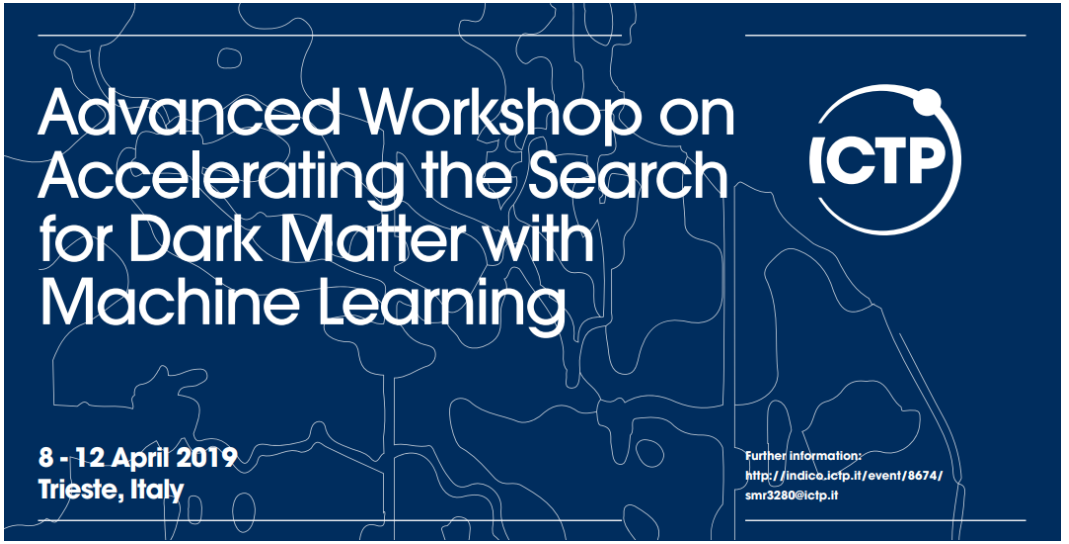


Noisy-input classification of Fermi-LAT unidentified point-like sources

Bryan Zaldivar
IFT/UAM Madrid

work in progress with:

Javier Coronado-Blázquez,
Viviana Gammaldi
Miguel A. Sánchez-Conde

Machine Learning Group
Escuela Politécnica Superior

Carlos Villacampa-Calvo,
Eduardo Garrido Merchán
Daniel Hernández-Lobato

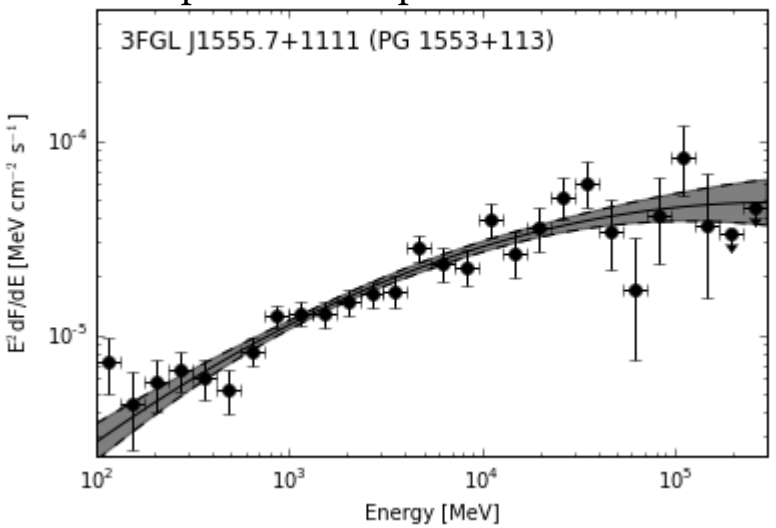
Motivation and Data origin

`4FGL` Fermi-catalog

circa 5000 point-like sources, out of which ~ 1500 are unidentified (unID)

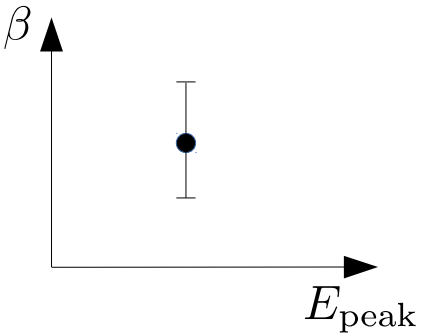
Can we classify those unIDs alla supervised learning ?

