*DataSet (Набор данных)*



*DataSet* – это коллекция событий. Каждое событие состоит из входа, выхода и целевого вектора. Когда события обрабатываются, сеть считывает входной вектор и хранит ответы в выходных. Целевые вектора используются в контролируемом обучающем процессе, чтобы сохранять желаемые ответы сети для каждого события в наборе данных для обучения. Также целевые вектора могут быть использованы для расчёта ошибки. Каждое событие может сопровождается вектором не используемых значений, который используется для фильтрации расчётов, чтобы перейти к следующим блокам или для разных графиков.

Все данные могут храниться в текстовом файле и подключатся к *DataSet* блоку. Чтобы добавить блок в проект в меню нужно выбрать **Edit - Add Component - Data Set**.

*Синтаксис строки форматирования*

Строка форматирования используется для описания формата файла с данными. Каждая строка включает в себя разделённые точкой с запятой записи (элементы вектора) состоящие из записи ***type*** (тип) и ***index*** (индекс); доступные типы **i** (вход), **o** (выход), **t** (целевой), **b** неопределённость, **n** (ничего или не используемые) **e** (ошибка);

*Примеры:*

**i1; i2; i3; t1; t2** – применительно к вход/выход формату файла: файл включает события входных векторов (длинна = 3) и целевыми векторами (длинна = 2); подходящие вектора выхода (длинна = 2) будут созданы автоматически со значением “0”.

**i1; o1; t1; i2; o2; t2** - применительно к вход/выход формату файла: все данные хранятся в файле, но векторные компоненты перемешаны.

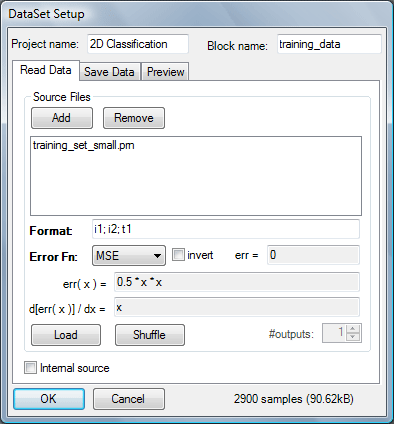
**n1; i1; i2; n2; t1** - применительно к вход/выход формату файла: два числа которые не используются, они добавляются в вектор не используемых значений; кроме того, двойной входной вектор и единственное целевое значение доступны для каждого события.

**ri:1-10; o1; t1** – последовательность записей может быть заменена одной записью **r*t*:start-stop**, где ***t*** тип записей, **start** и **stop** индексы первой и последней записи; этот формат строки эквивалентен **i2; i3; i4; i5; i6; i7; i8; i9; i10; o1; t1**;  
**ri:1-1000; rt:1-16; ri:1001-1024 –** ещё один пример использования области записью.

*Настройка DataSet (Кнопка Setup)*

Здесь можно проводить операции над входными и выходными данными, а также просматривать данные событий. Можно изменить имя блока и проекта.

*Считывание из файла:*



**Source Files:** список файлов для записи в коллекцию событий *DataSet* блока. Нажатие на кнопки позволяет изменить содержимое.

**Format:** представляет выравнивание данных в исходных файлах

**Error Fn:** функция ошибки, которая должна быть минимизирована в процессе обучения.

**invert:** переворачивает асимметричные функции ошибки  einverted(*x*) = e(-*x*); для симметричных функций нет разницы. Показывает последнее рассчитанное значение ошибки.

**err:** показывает последнее рассчитанное значение ошибки. Выражение функции ошибки

**err( x ):** выражение функции ошибки.

**d[err( x )] / dx:** производное выражение функции ошибки.

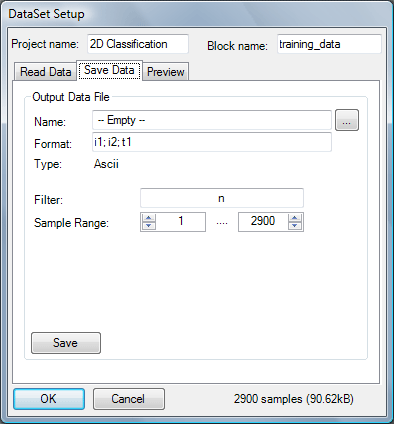
**#outputs:** если нет целевых и выходных векторов в формате строки, эта опция позволяет выделить память для определённого числа выходов.

