Machine Learning with Support Vector Machines

An (HEP oriented) Introduction

Dirk Krücker

February 18, 2016

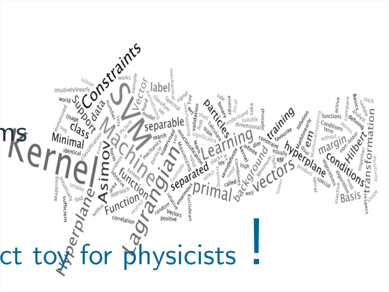
SVM – Basics

SVM – Basics

I will mention:

- Lagrangian minimisation
- Infinite dimensional Hilbert spaces
- Operators with complete eigen-systems

Is this going to be a QM lecture ?



Support Vector Machines are the perfect toy for physicists !

They appeals to your geometric intuition and can be understood as an “mechanical” device

SVM – Basics

Support Vector Machines are

- A successful ML technique
- Conceived in the 60th
  (Vapnik and Chervonenkis 1964)
- Final shape in the 90th
  (Cortes and Vapnik 1995, etc.)
- and widely used since then

- A Google search for "Support Vector Machine" results in $\approx$ 1.9 million hits
- Dozens of youtube videos
- Thousands of papers, tutorials, lectures

E.g.:

Wikipedia! SVM, KKT SMO etc.

- `http://www.support-vector-machines.org`
- `http://www.svms.org`
- `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

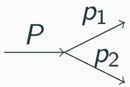
our favourite library $\Rsh$

But why is it not popular in HEP? (Somewhat limited TMVA implementation. The most important aspect of an SVM is not the training but the parameter tuning!)

Machine learning
What's that all about? ■●▲

Machine learning – What's that all about?

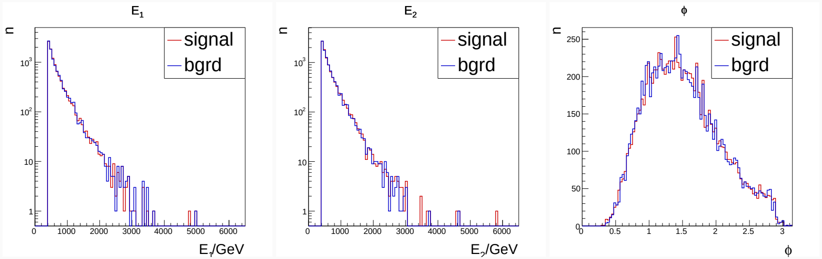
We need a toy example:



- Assume we have 1 particle decaying into 2 particles
  - E.g. a 750 GeV particle decays into 2 massless particles that we observe in our favourite detector
- The heavy particle does have some varying z-momentum in the lab-frame
- There are many other background particles
- We measure (lab system) the energies and the relative angle: E_1,E_2 and ϕ
  - Some cuts to select high energetic particles, e.g.: $E > 400$ GeV cut

Toy Example – Invariant mass implied correlations

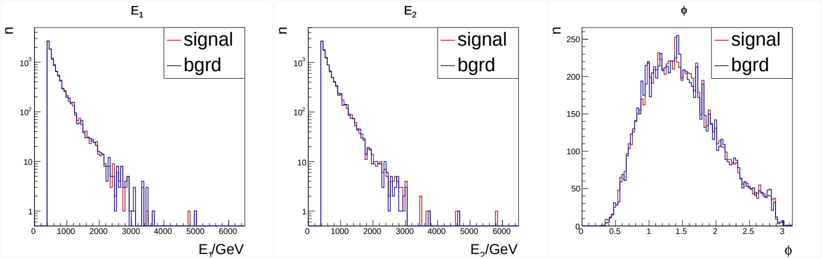
Invariant mass implied correlations with a wicked background:



- The signal and background distribution for E_1, E_2 and ϕ are identical in this example

Toy Example – Invariant mass implied correlations

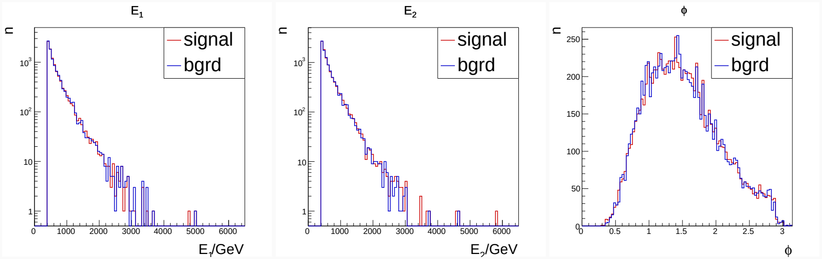
Invariant mass implied correlations with a wicked background:

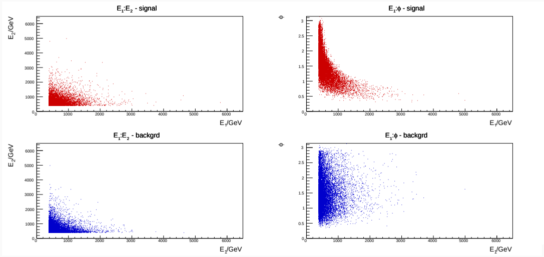


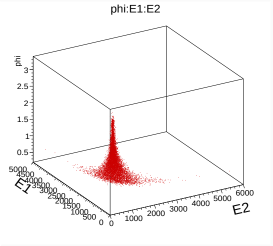
- The signal and background distribution for E_1, E_2 and ϕ are identical in this example
- The "background" had been created by resampling from the E_1,E_2 and ϕ distribution but with mixing particles from different events $\Longrightarrow$ correlation destroyed!