Spectrum of a particular blazar



`Log-parabola`

$$\frac{dN}{dE} = N_0 \left(\frac{E}{E_b}\right)^{-(\alpha+\beta\log(E/E_b))}$$



Available data contains 3 known classes:
pulsars, quasars, blazars

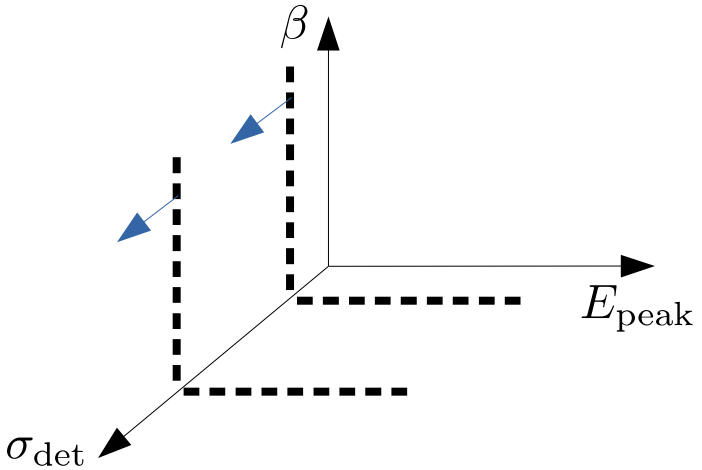
and 4 features: $E_{\rm peak}, \beta, \sigma_{\rm det}, \sigma_{\rm curv}$

- $\sigma_{\rm det}$: significance of detection
- $\sigma_{\rm curv}$: improvement in fitting log-parabola vs. power-law

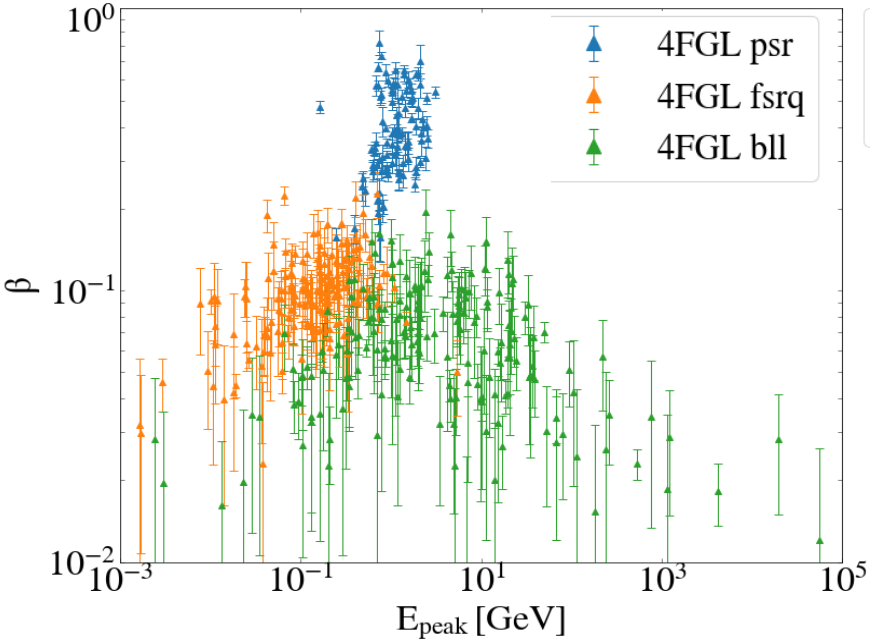
What if some of the unIDs are better classified as dark matter?

Include the dark matter into the β-plane!

