Optimization of tau identification in ATLAS experiment using multivariate tools

Marcin Wolter\textsuperscript{a}, Andrzej Zem\l a\textsuperscript{a,b}

\textsuperscript{a}Institute of Nuclear Physics PAN, Kraków
\textsuperscript{b}Jagiellonian University, Kraków

Physics processes with $\tau$’s

**Higgs Processes**:  
- **Standard Model Higgs (VBF, ttH)**  
  $qqH \rightarrow qq\tau\tau$, $ttH \rightarrow tt\tau\tau$
- **MSSM Higgs (A/H, H$^+$)**  
  $A/H \rightarrow \tau\tau$, $H^+ \rightarrow \tau\nu$

**Exotic Processes**:  
- **SUSY** signature with $\tau$’s in final state  
- **Extra dimensions** ... new theories (?)

**Standard Model**:  
- $Z \rightarrow \tau\tau$  
- $W \rightarrow \tau\nu$, important for first physics data analysis

**Observed: $\tau$ jets**  
- characteristics  
  well-collimated calorimeter cluster with a small number of associated charged tracks
- main background QCD jets

**Two algorithms used by ATLAS for $\tau$ jet reconstruction**:  
- **TauRec** - cluster based  
- **Tau1P3P** – track-based

08.02.2007

M. Wolter, A. Zemła
Tau1p3p algorithm

Algorithm for τ hadronic jet reconstruction and identification

- Starts from a good quality hadronic track \( p_T > 9 \) GeV
- Find nearby good quality tracks \( p_T > 1 \) GeV, \( ΔR < 0.2 \)
  - 1 track + n \( π^0 \) (1 Prong candidate)
  - 3 tracks + n \( π^0 \) (3 Prong candidate)
  - 2 tracks + n \( π^0 \) (2 Prong candidate)
  - 3P and one track lost
  - 1P and one fake

Signal and background separation

- Based on tracking and calo information
- 2 Prong & 3 Prong – 11 id. variables
- 1 Prong -- 9 id. variables
- None of them alone provides a good separation
- Smart (multivariate) analysis needed!
Multivariate tau identification

  Decorrelation -> many random cuts applied -> optimal set of cuts chosen.

- **PDE_RS** – Probability Density Estimator with Range Searches.
- **Neural network** – Stuttgart Neural Network Simulator (SNNS), feed-forward NN with two hidden layers. Trained network converted to the C code – very fast.
- **Support Vector Machine** – new tool in the TMVA package (implemented by us), now for the first time applied to the data analysis.

- Analysis performed on ATLAS MC data.
  
  **Signal:** \( Z\rightarrow\tau\tau, \ W\rightarrow\tau\nu(\text{had}) \)
  **Background:** QCD jet events
PDE_RS method


• Parzen estimation (1960s) – approximation of the unknown probability as a sum of kernel functions placed at the points $x_n$ of the training sample.

• Make it faster - count signal ($n_s$) and background ($n_b$) events in N-dim hypercube around the event classified – only few events from the training sample needed (PDE_RS).

• Hypercube dimensions are free parameters to be tuned.

• Discriminator $D(x)$ given by signal and background event densities:

$$D(x) = \frac{n_s}{n_s + n_b}$$

• Events stored in the binary tree – easy and fast finding of neighbor events.

  • Implementation developed by E. Richter-Wąs & L. Janyst
Support Vector Machine (SVM)

- **Basic ideas:**
  - Algorithm learning on examples.
  - Builds a separating hyperplane using the minimal set of vectors from the training sample (**Support Vectors**).
  - Is able to model any arbitrary function.

- **Functionally similar to Neural Networks.**

- **History:**
  - 1995: regression SVM to estimate the continuous function (Vapnik 1995)
Linear SVM classifier

\[ y_i = f(x, w, b) = \text{sign}(\langle w, x \rangle - b) \]

- \( y_i = +1 \)
- \( y_i = -1 \)

Point defining the margin:

\[ y_i (\langle \tilde{w}, \tilde{x}_i \rangle + b) - 1 \geq 0 \]

Support vectors

Margin maximization:
- Intuitively the best.
- Not sensitive for small errors.
- Separation defined by support vectors only.