Toy Example – Invariant mass implied correlations

Invariant mass implied correlations with a wicked background:

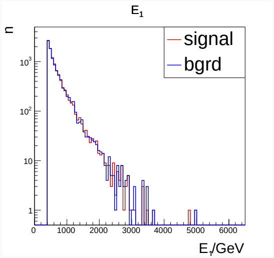


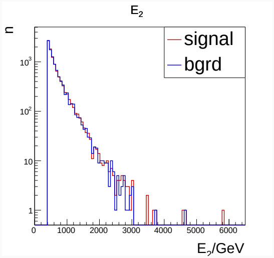


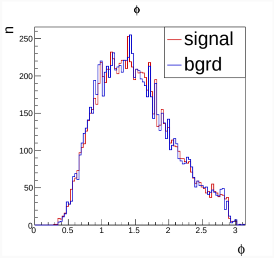


Toy Example – Invariant mass implied correlations

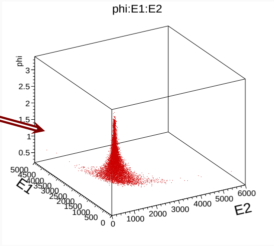
Invariant mass implied correlations with a wicked background:

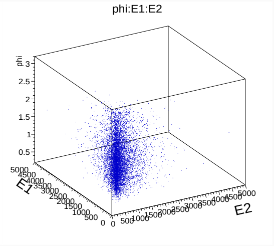




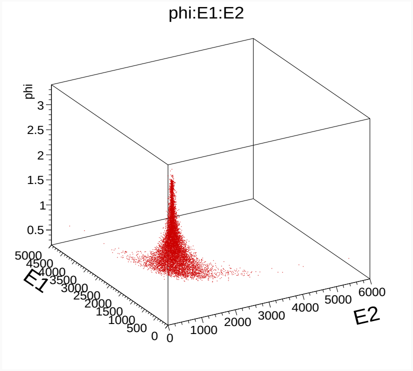


Identical marginal distribution but visible correlation in signal 3d-plot





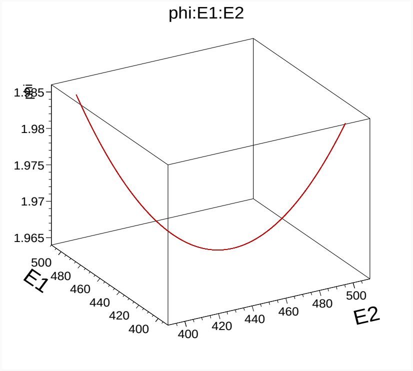
Toy Example – Invariant mass implied correlations



The used minimalistic toy "Monte Carlo":

$$\begin{tikzcd}
& \boxed{P_z \leftarrow \exp(P_Z/\tau)} \arrow[d] & \\
& \boxed{\text{isotropic (M=750GeV) 2-body decay}} \arrow[dl] \arrow[dr] & \\
\boxed{E_1 \pm 10\%} & & \boxed{E_2 \pm 10\%}
\end{tikzcd}$$

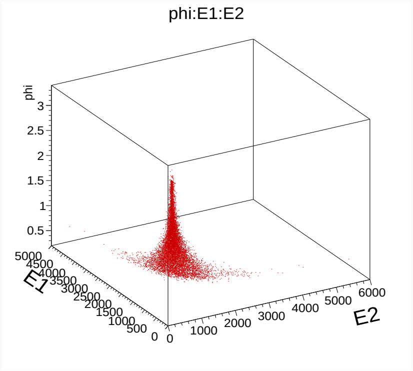
Toy Example – Invariant mass implied correlations

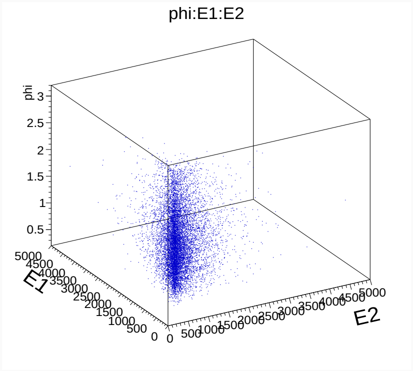


The used minimalistic toy "Monte Carlo":

$$\begin{tikzcd} \cancel{P_z \leftarrow \exp(P_Z/\tau)} \arrow[d] & \\ \boxed{\text{isotropic (M=750GeV) 2-body decay}} \arrow[d] \arrow[dr] & \\ \cancel{E_1 \pm 10\%} & \cancel{E_2 \pm 10\%} \end{tikzcd}$$

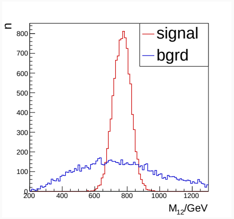
Toy Example – Invariant mass implied correlations





Strategies

Of course, as physicists we immediately 😎 understand that the long story about correlations is just about the invariant mass of the mother particle and calculate the invariant mass $M_{12} = \sqrt{2E_1E_2(1-\cos(\phi))}$



The way of the physicists

- Understand the nature of the problem
- Apply theory
  - Here: SR i.e. kinematics

$\Rightarrow$ **High level variables**

The way of the data analyst

- Take the data – the more the better
- Run complicated algorithms to find pattern in your data

$\Rightarrow$ **Low level variables**

SVM – Introduction

Input for Supervised Learning

- A set of n variables (in ML called features)
  - $\vec{\mathbf{x}} = (x^{(1)}, \dots, x^{(n)})$
- We assume them to form an Euclidean vector space. $\vec{\mathbf{x}} \in \mathbb{R}^n$
  - That's not trivial! We lump together totally different things e.g. b-tag estimators, energies, pixel-counts etc.; without a common scale or unit but we calculate distances
- In HEP the properties of signal and background are encoded in Monte Carlo programs
  - Two sets of trainings data: $N_{sig} + N_{bgrd} = N$ events
  - The training data gets a label: $y_i \in \{+1, -1\}$
    signal | background
  - A ML algorithm is called supervised if the class membership of all training vectors is known and used
- The training data is a set of labeled vectors:

$$\boxed{(y_1, \vec{\mathbf{x}}_1), (y_2, \vec{\mathbf{x}}_2), ..., (\vec{\mathbf{x}}_i, y_i), ..., (y_N, \vec{\mathbf{x}}_N)}$$

Terminology

- We have a classification problem

⇒ The aim is that the ML algorithm learns to decide between the classes

- We do supervised learning

⇒ We train the ML algorithm on data where we know the class membership

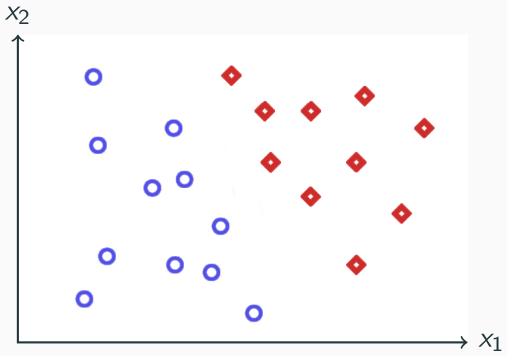
- The training data consists of a set of labeled vectors