**Load:** загружает/перезагружает данные из файла.

**Shuffle:** перемешивает события в *DataSet*.

**Internal source:** если выбрано, выключает *SourceFiles* панель и включает получение данных из *Transform* или *Network* блоков.

*Запись данных в файл:*



**Name:** имя выходного файла.

**Format:** представляет выравнивание данных в файле.

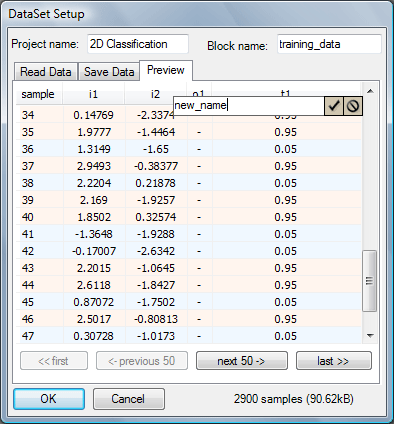
**Filter:** события запишутся в файл только если они соответствуют условию фильтра.

**Sample range:** область индексов событий для записи в файле.

**Save:** сохранить в файл.

*Отображение содержимого DataSet:*

Вкладка preview позволяет увидеть содержимое данных. Столбик output показывает “-“ если событие ещё не было обработано сетью.



**<- previous 50 / next 50 ->:** предыдущие 50 / следующие 50 событий.

**<< first / last >>:** первое/последнее событие.

*Network (сеть)*



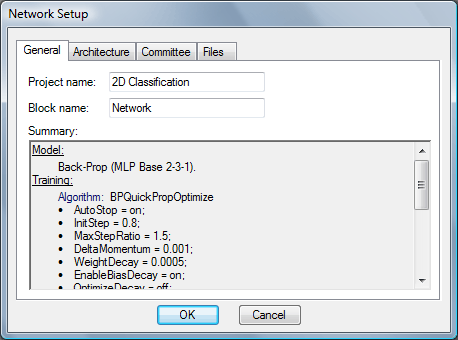
Блок *Network* поводит все нейронные вычисления и оптимизации. Он считывает события из *DataSet* блока подключённого через **I/O,** вычисляет ответы сети (выходной вектор) и кладёт их обратно во входные события.

Чтобы добавить блок *Network* к проекту в меню выбираем **Edit - Add Component - Network**.

*Структурная настройка сети (Кнопка Setup).*

Опции расположены во вкладках: общая информация, ручная настройка и работа с файлами.

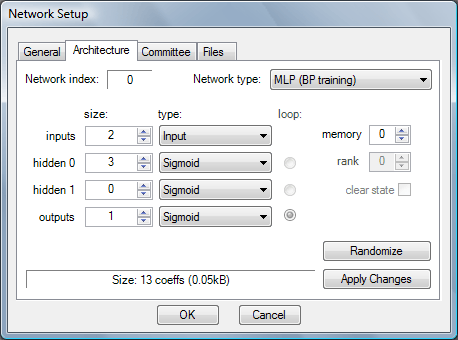
*Общая информация:*



Здесь можно изменить имена проекта и блока. *Summary* показывает текущие настройки сети.

*Настройка структуры:*

Структура сети может быть установлена пользователем или может быть автоматически настроена во время обучения, но даже в этом случае необходима начальная настройка некоторых параметров. Число входов и выходов сети обусловлено применением сети. Число скрытых нейронов и активация функции (f*act*) настраивается пользователем. Все слои нейронов пользуются одинаковой функцией активации, но f*act* может различаться между слоями.



**Network type** – тип нейронной сети:

* **BP Training –** это многослойный персептрон c контролируемым шаблоном обучением обратного распространения. Используется в прямом (**MLP**) и рекурсивном (**RMLP)** персептроне.
* **Cascade-Correlation** **–** архитектура и шаблон обучения предложеные S. Fahlman.