Non-separable data - additional penalty function:

\[ \text{minimize:} \frac{1}{2} |\tilde{w}|^2 + C \sum_i \xi_i \]

Quadratic programming problem – a single minimum

08.02.2007
Non-linear extension

- Transformation from the input space into the high-dimensional feature space makes the LINEAR separation by a hyperplane possible:

$$\Phi(x) : \text{input } \mathbb{R}^n \rightarrow \mathbb{R}^N \quad (N \geq n)$$

- The classifier depends only on the scalar products of the vectors: $$\langle x_i, x_j \rangle$$.

- Therefore we do not need to know explicitly the function $$\Phi$$, it is sufficient to know the kernel function $$K$$:

$$\langle x_i, x_j \rangle \rightarrow \langle \Phi(x_i), \Phi(x_j) \rangle = K(x_i, x_j)$$

- Commonly used kernel:
  
  - **polynomial**
    
    $$K(\tilde{x}_i, \tilde{x}_j) = (\langle \tilde{x}_i, \tilde{x}_j \rangle + c)^d$$
  
  - **sigmoid**
    
    $$K(\tilde{x}_i, \tilde{x}_j) = \tanh(\kappa \langle \tilde{x}_i, \tilde{x}_j \rangle + \theta)$$
  
  - **Gaussian**
    
    $$K(\tilde{x}_i, \tilde{x}_j) = \exp\left(\frac{-1}{2\sigma^2}\|\tilde{x}_i - \tilde{x}_j\|^2\right)$$
Why SVM?

- **Advantages:**
  - Statistically well motivated.
  - Minimization is a quadratic programming problem, always finds minimum.
  - Only two parameters to tune despite the dimensionality (in case of Gaussian kernel).
  - Insensitive for overtraining.

- **Limitations:**
  - Slow training (compared to neural network) due to computationally intensive solution to QP problem. We have used J. Platt's SMO (Sequential Minimal Optimization) to speed it up.
  - For complex problems even 50% of input vectors can be the support vectors.
Discriminant distributions

PDE_RS

Neural Network

SVM

3 Prong events

- Good signal and background separation.
Bkg. rejection vs. signal efficiency – ROC curve (identification only)

1-eff

15 GeV < E_T < 25 GeV

- NN, PDE_RS and SVM significantly better than cuts.
- SVM is trained on a small subsample of all MC events (few thousands) – good performance with a limited training sample
- Best performance achieved with Neural Networks.

Algorithms trained on a subsample of all data, tested on the remaining events.

08.02.2007

M. Wolter, A. Zemla
Summary

- Tau1p3p identification significantly improved by using multivariate analysis tools.
- All of the presented classification methods are performing well:
  - **Cuts** – fast, robust, transparent for users.
  - **Neural network** – best performance, very fast classification while converted to the C function after training.
  - **PDE_RS** – robust and transparent for users, but large samples of reference candidates needed, classification slower than for other methods.
  - **Support Vector Machine** – not commonly used in HEP, new implementation of the algorithm (beta release). It works, gives competitive results. Tests show, that only small data samples are needed for training.
Energy-scale: 
- $e/\gamma \sim 0.1\%$ muons
- $\sim 0.1\%$ Jets

Muon Detectors: fast response for trigger, good $p$ resolution

Electromagnetic Calorimeters: excellent $e/\gamma$ identification, $E$ and angular resolution, response uniformity

Hadron Calorimeters: Good jet and $E_T$ miss performance

Inner Detector: high efficiency tracking, good impact parameter resolution

• 40MHz beam crossing
• Readout: 160M channels (3000 km cables)
• Raw data = 320Mbyte/sec (1TB/hour)
How candidates are defined: tau1P, tau2P, tau3P

- **tau1P (Ntrack=1):** exactly one leading good quality track, no more good quality tracks within $R^{\text{core}}$

- **tau2P (Ntrack=2):** exactly two good quality tracks within $R^{\text{core}}$, one of them leading, no more associated tracks within $R^{\text{core}}$; by definition $E_T^{\text{flow}}$ not correct here as one track is missing for true three-prong.

