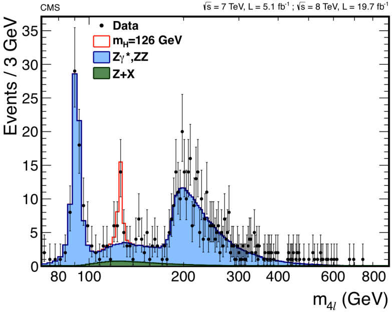
Real data

- Simulated data $\neq$ reality
- Is our model picking up on errors in the simulation?
- Will results generalize to real data?

- Need physics knowledge to restrict model structure (a la CAKE)
- Can use generative modelling techniques on real (unlabeled) data

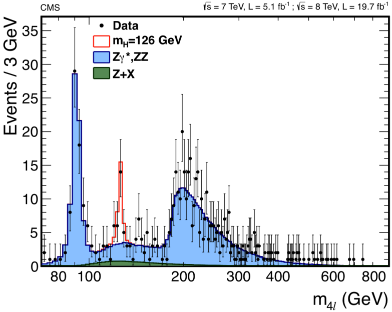
Generative modelling

Bump hunting = unsupervised learning / generative modelling



Generative modelling

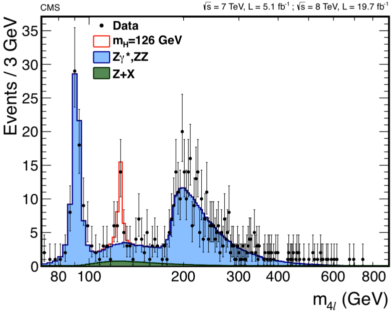
Bump hunting = unsupervised learning / generative modelling



- general idea: high density areas define unique physical events
  - prior knowledge: classification boundaries can only occur in low density areas

Generative modelling

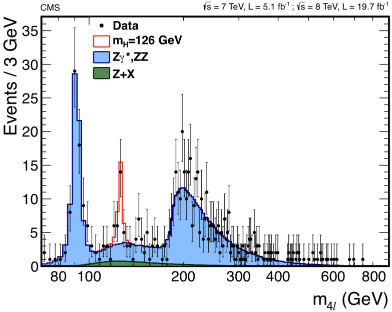
Bump hunting = unsupervised learning / generative modelling



- general idea: high density areas define unique physical events
  - prior knowledge: classification boundaries can only occur in low density areas
- we can also do this in higher dimensions

Generative modelling

Bump hunting = unsupervised learning / generative modelling



- general idea: high density areas define unique physical events
  - prior knowledge: classification boundaries can only occur in low density areas
- we can also do this in higher dimensions
- combine real unlabeled data with labeled (simulated) data: *semi-supervised learning*

Conclusion

- Machine Learning methods are a powerful tool for analyzing HEP data
  - supervised model faster & more accurate than MMC or SVFIT
  - crucially depends on accuracy of simulated data

Conclusion

- Machine Learning methods are a powerful tool for analyzing HEP data
  - supervised model faster & more accurate than MMC or SVFIT
  - crucially depends on accuracy of simulated data
- real promise lies in semi-supervised methods combining simulated data and real data

Conclusion

- Machine Learning methods are a powerful tool for analyzing HEP data
  - supervised model faster & more accurate than MMC or SVFIT
  - crucially depends on accuracy of simulated data
- real promise lies in semi-supervised methods combining simulated data and real data

XKCD, http://xkcd.com/1437/