⇒ $(y_1, \vec{\mathbf{x}}_1), (y_2, \vec{\mathbf{x}}_2), ..., (\vec{\mathbf{x}}_i, y_i), ..., (y_N, \vec{\mathbf{x}}_N)$

- The $\vec{\mathbf{x}}$ vector components are called features

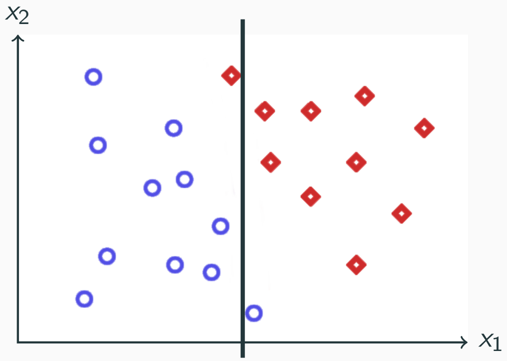
SVM – Intuitively

- Let's assume we have a 2-dimensional problem $\vec{\mathbf{x}} = (x_1, x_2)$
- Signal and background training data points form 2 separate cluster



SVM – Intuitively

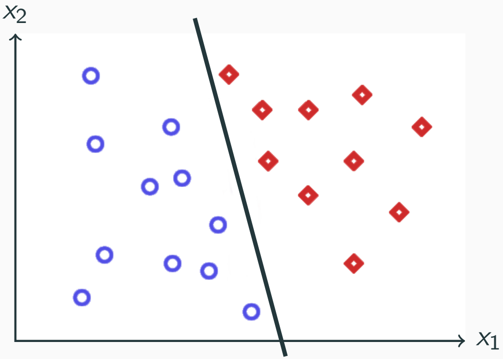
- Let's assume we have a 2-dimensional problem $\vec{\mathbf{x}} = (x_1, x_2)$
- Signal and background training data points form 2 separate cluster
- A simple 1d-cut is not sufficient



SVM – Intuitively

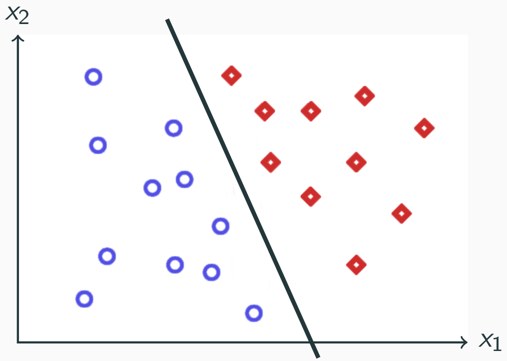
- Let's assume we have a 2-dimensional problem $\vec{\mathbf{x}} = (x_1, x_2)$
- Signal and background training data points form 2 separate cluster
- A simple 1d-cut is not sufficient
- There is a separating line

  $\Rightarrow$ The problem is linearly separable



SVM – Intuitively

- Let's assume we have a 2-dimensional problem $\vec{\mathbf{x}} = (x_1, x_2)$
- Signal and background training data points form 2 separate cluster
- A simple 1d-cut is not sufficient
- There is a separating line
  $\Rightarrow$ The problem is linearly separable
- Which separating line is the best?

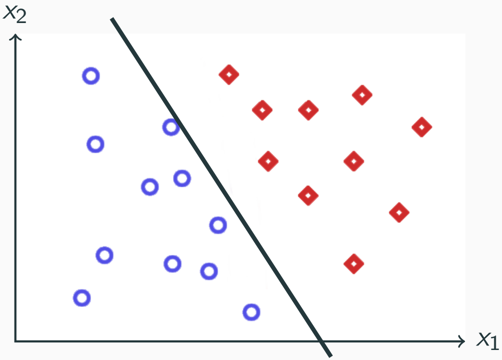


SVM – Intuitively

- Let's assume we have a 2-dimensional problem $\vec{\mathbf{x}} = (x_1, x_2)$
- Signal and background training data points form 2 separate cluster
- A simple 1d-cut is not sufficient
- There is a separating line

  $\Rightarrow$ The problem is linearly separable
- Which separating line is the best?

  What do we mean by "best"?

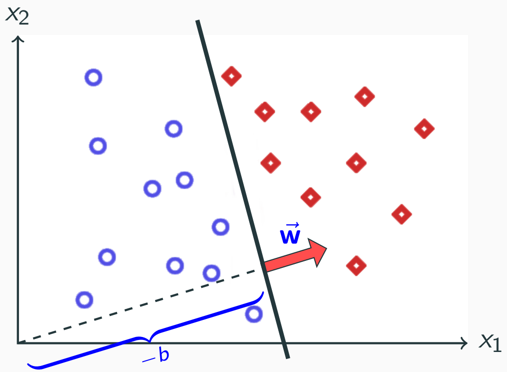


SVM – Intuitively

- A separating hyperplane in $\mathbb{R}^n$ can be described by offset b and normal vector $\vec{\mathbf{w}}$

$$\boxed{\vec{\mathbf{w}} \cdot \vec{\mathbf{x}} + b = 0}$$

- Scanning over all possible b and $\vec{\mathbf{w}}$ allows to find the optimal hyperplane

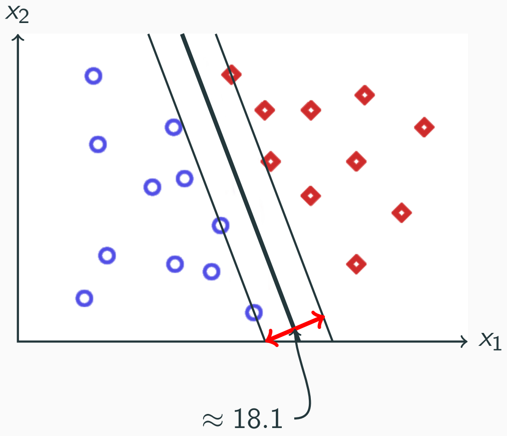


- Intuitively we would prefer the one with the largest margin

SVM – Intuitively

We train the SVM on a random sample from some unknown distribution. The best we can do is to put a hyperplane in between the two cluster such that the margin becomes maximal.