- **tau3P (Ntrack=3):**
  - exactly three good quality tracks within $R^{\text{core}}$ (one of them leading)
  - exactly two good quality tracks within $R^{\text{core}}$ (one of the leading)
  - + one non-qualif track within $R^{\text{core}}$

- **tau3P (Ntrack=4,5):** exactly three good quality tracks within $R^{\text{core}}$, + 1-2 non-qualif tracks within $R^{\text{core}}$

From Elzbieta's talk 22.08.2006
Identification variables

- Few new quantities introduced, full set given below:
  - tracking part:
    → number of associated tracks in isolation cone \((NEW!)\)
    → weighted width of track with respected to tau axis
      (for tau2P, tau3P only) \((NEW!)\)
    → invariant mass of tracks (for tau2P, tau3P only) \((NEW!)\)
  - calorimetry part:
    → number of hitted strips
    → energy weighted width in strips
    → fraction of energy in half core cone to energy in core cone
    → energy weighted radius in EM part
  - calor+tracking part
    → fraction of energy in isolation cone to energy in core cone, at EM scale
    → invariant mass calculated from energy-flow \((NEW!)\)
    → ratio of energy in HAD part to energy of tracks

- Common vertex or life-time information not used yet.
- Electron-track and muon-track veto still from truth.

From Elzbieta's talk 22.08.2006
Implementation of NN

- The trained network is converted into C function and integrated to the ATHENA package.
- To get a flat acceptance in $E_{\text{vis}}$ and flat distribution of NN output the compensation procedure is applied.
- The compensation function is added to the NN function code.
- Two variables are returned by the function:
  - DiscriNN – “raw” NN output
  - EfficNN -- NN output with compensation

Corrected efficiency

Corrected NN output
Support Vector Machine
Support Vector Machines

- Relatively new algorithm

- Basic idea:
  - build a separating hyperplane using the minimal number of vectors from the training sample (Support Vectors).
  - should be able to model any arbitrary function.

- Functionally algorithm similar to Neural Network
Support Vector Machine

• **Early sixties:** – “support vectors” method developed for pattern recognition using the separating hyperplane (Vapnik and Lerner 1963, Vapnik and Czervonenkis 1964).

• **Early 1990:** method extended to the non-linear separation (Boser 1992, Cortes and Vapnik 1995)

• **1995:** further development to perform regression (Vapnik 1995)
Linear classifier

- means $y_i = +1$
- means $y_i = -1$

Support Vectors
- points defining the margin.

$\forall i \ y_i (\hat{w} \cdot \hat{x}_i + b) - 1 \geq 0$
$\hat{w} \cdot \hat{x} + b = 0$ straight line

$margin = \frac{2}{|\hat{w}|}$

$\minimize |\hat{w}|^2$

Maximum margin linear classifier

- Intuitively is the best.
- Not sensitive on errors in the hyperplane position.
- Separation depends on support vectors only.
- Works in practice!

Maximize margin = minimize $|w|^2$
If data are not separable

- Additional slack variable:
  \[ \xi_i = \begin{cases} 0 & \text{correctly classified} \\ \text{distance} & \text{incorrectly classified} \end{cases} \]

- And we get:
  \[ \forall \ y_i (\hat{w} \cdot \hat{x}_i + b) - 1 + \xi_i \geq 0 \]

- For the linear classifier:
  \[ \hat{w} \cdot \hat{x} + b = 0 \]

\[
\text{minimize: } \frac{1}{2} |\hat{w}|^2 + C \sum_i \xi_i
\]

- C is a free parameter - cost variable.
Problem – data separable but non-linear

Transformation to higher dimensional feature space, where data ARE linearly separable.

