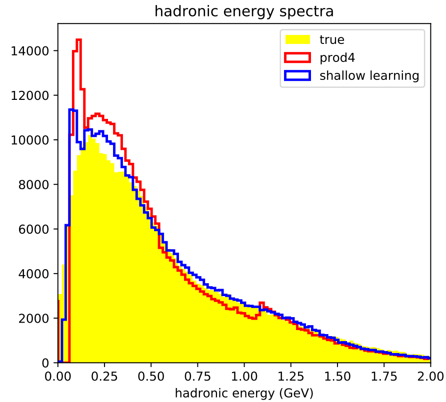
A Shallow Learning Hadronic Energy Estimator

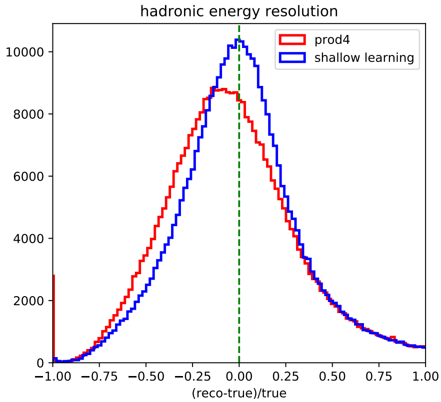
Shih-Kai Lin

Colorado State University

April 2, 2018

Some Teaser


hadronic energy spectra
true
prod4
shallow learning
14000
12000
10000
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
hadronic energy (GeV)


hadronic energy resolution
prod4
shallow learning
10000
8000
6000
4000
2000
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
(reco-true)/true

Motivation

- NOvA has put a lot of effort into PID (classification) with the state-of-the-art machine learning techniques, but not as much in energy reconstruction (regression).
  - Except CVN regression (UCI)
- Why one more attempt at energy reconstruction besides the current prong-based one (Erica, Michael) and CVN regression?
  - It is a natural generalization to the current official spline fit.
    - In the sense that it also uses event-level variables to fit a regression function.
  - It has welcoming mathematical properties and beautiful underlying theory.
  - The nice mathematical properties are reflected in the results.
  - Better tools! There are many CVN final state particle scores available at the moment.

Shallow Learning

- As opposed to deep learning. Some authors use this term in literature.
    - I personally like it due to my initials...
- Below is why this class of methods is called shallow learning in contrast to deep learning:

deep architecture	CNN	$\longrightarrow$	many hidden layers	$\longrightarrow$	classification regression
shallow architecture	support vector machine kernel ridge regression	$\longrightarrow$	one hidden layer (feature map)	$\longrightarrow$	classification regression

- A cohort of *kernel methods* belongs to shallow architecture, among which the support vector machine was so popular that it almost killed neural network in the early 2000s before CNN took the crown.
- I will quickly go through the ideas behind kernel methods to justify the use of them for an energy estimator.

Ridge Regression

Given N training samples $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^\ell$ are regressors and $y_i \in \mathbb{R}$ are targets, we want to find a linear function $f_\mathbf{w}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that minimizes the squared error loss function with L_2 regularization,

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2}_{\text{squared error}} + \underbrace{\alpha\|\mathbf{w}\|^2}_{\text{Tikhonov regularization}} \qquad (1)$$

, where α is a hyperparameter[1] that controls the degree of overfitting.

[1] A hyperparameter is a parameter whose value is set before the learning process begins.

Regression Function and Dual Form

Solution to 1 is

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{2}$$

, where $\mathbf{X}$ is the so called *design matrix*,

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} \tag{3}$$

$\mathbf{w}$ can be rewritten as

$$\mathbf{w} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \alpha\mathbf{I})^{-1}\mathbf{y} \tag{4}$$

With this *dual form*, given a test sample $\mathbf{x}_t$, the predicted value is

$$\hat{y}_t = \sum_{i=1}^{N} a_i \mathbf{x}_i^T \mathbf{x}_t \tag{5}$$

Here, $\mathbf{a} = (\mathbf{X}\mathbf{X}^T + \alpha\mathbf{I})^{-1}\mathbf{y}$, and $\mathbf{X}\mathbf{X}^T$ is a Gram matrix with elements $[\mathbf{X}\mathbf{X}^T]_{ij} = \mathbf{x}_i^T\mathbf{x}_j$.