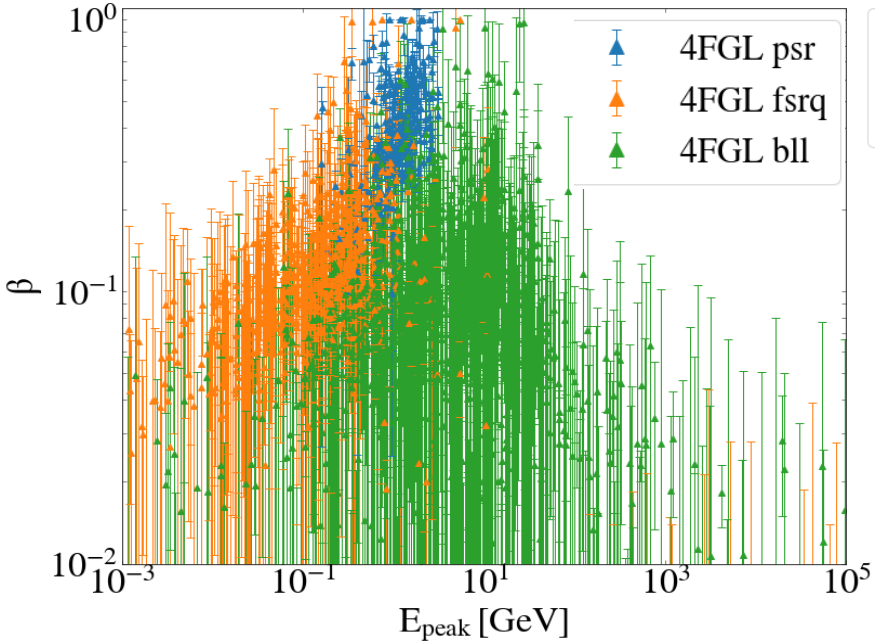
Data visualization



$\sigma_{\rm det} > 30$



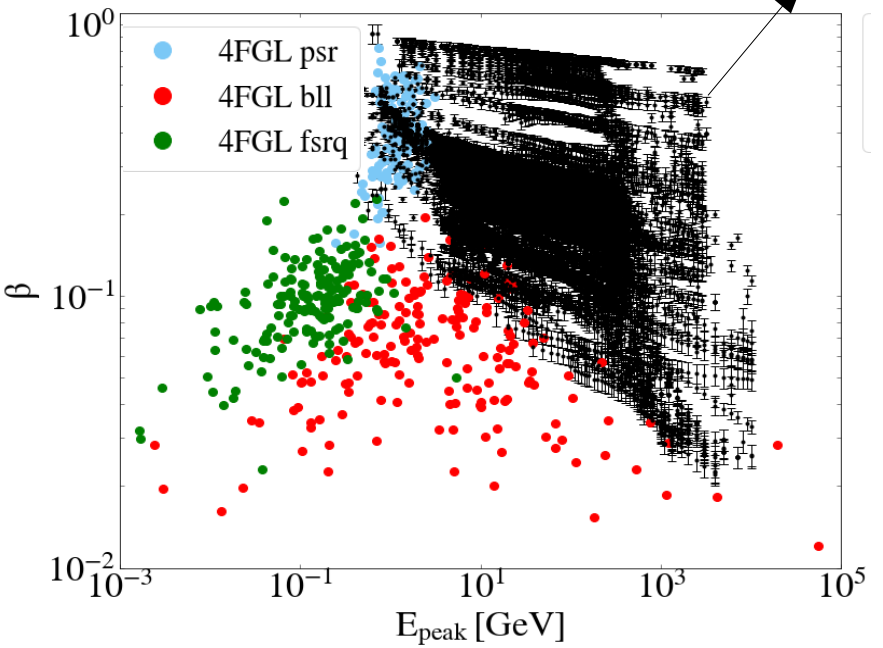
$\sigma_{\rm det} > 0$



credits to Javier Coronado-Blázquez

$\sigma_{\rm det} > 30$

dark matter class

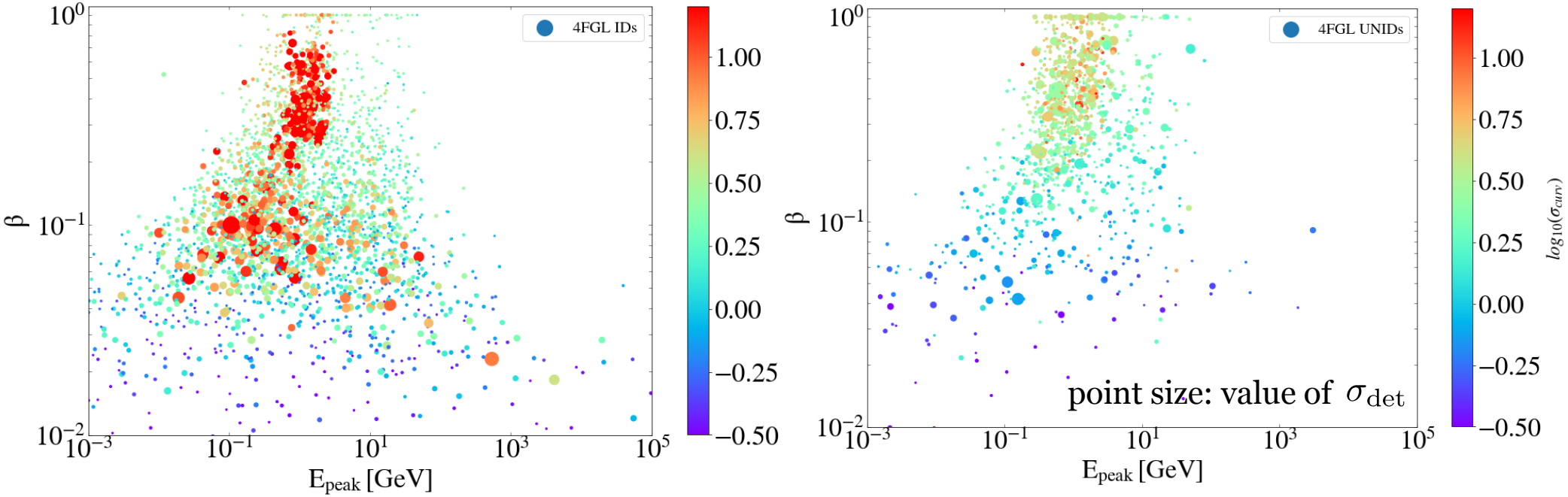


Data visualization

identified

credits to Javier Coronado-Blázquez

unidentified



- unID's seem to be distributed similarly to the ID's

- error bars on β partially correlated also with $\sigma_{\rm curv}$

Machine Learning procedure
Step # 1

Standard classification without input uncertainties

warm up:
want to know what the simplest thing to do can give you

Considered classifiers:

- `Naive Bayes`
- `Logistic regression`
- `Random Forest`

work in progress, but conceptually trivial...

Next steps:

- search for an out-of-the-box classifier dealing with noisy inputs ✗
- search for a paper addressing the classification with noisy inputs ✗
- call your ML-expert colleagues, ask them for references! ✗
- do it ourselves!! ✓

Machine Learning procedure
Step # 2:
incorporating input uncertainties

Bayesian classification with parametric models

$$\mathcal{D}_{\text{train}} = \{\mathbf{x}_i, y_i\}_{i=1}^{N} \qquad y_i = 1, 2, ..., K$$

t_{ik} : one-hot-encoding of y_i

Parametric models assume a specific form for the Likelihood of data

$$-\log p(\mathcal{D}_{\text{train}}|\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik} \log p(C_k|\mathbf{x}_i, \mathbf{w})$$

Cross-entropy

$$p(C_k|\mathbf{x}_i, \mathbf{w}) = \sigma_k(\ f(\mathbf{x}_i, \mathbf{w})\ ) \qquad \text{(softmax function)}$$

and assume a specific form for the function $f(\mathbf{x}_i, \mathbf{w})$ (e.g. a neural network)