In this example data separable by elliptic curve in $\mathbb{R}^2$ are linearly separable in $\mathbb{R}^3$.

We need a universal method of transformation to the higher dimensional space!
Kernel trick

- Margin $w$ maximization using Laplace multipliers $\alpha_i$:
  $$W(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \vec{x}_i, \vec{x}_j \rangle$$

- Margin depends on **dot products** $x_i^* x_j$ only.

- Function $\Phi$ transforms the input space to the higher dimensional feature space, where data are linearly separable:
  $$\Phi(x): \text{input } \mathbb{R}^n \rightarrow \mathbb{R}^N \ (N \geq n)$$

  - We can replace:
    $$\langle x_i, x_j \rangle \rightarrow \langle \Phi(x_i), \Phi(x_j) \rangle = K(x_i, x_j)$$

- The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.
  $$W(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

- **No need to know function** $\Phi(x)$, **enough to know kernel** $K(x_i, x_j)$. 
Commonly used kernels

- Polynomial
  \[ K(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d \]
- Sigmoid
  \[ K(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \theta) \]
- Gaussian
  \[ K(x_i, x_j) = \exp \left( - \frac{1}{2\sigma^2} \| x_i - x_j \|^2 \right) \]

Must be symmetric:

\[ K(x_i, x_j) = K(x_j, x_i) \]

If the kernel used is a radial base function (Gaussian) the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results.
Regression – “ε insensitive loss”

We have input data:

\[ X = \{(x_1, d_1), \ldots, (x_N, d_N)\} \]

We want to find \( f(x) \), which has small deviation from \( d \) and which is maximally smooth.

- Define a cost function:

\[
|y - f(x)|_\varepsilon := \max\{0, |y - f(x)| - \varepsilon\}
\]

- Minimize:

\[
\frac{1}{2}\|w\|^2 + \frac{C}{m} \sum_{i=1}^{m} |y_i - f(x_i)|_\varepsilon
\]

- And repeat the kernel trick
Non-linear Kernel example

**Example: SVM with RBF-Kernel**

**Gaussian kernel**

Kernel: \[ K(\hat{x}_i, \hat{x}_j) = \exp\left(-\frac{|\hat{x}_i - \hat{x}_j|^2}{\sigma^2}\right) \]

plot by Bell SVM applet
SVM and feed-forward neural network
A comparison

**NN**  – complexity controlled by a number of nodes.

**SVM**  – complexity doesn't dependent on dimensionality.

**NN**  – can fall into local minima.

**SVM**  – minimization is a quadratic programming problem, always finds minimum.

**SVM**  – discriminating hyperplane is constructed in a high dimensionality space using a kernel function.
SVM strength

• Statistically well motivated => Can get bounds on the error, can use the structural risk minimization (theory which characterizes generalization abilities of learning machines).

• Finding the weights is a quadratic programming problem - guaranteed to find a minimum of the error surface. Thus the algorithm is efficient and SVM generates near optimal classification and is quite insensitive to overtraining.

• Obtain good generalization performance due to high dimension of the feature space.

Jianfeng Feng, Sussex University
SVM weakness

• Slow training (compared to neural network) due to computationally intensive solution to QP problem especially for large amounts of training data => need special algorithms.

• Slow classification for the trained SVM.

• Generates complex solutions (normally > 60% of training points are used as support vectors), especially for large amounts of training data.

E.g. from Haykin: increase in performance of 1.5% over MLP. However, MLP used 2 hidden nodes, SVM used 285

• Difficult to incorporate prior knowledge.

Jianfeng Feng, Sussex University
Applications in physics

• While neural networks are quite commonly used, SVM are rarely applied.
  • LEP compared NN and SVM
    “Classifying LEP Data with Support Vector Algorithms”
    P. Vannerema K.-R. Muller B. Scholkopf A. Smola S. Soldner-Rembold
  • There were some tries in D0 and CDF (Tufts Group – top quark identification)
• SVM works well in other areas (for example handwriting recognition).
• One should look carefully, the method might be worth trying!