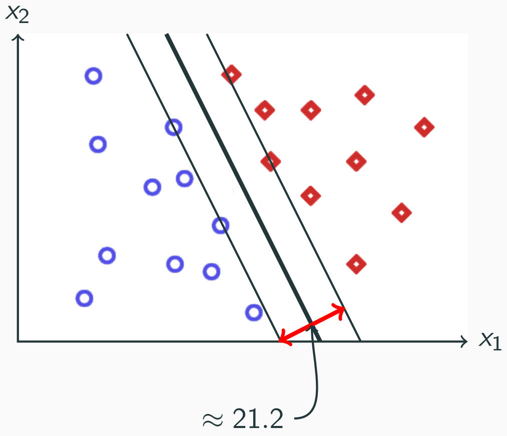
Principle of Minimal Risk



SVM – Intuitively

We train the SVM on a random sample from some unknown distribution. The best we can do is to put a hyperplane in between the two cluster such that the margin becomes maximal.

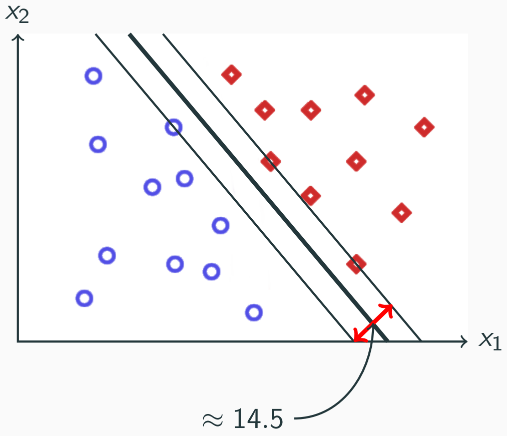
Principle of Minimal Risk



SVM – Intuitively

We train the SVM on a random sample from some unknown distribution. The best we can do is to put a hyperplane in between the two cluster such that the margin becomes maximal.

Principle of Minimal Risk

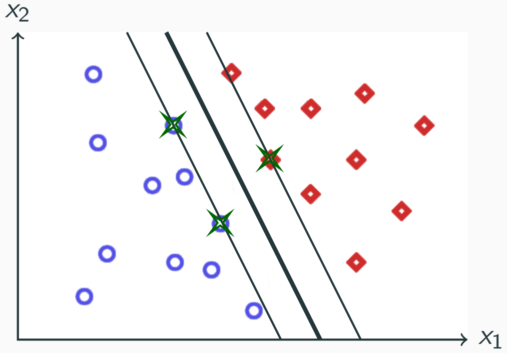


SVM – Intuitively

- The data points defining the hyperplane are called

  Support Vectors

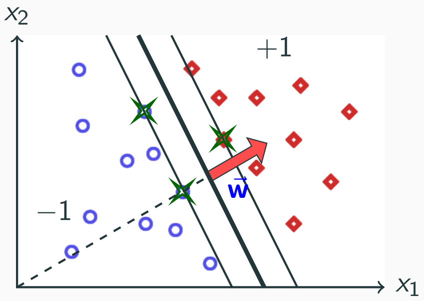
  Like a mechanical model
- In $\mathbb{R}^n$ we need at least $n+1$ to define the margin
- How to formulate this mathematically?



SVM – Math

- Separating hyperplane
  $\vec{\mathbf{w}}\cdot\vec{\mathbf{x}} + b = 0$
  - Training vectors are either "above" or "below"
- Rescaling of $\vec{\mathbf{w}}$ and b such that on the margin for the support vectors:
  $\vec{\mathbf{w}}\cdot\vec{\mathbf{x}}_k + b = \pm 1$
  (since scale of $\vec{\mathbf{w}}$ and b is arbitrary)
- Signal side labels are $+1$
  background side labels are -1
- Multiply with class label y_i for all vectors:

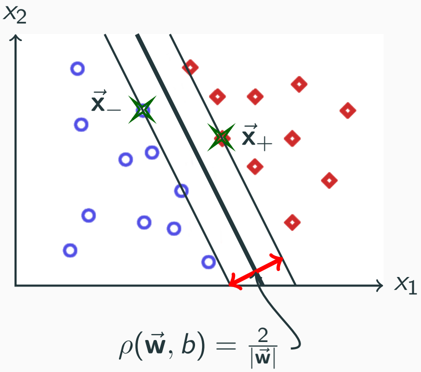
$$\boxed{y_i(\vec{\mathbf{w}}\cdot\vec{\mathbf{x}}_i + b) \geqslant 1}$$



$\Longleftarrow$ These conditions are fulfilled iff all trainings vectors are properly classified

SVM – Math

- Separating hyperplane:
  $\vec{\mathbf{w}} \cdot \vec{\mathbf{x}} + b = 0$
- Constraints:
  $y_i(\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i + b) - 1 \geqslant 0$
- For the width of the margin take 2 arbitrary support vectors $\vec{\mathbf{x}}_+$, $\vec{\mathbf{x}}_-$ and construct
  $\rho(\vec{\mathbf{w}}, b) = \frac{\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_+}{|\vec{\mathbf{w}}|} - \frac{\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_-}{|\vec{\mathbf{w}}|} = \frac{2}{|\vec{\mathbf{w}}|}$



Maximizing the margin $\rho = 2/|\vec{\mathbf{w}}|$ is equivalent to minimizing $|\vec{\mathbf{w}}|^2$

SVM – Math linearly separable

- Finding the optimal separating hyperplane is identical to solving a quadratic (convex) optimization problem
- The correct classification is enforced by the constraints

primal problem

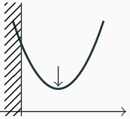
$$\min_{\vec{\mathbf{w}}\in\mathbb{V},\ b\in\mathbb{R}} \quad \frac{1}{2}\left|\vec{\mathbf{w}}\right|^2$$

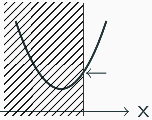
$$\text{subject to} \quad y_i(\vec{\mathbf{w}}\cdot\vec{\mathbf{x}}_i + b) \geqslant 1 \ \text{ for all } i = 1\dots N$$