Nonlinear Regression with Feature Map

A second order polynomial can be written as

$$f_{\mathbf{w}}(x) = w_0 + w_1 x + w_2 x^2 = \mathbf{w}^T \boldsymbol{\phi}(x) \tag{6}$$

With the *feature map* $\boldsymbol{\phi} : \mathbb{R} \to \mathbb{R}^3$, $\boldsymbol{\phi}(x) = (1, x, x^2)^T$, nonlinear regression in the *input space* $\mathbb{R}$ is equivalent to linear regression in the *feature space* $\mathbb{R}^3$.
Formula obtained with the linear case apply here, as long as $\mathbf{x}$ is replaced by $\boldsymbol{\phi}(\mathbf{x})$.

Kernel Trick

- Note that in the solution formula 5, every occurrence of a regressor $\mathbf{x}$ is accompanied by another regressor $\mathbf{x}'$ in the form of an inner product of the two.
- In the nonlinear case, it's $\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{x}')$.
  - Dimensionality of range of $\boldsymbol{\phi}$ becomes implicit and not important anymore.
- If we can find a kernel function $k : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ such that $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{x}')$, $\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^\ell$, then we can obtain the solution without actually performing the feature map, which is computationally heavy and sometimes even impossible (ex. infinite-dimensional feature space).
- Note that given a kernel the feature map and feature space are not unique.

The RBF (Gaussian) Kernel

One of the most commonly used kernels is the radial basis function (RBF), or Gaussian kernel:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} \tag{7}$$

For $\ell = 1$, a heuristic feature map of the RBF kernel is

$$\phi(x) = \underbrace{e^{-\gamma x^2}}_{\text{local}} \underbrace{\left(1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \dots\right)^T}_{\text{polynomial of all orders}} \tag{8}$$

, where we can see that the feature map of the RBF kernel is like local fit to polynomials of all orders.

- The RBF kernel works so well in many cases that it is usually one of the default kernels to try out.
  - I will use this kernel throughout the study.
- The hyperparameter γ controls how far the effect of a training sample can reach.

Representer Theorem and Kernel Ridge Regression

The solution function in the nonlinear problem to minimize a class of loss functions, including the squared loss with L_2 penalty, is

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{N} a_i k(\mathbf{x}_i, \mathbf{x})} \tag{9}$$

, where $\mathbf{a} = (\mathbf{K} + \alpha \mathbf{I})^{-1} \mathbf{y}$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

- Clearly a generalization to eq. 5.
- Ridge regression with kernel trick is the kernel ridge regression (KRR), one of the few machine learning algorithms with a closed form solution.
- I have tried support vector regression (SVR) as well. Since KRR works better, I will only show results from KRR.

Features of Kernel Methods

- Nonparametric model
  - Number of parameters grows with number of training samples.
  - In this case, it's the $\mathbf{a}$ vector.
- Unlike the *binned* spline fit, this is an *unbinned* fit.
- Positive definite kernels[2] make the loss function convex. Therefore, a global minimum is guaranteed.
  - Very different from neural networks.
- Functions drawn from the RBF kernel are *very smooth*.
  - No more kinks in the regression curve/surface/hypersurface.
- Including more variables is a no-brainer.
  - How to design a regression surface embedded in 3D after all? More dimension?
  - Opens up "feature engineering".

[2]Most commonly used kernels belong to this class, including RBF, but not sigmoid.

Hands-on!

Time to get hands dirty:

- datasets
  `prod_caf_R17-11-14-prod4reco.d_nd_genie_nonswap_fhc_nova_v08_period3_v1`
- cuts
  `kNumuCutND2018&&kIsNumuCC`
- weights
  - No weight for the proof of concept rounds
  - For the newest results, `kXSecCVWgt2018*kPPFXFluxCVWgt`
- variables
  regressor
  - `kNumuHadVisE` for first attempts
  - later add CVN particle final state scores for p, n, π^0, $\pi^{\pm}$, and number of prongs

  target
  - always `kTrueE` (true neutrino energy) - `kMuE` (prod4 reco muon energy)

1D regressor (E_{vishad}), 0.5% total statistics