**Network index**: выбранный индекс в комитете сетей.

**inputs:** число входов сети (длинна вектора входа). Тип *Input* для *MLP* и *Recurrent* для *RMLP* сети.

**hidden 0/1:** число скрытых нейронов и тип функции активации скрытого слоя.

**outputs:** число выходов (длинна вектора выхода) и тип функции активации слоя выхода

**Randomize:** устанавливает случайные коэффициенты взаимосвязи.

**Apply Changes:** если параметры были изменены создаёт новую сеть, со случайными коэффициентами.

**memory:** число событий, ранее обработанных и показываемых сети одновременно с обрабатываемым событием.

Следующие опции доступны только в *RMLP* сети:

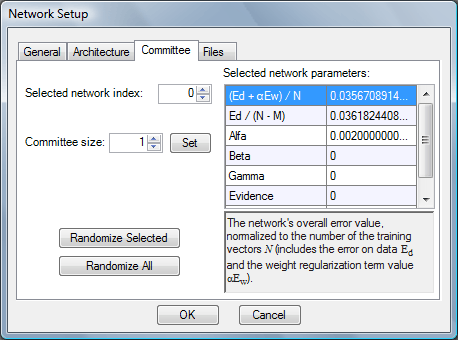
**rank** - число предыдущих выходов сети из слоя указанного в *loop*

**loop** - помечает слой как источник обратной связи.

**clear state** – если помечено, обратная связь сбрасывается после обработки каждого *DataSet*.

*Комитет сетей*

Комитет это несколько связанных сетей, которые подготовлены для выполнения одной цели, но обучены порознь.



**Selected network index:** индекс сети в комитете.

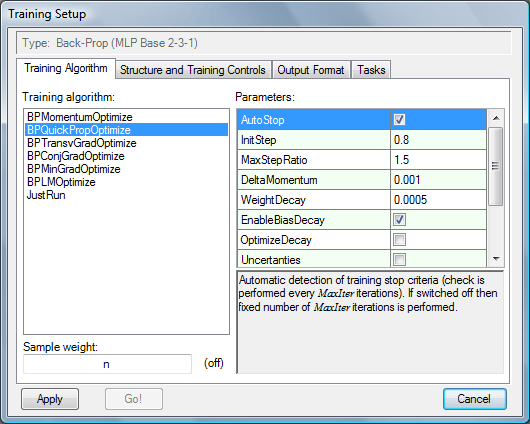
**Committee size:** число сетей в комитете. Кнопка Set утверждает изменения.

**Randomize Selected:** делает случайными коэффицеты связности выбранной сети.

**Randomize All: включает все сети в комитете используя модель выбранной сети. Все сети** рандомизируются.

*Параметры обучения и запуска (Кнопка Go)*

*Алгоритмы обучения и параметры:*



**Training algorithm:** список доступных алгоритмов.

**JustRun:** запуск сети без использования DataSet и обучения, просто для расчёта выходов сети.

***Parameters:*** ***randomized***

**AutoStop:** если выключен, обучение останавливается при первом приемлемой итерации после МaxIter. Если включен, то проверяется определённый набор условий после каждой итерации.

**MaxIter:** число итераций обучения/ интервал проверки условий остановки.

**MinError:** удовлетворяющий уровень ошибки. Если достигнут, обучение останавливается.

**MaxError:** максимальный допустимый уровень ошибки. Если он не достигнут после остановки обучения, то сеть рандомизируется и обучение проходит заново. 0 выключает эту опцию.

**WeightDecay:** фактор регуляризации *α* для *weight decay* в функции ошибки. Чем выше значение, тем вывод сети становится более гладким и устойчив к статистическим колебаниям данных обучения.