  - Can be solved with Lagrangian Multiplier $\alpha_i \geqslant 0$
  - This is pure classical mechanics with constraints
  - Unequality constraints needs special care!
    $\Longrightarrow$ KKT (Karush-Kuhn-Tucker) conditions

Karush-Kuhn-Tucker Conditions

- KKT conditions generalizes the method of Lagrange multipliers
  - Inequality constraints
- It combines the two cases
  - Minimum within feasible area, constraint inactive $\lambda^* = 0$
  - Minimum at the border of the constraint area, constraint active as equality $g(x) = 0$ and $\lambda^* > 0$





Given the optimization problem

$$\begin{aligned} \min_{\vec{\mathbf{x}} \in \mathbb{R}^n} \quad & f(\vec{\mathbf{x}}) \\ \text{subject to} \quad & g(\vec{\mathbf{x}}) \leq 0 \end{aligned}$$

Define Lagrangian as

$$\mathscr{L} = f(\vec{\mathbf{x}}) + \lambda g(\vec{\mathbf{x}})$$

Then $\vec{\mathbf{x}}^*$ is a local minimum

$\iff$ there exist a uniq λ^* s.t. the KKT

$$\begin{aligned} &1.) \quad \nabla_{\vec{\mathbf{x}}} \mathscr{L}(\vec{\mathbf{x}}^*, \lambda^*) = 0 \\ &2.) \quad \lambda^* \geq 0 \\ &3.) \quad \lambda^* g(\vec{\mathbf{x}}^*) = 0 \\ &4.) \quad g(\vec{\mathbf{x}}^*) \leq 0 \\ &5.) \quad H = \left[\frac{\partial^2}{\partial x_i \partial x_j} \mathscr{L}(\vec{\mathbf{x}}^*, \lambda^*) \right] \succ 0 \end{aligned}$$

Hessian wrt. $\vec{\mathbf{x}}$ positive definite

SVM – Math

linearly separable

Lagrangian multiplier α_i to include constraints

primal Lagrangian

$$\mathscr{L} = \frac{1}{2}|\vec{\mathbf{w}}|^2 - \sum_{i=1}^{N} \alpha_i [y_i(\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i + b) - 1]$$

The solution is a saddle point $(\vec{\mathbf{w}}^*, b^*, \alpha_i^*)$ and minimal with respect to $\vec{\mathbf{w}}$ and b

Stationarity wrt. the primal variables

$$0 = \frac{\partial \mathscr{L}}{\partial \vec{\mathbf{w}}} = \vec{\mathbf{w}}^* - \sum_{i=1}^{N} \alpha_i y_i \vec{\mathbf{x}}_i; \qquad 0 = \frac{\partial \mathscr{L}}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i$$

Substituting these conditions gives

dual Lagrangian

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \, \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j + \sum_{i=1}^{N} \alpha_i$$

We are left with an quadratic optimization problem in $\vec{\alpha}$

SVM – Math

Lagrangian multiplier α_i to include constraints

primal Lagrangian

$$\mathscr{L} = \frac{1}{2}|\vec{\mathbf{w}}|^2 - \sum_{i=1}^{N} \alpha_i [y_i(\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i + b) - 1]$$

The solution is a saddle point $(\vec{\mathbf{w}}^*, b^*, \alpha_i^*)$ and minimal with respect to $\vec{\mathbf{w}}$ and b

Stationarity wrt. the primal variables

$$\vec{\mathbf{w}}^* = \sum_{i=1}^{N} \alpha_i y_i \vec{\mathbf{x}}_i; \qquad 0 = \sum_{i=1}^{N} \alpha_i y_i$$

Substituting these conditions gives

dual Lagrangian

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \, \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j + \sum_{i=1}^{N} \alpha_i$$

We are left with an quadratic optimization problem in $\vec{\alpha}$

SVM – Math

dual Lagrangian

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\,\vec{\mathbf{x}}_i\cdot\vec{\mathbf{x}}_j + \sum_{i=1}^{N}\alpha_i$$

- Most SVM implementation solve the dual problem
  - Although the dual is of dim N (number of traning vectors) while the primal is of dim n (number of features)
- Remaining KKT conditions

$$\alpha_i(y_i(\vec{\mathbf{w}}^*\cdot\vec{\mathbf{x}}_i + b^*) - 1) = 0; \qquad i = 1\ldots N, \qquad \alpha_i \geqslant 0$$

  - All non-support vectors are forced to have vanishing Lagrange multipliers $\alpha_i = 0$

$\Longrightarrow$ Only the support vectors contribute to the sums

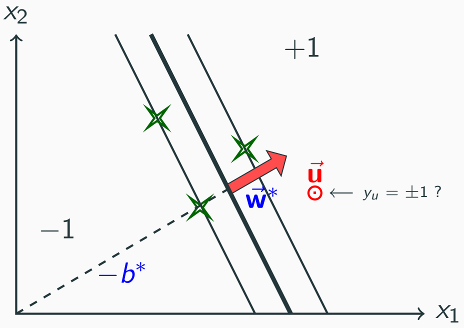
- Sparse solution i.e. only a small subset of training vectors contribute

SVM – Math

The expression to predict the class label $\hat{y}_u$ of a new vector $\vec{\mathbf{u}}$ follows from the hyperplane equation at the optimum

decision function

$$\hat{y} = \text{sign}\left(\sum_{k=1}^{N_{SV}} y_k \alpha_k \vec{\mathbf{x}}_k \cdot \vec{\mathbf{u}} + b^*\right)$$



N_{SV} number of support vector

To train an SVM means we have to find the parameters

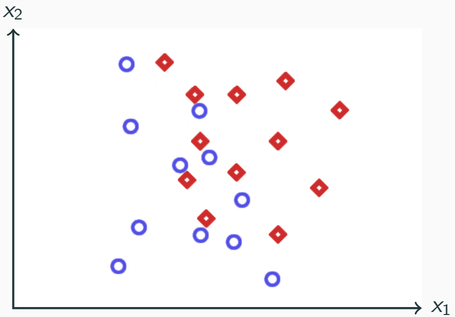
$\alpha_k, \quad k = 1 \dots N_{SV}$

SVM codes typical use the iterative Sequential Minimal Optimization (SMO)