In Bayesian approach, we build the predictive distribution for a new point $\mathbf{x}_*$

$$p(y_*|\mathbf{x}_*, \mathcal{D}_{\text{train}}) = \int p(y_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}_{\text{train}}) d\mathbf{w}$$

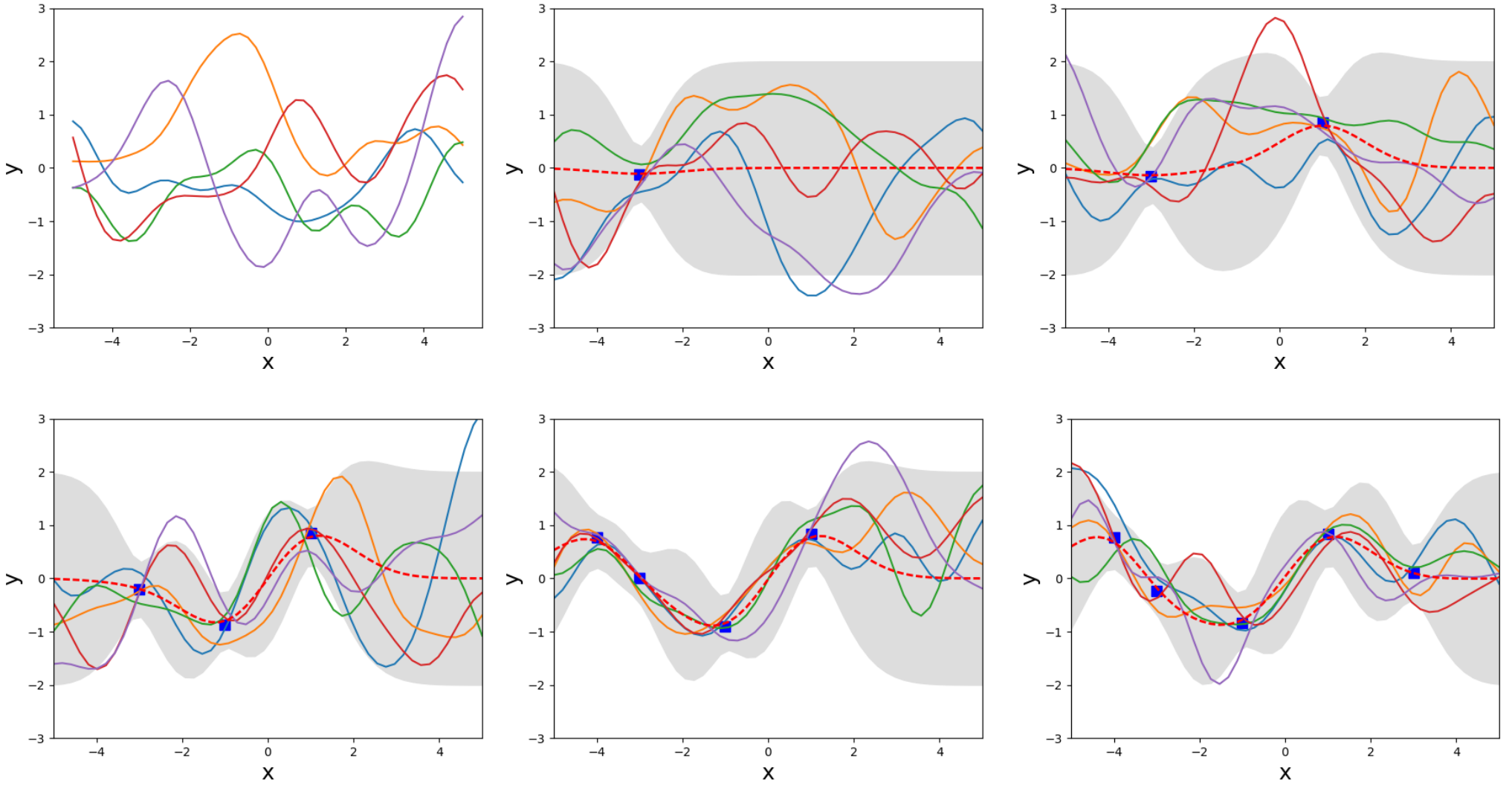
$$p(\mathbf{w}|\mathcal{D}_{\text{train}}) = \frac{p(\mathcal{D}_{\text{train}}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D}_{\text{train}})}$$