**Sample weight:** вес относящийся к каждому событию в DataSet соединённой к сети. Он используется для изменения значений ошибок и градиентов, рассчитанных для каждого события. Выражение ***Sample weight*** может использовать переменные, доступные с событием, например любые входящие, целевые и не используемые элементы вектора могут быть использованы с i1, t1, b1, n1, ... кодами. Если существуют неопределенности событий, тогда ошибка *χ*2 функции может быть реализована, установив ***Sample weight*** на 1/(b1\*b1) и ошибку функции MSE, где b1 предполагается как значение неопределённости события.

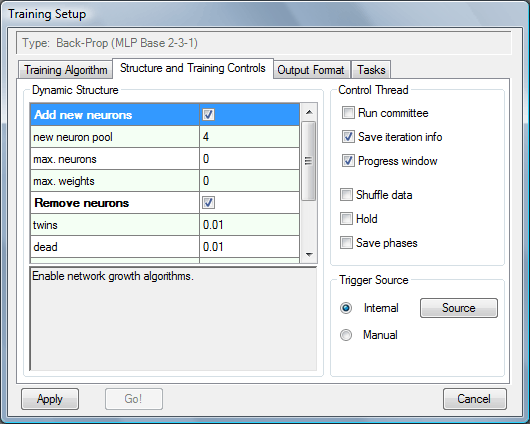
**RMPL** сети:

**TeacherForcing:** коэффициент смешивания ( *f* ) для рекуррентных вычислений

*inpt* = *outt*-*n* ⋅ (1 - *f*) + *tgtt*-*n* ⋅ *f*,   *n* = 1...ранг, где *outt*-*n* это выход сети на итерации *t*-*n.* *tgtt*-*n* Это целевое значение для этого выхода; если *f* = 1 , рекуррентный вход полностью заменён на желаемое целевое значение, если *f* = 0  тогда принуждение учителя отключено.

**ForcingDecay:** коэффициент распада *δ*  для коэффициента принуждения учителя; если ошибка сети уменьшается, *f* пересчитывается на каждой итерации как *ft* = *δ* ⋅ *ft*-1; if *δ* = 1 тогда коэффициента принуждения учителя это константа, *δ* = 0  не допустим.

*Динамическая структура и настройка обучения:*



*Динамическая структура***:**

**Add new neurons:**  включает рост сетевой структуры.

**pool:**  число случайных кандидатов – каждый раз когда размер сети увеличивается пул новых рандомизированных нейронов обучен, а лучший выбирается для расширения сети (но только если кажется что он может быть полезен всей сети.  
**neuron split:** спрятанные нейроны с наивысшей вариацией ошибка выхода разделяются на два немного другие нейроны. Затем пара новых нейронов проходит предварительное обучение; эти нейроны встраиваются в сеть, если они выдают лучший результат, чем случайные кандидаты; эта настройка всегда включена.

**max neurons, max weights:** ограничивает число нейронов и коэффициенты сетевого соединения. Обучение останавливается когда любой из этих лимитов достигнут. Значение 0 выключает ограничение.

**Remove neurons:** включает обрезку неиспользуемых или избыточных нейронов; **twins** - мера разницы между активациями двух нейронов, которые будут заменены одним эквивалентным нейроном; **dead** – мера значимости нейронов которые могут быть безопасно удалены; **const** – мера активации нейронов σ/μ , которая будет использоваться для учета нейрона как константы ; увеличивая эти значения будет удалятся больше нейронов.

**OBS (Optimal Brain Surgeon):** техника удаления единственного соединения из сети; Если OBS включён, алгоритм применяется в конце процесса обучения; если **iterative** тоже включён, алгоритм применяется после каждой попытки вырезания нейронов. Этот алгоритм использует большое количество времени и памяти.