SVM – Math overlapping distribution

- The SVM described so far works for linearly separable data
- In most real world problems we have overlapping distribution for signal and background
- By allowing misclassification, we get from the hard margin to the

Soft margin approach



This can be achieved by introducing slack variables ($\xi_i \geqslant 0,\ i = 1 \ldots N$) which measure for each training vector the distance by which it enters the separating margin

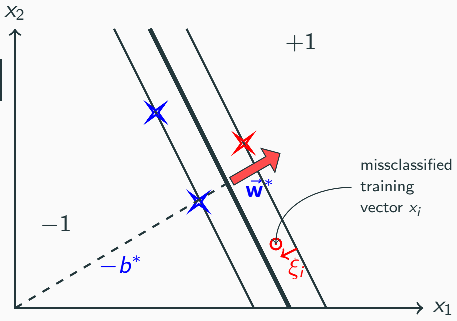
SVM – Math overlapping distribution

Slack variables

- Constraints are weakened

$$\boxed{y_i(\vec{\mathbf{w}}\cdot\vec{\mathbf{x}}_i + b) \geqslant 1 - \xi_i}$$

- Positivity constraint on slacks $\xi_i \geqslant 0 \Rightarrow$ new Lag. mult. β_i
- Sum of the slacks $\sum_i^N \xi_i$ as a penalty term
- Penalty is to be minimized $\partial\mathscr{L}/\partial\xi_i = 0$



This can be achieved by introducing slack variables ($\xi_i \geqslant 0,\ i = 1\ldots N$) which measure for each training vector the distance by which it enters the separating margin

SVM – Math overlapping distribution

We get a new constraint optimization problem with an additional variable ξ_i

- Penalty term
- Positiv slacks
- Soft margin

$$\mathscr{L} = \frac{1}{2}|\vec{\mathbf{w}}|^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i(\vec{\mathbf{w}} \cdot \vec{\mathbf{x}} + b) - 1 + \xi_i] - \sum_i \beta_i \xi_i$$

primal Lagrangian with soft margin

If we do the math ⇒ we get exactly the same Lagrangian up to: $0 \leqslant \alpha_i \leqslant C$. Can be solved by the same code!

dual Lagrangian with soft margin

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j + \sum_{i=1}^{N} \alpha_i$$

We must set the strength C of the penalty term before we do the training!

SVM – Math overlapping distribution

We get a new constraint optimization problem with an additional variable ξ_i

- Penalty term
- Positiv slacks
- Soft margin

$$\mathscr{L} = \frac{1}{2}|\vec{\mathbf{w}}|^2 + C \underbrace{\sum_i \xi_i}_{\text{Penalty term}} - \sum_i \alpha_i[y_i(\vec{\mathbf{w}}\cdot\vec{\mathbf{x}} + b) - 1 + \underbrace{\xi_i}_{\text{Soft margin}} \,] - \underbrace{\sum_i \beta_i\xi_i}_{\text{Positiv slacks}}$$

primal Lagrangian with soft margin

If we do the math ⇒ we get exactly the same Lagrangian up to: $0 \leqslant \alpha_i \leqslant C$. Can be solved by the same code!

dual Lagrangian with soft margin

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\,\vec{\mathbf{x}}_i\cdot\vec{\mathbf{x}}_j + \sum_{i=1}^{N}\alpha_i$$

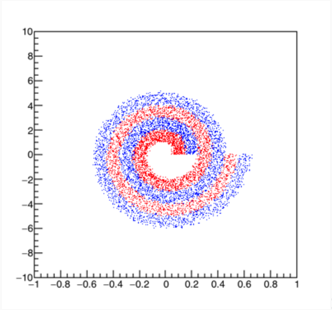
We must set the strength C of the penalty term before we do the training!

SVM – Math

So far we can classify linearly separated, even overlapping data sets. This is nice but not really impressive ...

- The spiral consists of two clearly separated areas
  - Certainly not linearly separable
  - But there is a simple transformation: $\Phi : (x, y) \mapsto (r, \phi)$
    $$r = \sqrt{x^2 + y^2}$$

If we apply this

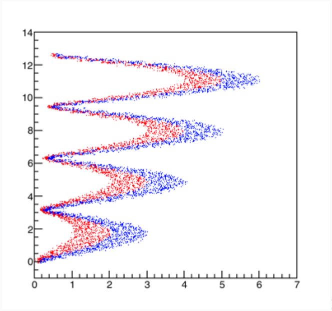


;

SVM – Math

So far we can classify linearly separated, even overlapping data sets. This is nice but not really impressive ...

- The spiral consists of two clearly separated areas
  - Certainly not linearly separable
  - But there is a simple transformation:
    $\Phi : (x, y) \mapsto (r, \phi)$
    $$r = \sqrt{x^2 + y^2}$$



;

Not really what we need ...

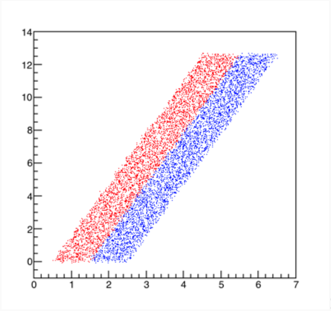
SVM – Math non-linear border

So far we can classify linearly separated, even overlapping data sets. This is nice but not really impressive …

- The spiral consists of two clearly separated areas
    - Certainly not linearly separable
    - But there is a simple transformation: $\Phi : (x, y) \mapsto (r, \phi)$

$$r = \sqrt{100 * x^2 + y^2}$$

We missed the scale difference. Now it is linearly separable!



;

Can we find such a transformation automatically?