Gaussian Process

- Rasmussen & Williams, 2006

GP approach is non-parametric: no predefined form for $f(\mathbf{x}_i, \mathbf{w})$

Instead you have a Gaussian ***distribution over functions*** (in case of regression)



Classification with noisy input using Gaussian Processes

$$\mathcal{D}_{\text{train}} = \{\mathbf{x}_i + \epsilon_i, y_i\}_{i=1}^{N} \qquad y_i = 1, 2, ..., K$$

- As usual: introduce one output latent variable f_i^k per point *i* per class *k*,

- NEW: introduce one input latent variable $\tilde{\mathbf{x}}_i$ per point *i*

$$\mathbf{x}_i = \tilde{\mathbf{x}}_i + \epsilon_i$$

The predictive distribution for a class y_* at a test point $\mathbf{x}_*$

$$p(y_*|\mathbf{x}_*, \mathcal{D}_{\text{train}}) = \int p(y_*|\mathbf{f}_*) p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathcal{D}_{\text{train}}) p(\tilde{\mathbf{x}}_*|\mathbf{x}_*) \, d\mathbf{f}_* d\tilde{\mathbf{x}}_*$$

usual term

Gaussian posterior (new term)

$$p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathcal{D}_{\text{train}}) = \int d\mathbf{F} d\tilde{\mathbf{X}} \; p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathbf{F}) p(\mathbf{F}|\tilde{\mathbf{X}}, \mathbf{y}) p(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})$$

Costly: $\mathcal{O}(N^3)$ ➡ **Sparse GP** $\mathcal{O}(M^2 N)$

intractable ➡ Variational Inference

$$p(\mathbf{F}|\tilde{\mathbf{X}}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{F}, \tilde{\mathbf{X}}) p(\mathbf{F}|\tilde{\mathbf{X}})}{p(\mathbf{y})}$$

Non-Gaussian Likelihood

Sparse Gaussian Process

- here inspired in Titsias (2009)

$$p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathcal{D}_{\text{train}}) = \int d\mathbf{F} d\tilde{\mathbf{X}}\; p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathbf{F}) p(\mathbf{F}|\tilde{\mathbf{X}}, \mathbf{y}) p(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})$$

involves inverting an *N*x*N* matrix, **cost** $\mathcal{O}(N^3)$

Idea is to make inference on a smaller $M < N$ set of function points, which represent approximately the entire posterior over the *N* function points.

$$p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathcal{D}_{\text{train}}) = \int d\mathbf{F} d\tilde{\mathbf{X}} d\mathbf{U}\; p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathbf{F}, \mathbf{U}) p(\mathbf{F}, \mathbf{U}|\tilde{\mathbf{X}}, \mathbf{y}) p(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})$$

$$= \int d\mathbf{F} d\tilde{\mathbf{X}} d\mathbf{U}\; p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathbf{F}, \mathbf{U}) p(\mathbf{F}|\mathbf{U}, \tilde{\mathbf{X}}, \mathbf{y}) p(\mathbf{U}|\tilde{\mathbf{X}}, \mathbf{y}) \mathbf{p}(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})$$

$\mathbf{F}$: $\dim = N \times K$

$\mathbf{U}$: $\dim = M \times K$

If $\mathbf{U}$ were sufficient statistics for $\mathbf{f}_*$, we were left with

$$p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathcal{D}_{\text{train}}) \approx \int d\tilde{\mathbf{X}} d\mathbf{U}\; p(\mathbf{f}_*|\tilde{\mathbf{x}}_*, \mathbf{U}) p(\mathbf{U}|\tilde{\mathbf{X}}, \mathbf{y}) p(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})$$