SVM – Math

Kernelizing

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j + \sum_{i=1}^{N}\alpha_i \ , \qquad 0 \leqslant \alpha_i \leqslant C$$

- The dual Lagrangian and the decision function only depend on scalar products of the data vectors!
- If we map $\vec{\mathbf{x}}$ into some other space we must define a scalar product $\vec{\mathbf{x}} \cdot \vec{\mathbf{y}} \mapsto \langle \Phi(\vec{\mathbf{x}}), \Phi(\vec{\mathbf{y}}) \rangle$

$$\hat{y} = \mathrm{sign}\Big(\sum_{k=1}^{N_{SV}} y_k \alpha_k \vec{\mathbf{x}}_k \cdot \vec{\mathbf{u}} + b^*\Big)$$

- Instead of defining the transformation and the scalar product explicitly we can as well define a

Kernelizing

$$\mathscr{L}(\vec{\alpha}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\, K(\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j) + \sum_{i=1}^{N}\alpha_i\ , \qquad 0 \leqslant \alpha_i \leqslant C$$

- The dual Lagrangian and the decision function only depend on scalar products of the data vectors!
- If we map $\vec{\mathbf{x}}$ into some other space we must define a scalar product $\vec{\mathbf{x}}\cdot\vec{\mathbf{y}} \mapsto \langle\Phi(\vec{\mathbf{x}}),\Phi(\vec{\mathbf{y}})\rangle$

$$\hat{y} = \text{sign}\Big(\sum_{k=1}^{N_{SV}} y_k\alpha_k K(\vec{\mathbf{x}}_k,\vec{\mathbf{u}}) + b^*\Big)$$

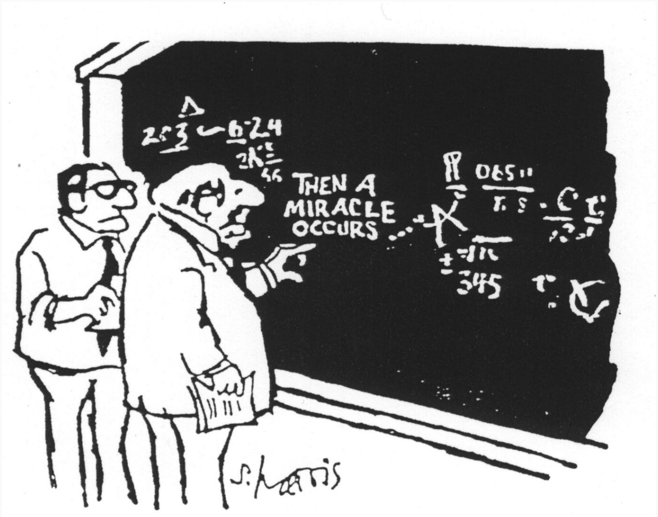
- Instead of defining the transformation and the scalar product explicitly we can as well define a

Kernel function
$K(\vec{\mathbf{x}},\vec{\mathbf{y}}) \in \mathbb{R}$

Now we can solve ALL kinds of non-linear Machine Learning Problems

!

SVM – Math



I think you should be a bit more precise, here in Step 3!

SVM – Math non-linear separation

Kernels

- There is no need to restrict the transformation to finite dimensional Vector spaces.
- $\mathscr{H}$ may be a even infinite dimensional Hilbert space $\Phi : \mathbb{R}^n \mapsto \mathscr{H}$
- This is the case for the most successful Gaussian Kernel aka RBF (Radial Basis Function)

$$\boxed{K(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = e^{-\gamma|\vec{\mathbf{x}}-\vec{\mathbf{y}}|^2}}$$

- Mapping data to a space with an enormous number of dimensions seems to be a bad idea for good generalization properties.

The maximum margin/minimal risk approach helps; see VC dimension

- Not all functions $K(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ are valid Kernels. They must fulfil Mercer's conditions, in particular symmetric and positive semidefinite $\implies$ $\mathscr{H}$ exists with a suitable scalar product
- If this sounds familiar from QM, you are right. There is a connection to the Spectral Theorem
- The "**Kernel trick**" is a general technique to transform a linear algorithm into a non-linear one

SVM — in a nut shell

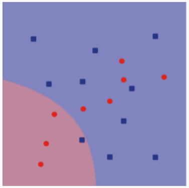
- The SVM selects a smaller subset of training vectors: Support Vectors
- A penalty term of strength C allows for overlapping distribution

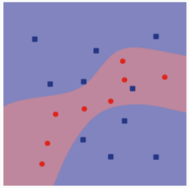
  BTW The penalty term can become event dependent

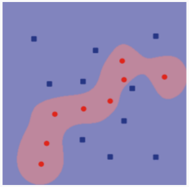
  $\rightarrow$ Weighted SVM
- The Kernel acts as a kind of "basis" functions to model a non-linear separating hyper-surface. For the RBF case

$$\hat{y}(\vec{\mathbf{u}}) = \mathrm{sign}\left(\sum_{k=1}^{N_{SV}} y_k \alpha_k e^{-\gamma|\vec{\mathbf{x}}_k - \vec{\mathbf{u}}|^2} + b^*\right);$$

- $\gamma \sim 1/\sigma$ controls the width of the Gaussian $\Longrightarrow$
- We must fix the two parameters C and γ before training


$\gamma = 1$


$\gamma = 10$


$\gamma = 100$

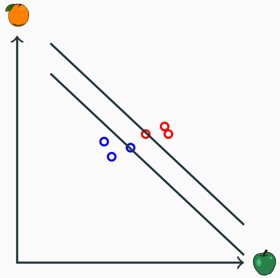
We are doing convex quadratic optimisation. Do you understand why we still can do convex quadratic optimization after the Kernel trick? Isn't it mapping wildly simple constraints into complex, non-linear borders?

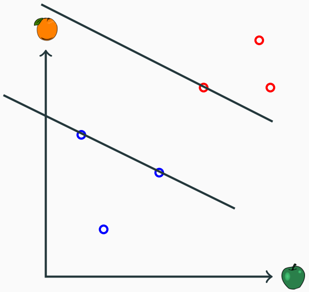
SVM – Usage (bits ad bolts) 🛠

SVM – Usage

The apples and oranges problem

- There is no natural common scale between features. Distance $|\vec{\mathbf{x}}| = \sum_i (x^{(i)})^2$ is arbitrary
- This influences the max margin decision and the performances of the SVM
- In addition, the RBF kernel knows only one scale γ and treats all dimensions equally
- A common approach is to scale each dimension by the range of the trainings data e.g. $x^{(i)} \to \frac{x^{(i)}}{x^{(i)}_{max} - x^{(i)}_{min}} \qquad i = 1, \dots, n$





SVM – Usage   Machine learning performance measures

- We have a binary classification problem. On a test sample we have true and a predicted labels $\Longrightarrow$ 4 cases of classified events
- Different ML quality measures are in use

$$\text{Accuracy} = \frac{N_{++} + N_{--}}{\text{all}}$$

$$\text{Precision} = \frac{N_{++}}{N_{++} + N_{-+}}$$

etc.

labels	predicted	
true	N_{++}	N_{+-}
	N_{-+}	N_{--}

events numbers

all $= N_{++} + N_{+-} + N_{-+} + N_{--}$

AUC: the area under the receiver operator curve (ROC)
The ROC curve shows the background rejection (false positive) against the signal efficiency (true positive) vs threshold of the decision function.
(You have seen this $\Longrightarrow$ TMVA)

SVM – Usage parameter tuning

Training a SVM is easy. Quadratic optimisation
$\Longrightarrow$ one minimum, uniqe solution!
But, how to find the parameters (C, γ)? $\Longrightarrow$ Brute force

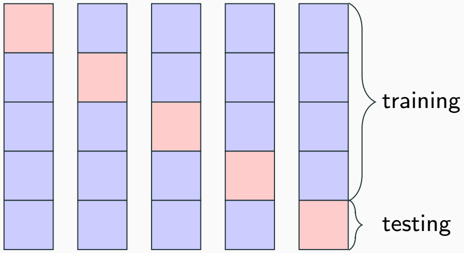
- We scan a reasonable part of the parameter space
  (grid search on a logarithmically defined grid or iteratively refined grid search)
- For each (C, γ)-pair we train a SVM and evaluate the classification performance e.g. accuracy
  need to split input sample* into training and test samples to avoid bias
- At the end we take the best (C, γ)-pair

*Alternatively cross validation instead of splitting into training and test sample

SVM – Usage

E.g. 5-fold cross validation

1. Split sample in 5 sub-samples
2. Train on 4/5th of the sample
3. Evaluate on remaining 5th
4. Repeat with changing roles
5. Evaluate final accuracy on 5 x 1/5



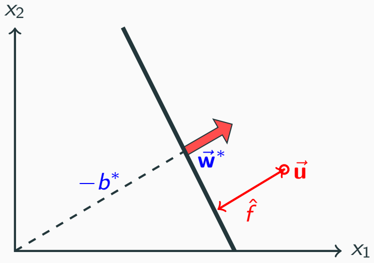
SVM – Usage Probabillity Estimate

- The SVM is a binary classifier with yes or no output.
- Often a probability P that quantifies the belief in the class label is useful
- Can be estimated by fitting a sigmoid model to the distancees to the hyperplane for training data $(y_i, \vec{\mathbf{x}}_i)$
  (max likelihood fit with 5fold cross validation to avoid overfitting)
- A cut on the probability can be used to improve the S/N ratio
- NB: $P > p_{\text{cut}} > 0.5 \Longrightarrow N_{++} + N_{--} < N$

$$P(+1|\hat{f}) = \frac{1}{1 + \exp(A\hat{f} + B)}$$

$$\hat{f} = \sum_{k=1}^{N_{SV}} y_k \alpha_k e^{-\gamma|\vec{\mathbf{x}}_k - \vec{\mathbf{u}}|^2} + b^* x$$

distance to the sepa-
rating hyperplane



SVM – Usage — Significance based tuning

- From a machine learning perspective, a natural performance measure describes how well a classifier separates the two distinct classes.
  E.g. accuracy $\frac{\text{correct}}{\text{all}}$
- Accuracy improves if we can identify a lot of background but we do not care much if the bgrd is predicted correct, especially if signal$<<$background
- From a physics perspective one wants to optimise the discovery significance
  $\Longrightarrow$ Tuning for best significance instead of accuracy
- No mathematical prove that this converges to the best solution but its works in practice `http://arxiv.org/abs/1601.02809`
- We tune C, γ and p_{cut} to optimise discovery significance

Machine learning classifier are independent of the number of events while significance based tuning is meant for a certain expected number of events (Luminosity)

SVM – Usage

Asimov significance

Assume a new physics search – a counting experiment

- You plan an analysis that would yield n-events
- You expect b background events
- Your data is Poisson distributed and you want to check for a signal s
- Your background only hypothesis $s = 0$ looks like this

$$\text{Poisson}(n|b) = \frac{b^n}{n!}e^b \mapsto G(n|\mu = b, \sigma = \sqrt{b})$$

- For $\underline{\text{large } n}$ the Poisson looks like a Gaussian and the significance to reject the background only hypothesis is

$$\text{discovery significance} = \frac{n_{obs} - b}{\sqrt{b}}$$

i.e. the number of "sigmas" you a away from the mean

- Since you have not yet done the experiment and you want to check for a signal s, you replace this by the expected $\langle n \rangle = s + b$

$$Z_A := \langle \text{disc. significance} = \rangle = \frac{\langle n_{obs} \rangle - b}{\sqrt{b}} = \frac{s}{\sqrt{b}}$$

SVM – Usage

Asimov significance

Q: But what if n is not large?

A: You read this paper (arXiv:1007.1727):
"Asymptotic formulae for likelihood-based tests of new physics"
Or a talk by Glen Cowan here

- It starts with the Poisson distribution, tests for discovery by using a profile likelihood ratio, approximate with Wilk's theorem and applies the "Asimov" data set which is just the $\langle n \rangle = s + b$

$$\text{Asimov significance} \quad Z_A = \left[2\left((s+b)\ln\left[1+\frac{s}{b}\right] - s\right)\right]^{1/2}$$

- This approach allows to include a uncertainty σ_b^2 on b

$$Z_A = \left[2\left((s+b)\ln\left[\frac{(s+b)(b+\sigma_b^2)}{b^2+(s+b)\sigma_b^2}\right] - \frac{b^2}{\sigma_b^2}\ln\left[1+\frac{\sigma_b^2 s}{b(b+\sigma_b^2)}\right]\right)\right]^{1/2}$$

SVM – Conclusions

- SVMs are an efficient ML algorithm, widely used
- Good performance, (BTW robust against a large number, partly correlated features)
- For an RBF kernel SVM only **2** parameters must be tuned
- Parameter tuning can be used to optimise a physical motivated target (discovery significance)

SVM – Tutorial