Cost: $\mathcal{O}(NM^2)$

Variational Inference

- Jordan, Ghahramani, Jaakkola & Saul, 1999

Idea is approximate the exact posterior distribution by an easier one (e.g. Gaussians) according to the *variational principle*

$$p(\mathbf{F}, \tilde{\mathbf{X}}, \mathbf{U}|\mathbf{X}, \mathbf{y}) = p(\mathbf{U}|\mathbf{F}, \tilde{\mathbf{X}})p(\mathbf{F}, \tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{F}|\mathbf{U}, \tilde{\mathbf{X}})p(\mathbf{U})p(\mathbf{y}|\mathbf{F})p(\tilde{\mathbf{X}}|\mathbf{X}, \mathbf{y})}{p(\mathbf{y})}$$

$$q(\mathbf{F}, \tilde{\mathbf{X}}, \mathbf{U}) = p(\mathbf{F}|\mathbf{U}, \tilde{\mathbf{X}})q(\mathbf{U})q(\tilde{\mathbf{X}})$$

Minimize $\mathrm{KL}\left[q(\mathbf{F}, \tilde{\mathbf{X}}, \mathbf{U})||p(\mathbf{F}, \tilde{\mathbf{X}}, \mathbf{U}|\mathbf{X}, \mathbf{y})\right]$ w.r.t. $q(\mathbf{F}, \tilde{\mathbf{X}}, \mathbf{U})$

Kullback-Leibler divergence:

$$\mathrm{KL}[q(z)||p(z|x)] = \int dz\ q(z)\ln\left(\frac{q(z)}{p(z|x)}\right) \geq 0$$

Likelihood of the model

Common form in parametric models:

$$p(\mathcal{D}_{\text{train}}|\mathbf{w}) = \prod_{i=1}^{N}\prod_{k=1}^{K} \sigma_k(f(\tilde{\mathbf{x}}_i, \mathbf{w}))^{t_{ik}}$$

"generalized Bernoulli" (the -log of which is the cross entropy)

$\sigma_k(f(\tilde{\mathbf{x}}_i, \mathbf{w})) = p(y_i|f(\tilde{\mathbf{x}}_i, \mathbf{w}))$ e.g. if 3 classes:

$y_i = 1$	$y_i = 2$	$y_i = 3$
0.05	0.80	0.15

"misclassification noise"

Instead here

→ Misclassification noise included in the prior for $f_k(\tilde{\mathbf{x}}_i)$

→ Labelling noise (with probability e) also included:

Labelling rule: $y_i = \underset{k}{\text{argmax}}\ f_k(\tilde{\mathbf{x}}_i)$

Likelihood for label at point i : (noiseless)

$$p(y_i|\mathbf{f}_i) = \prod_{k \neq y_i} \Theta(f_{y_i}(\tilde{\mathbf{x}}_i) - f_k(\tilde{\mathbf{x}}_i))$$

$$p(y_i|\mathbf{f}_i) = (1 - e) \prod_{k \neq y_i} \Theta(f_{y_i}(\tilde{\mathbf{x}}_i) - f_k(\tilde{\mathbf{x}}_i)) + \frac{e}{K-1}\left[1 - \prod_{k \neq y_i} \Theta(f_{y_i}(\tilde{\mathbf{x}}_i) - f_k(\tilde{\mathbf{x}}_i))\right]$$

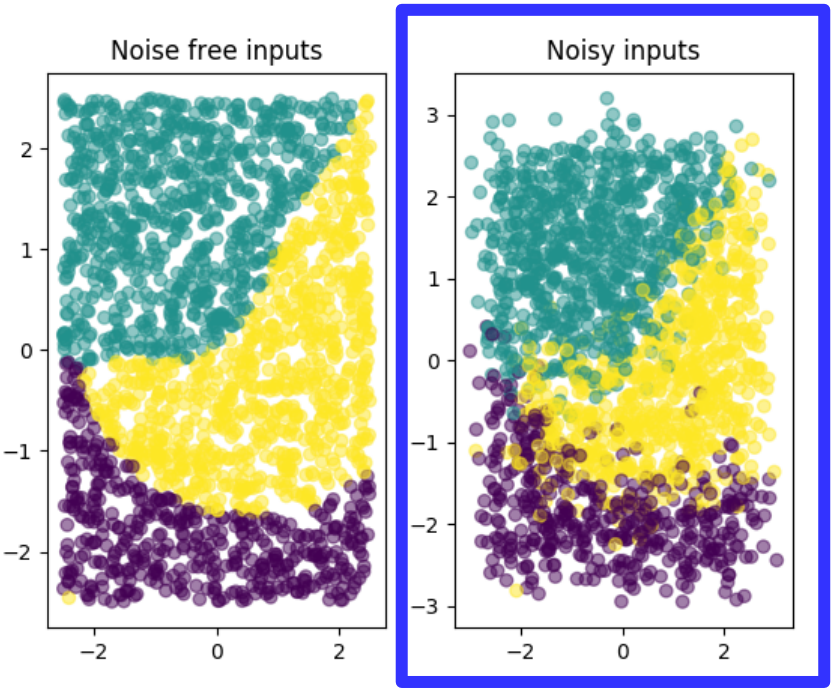
D. Hernandez-Lobato, J.M. Hernandez-Lobato & P. Dupont, 2011

Results

(python + TensorFlow)

Results on toy data

ex. of dataset



- Found no published model against which to compare!

  - we compare with a standard GP classif. without noise

  - we modify an existing GP noise model for regression

    McHutchon & Rasmussen, 2011

Generate a set (~100) of synthetic datasets
to evaluate average performance

	Input noise level 0.1		Input noise level 0.25		Input noise level 0.5	
	$-\ln\mathcal{L}_{\text{test}}$	Err. rate	$-\ln\mathcal{L}_{\text{test}}$	Err. rate	$-\ln\mathcal{L}_{\text{test}}$	Err. rate
Noiseless model	0.76	0.113	1.14	0.164	1.54	0.218
Rasmussen-like	0.321	0.109	0.53	0.158	0.77	0.209
This work	0.259	0.108	0.37	0.158	0.50	0.210

Conclusions/work in progress

- Unidentified point-like sources can be classified among predefined known classes (*including the potential dark matter class*)

- Interestingly, including the dark matter class into the well-known beta-plane for point-like sources results in a reasonably good separability

- Only non-straightforward issue with this problem:
  inputs come with their own error bars
  surprisingly not yet explicitly addressed in the context of ML classification!

- A warm-up classification exercise w/o error bars is being conducted

- Error bars are incorporated in a Gaussian Process model for multiclass classification, by treating the input as a noisy realization of extra latent variables to be learned.

- Very satisfactory preliminary results with synthetic data

- Time to apply it to real Fermi-LAT data!

Thank you!

bckp

Classification with error bars in the input (parametric approach)

Suppose you have data $\{\mathbf{x}_i \pm \Delta\mathbf{x}_i; \mathbf{t}_i\}_{i=1}^N$

$\dim(\mathbf{x}_i) = D$ $\quad$ $\mathbf{t}_i = \{t_{i1}, ..., t_{iK}\}$

"one-hot-encoding"

e.g. If $\mathbf{x}_i$ in class 2 $\quad K = 3$

$\mathbf{t}_i = \{0, 1, 0\}$

Are noisy samples from unknown means $\tilde{\mathbf{x}}_i$ $\quad$ assume $\mathcal{N}(\mathbf{x}_i|\tilde{\mathbf{x}}_i, \Delta\mathbf{x}_i)$

Then the (- log) joint Likelihood of data can be written as

$$-\log p(\mathbf{X}, \mathbf{T}|\tilde{\mathbf{X}}, \mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik} \log p(C_k|\tilde{\mathbf{x}}_i, \mathbf{w}) - \sum_{i=1}^{N} \log \mathcal{N}(\mathbf{x}_i|\tilde{\mathbf{x}}_i, \Delta\mathbf{x}_i)$$

$$p(C_k|\tilde{\mathbf{x}}_i, \mathbf{w}) = \sigma(f(\tilde{\mathbf{x}}_i, \mathbf{w}))$$

e.g. a linear model, or a NN model