**Control Thread:**

**Run committee:** запускает все сети в комитете.

**Save iteration info:** включает хранение информации о обучении/тестировании сети в каждой итерации. Необходимо для графиков  ***Network Error.***

**Progress window:** позволяет управлять окном с статистикой обучения.

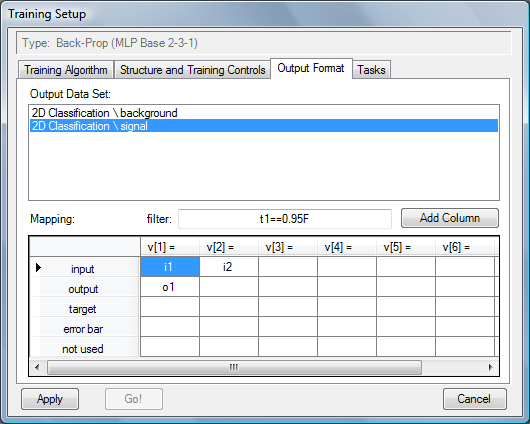
**Hold:** если выбрано, обучение приостановится перед и после каждым изменением структуры.

**Save phases: если выбрано, модель сети будет хранится в файлах перед и после каждым изменением структуры.**

**Shuffle data:** позволяет проводить рандомизированное переупорядочение обучающих векторов во время обучения.

**Trigger Source:** если выбран ***ручной*** способ, то кнопка **Go!** активна и выпускает классификационные вычисления. Если выбран ***внутренний*** способ, то кнопка Source активна. Она позволяет выбрать источник вызова через обычное диалоговое окно добавления соединений **Connection Add**. Обработка начинается, когда один из выбранных источников заканчивает своё задание

*Формат вывода и фильтр событий:*



Когда сеть заканчивает обработку, содержимое **DataSet** входа посылается на выход в соответствии со спецификациями отображения вывода. События назначения в каждом выходе ***DataSet*** могут быть составлены по-другому. Выбрав **DataSet**  в списке **Output Data Set**, добавьте желаемые выражения в таблицу **Mapping** для векторов событий назначения.

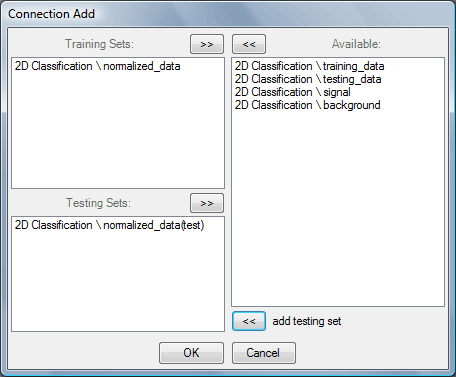
Пример выше выполняет следующее отображение:

* входной вектор события назначения будет состоять из первого и второго элементов входного вектора источника события;
* вектор выделения целевого события будет создан с длиной = 1, и он будет содержать первый элемент сетевого ответного вектора (выходной вектор);
* целевой объект назначения и неиспользованный вектор не выделяются.

События, хранящиеся в ***DataSet*** выходе, могут быть фильтрованы (в приведенном выше примере только события с t1 == 0.95F будут отправлены в «сигнал» ***DataSet***).

### *Input list (I/O button)*

Кнопка **I/O** открывает список **DataSet** обучения и тестирования. Двойной клик на элемент из списка добавляет новую выборку обучения. Нижняя кнопка << добавляет новый элемент к списку **Testing Sets**  выборкам тестирования. Двойной клик на элемент из **Training Sets**  и **Testing Sets** удаляет соединение.



Когда в процессе обучения присутствуют как тренировочные, так и тестовые наборы, для расчета изменений веса межсоединений нейронов используются обучающие наборы, затем вычисляется ошибка на наборах тестов. Если ошибка на наборах тестов постоянно увеличивается, обучение прекращается. Это хороший способ избежать перетренированности.

### *Output list (>> button)*

### *Подключает вывод сетевого блока к целевым данным. Открывает общее диалоговое окно «Добавить соединение».*

*Classification (блок классификации)*



Блок классификации выполняет один из классифицирующих алгоритмов, на входящий вектор из блока DataSet (Input) и посылает результат в DataSet блок (Output). События в DataSet выходе сконструированы в соответствии с форматом фильтров. Схоже с блоком *Network*,результат также хранится в векторах выхода событий во DataSet входных данных.

Каждый алгоритм использует выборки обучения, которые содержатся в DataSet указывающий на установку блока. События в обучающей выборке должны содержать класс указывающего события целевого вектора.

Целевой вектор должен состоять из одного элемента (метки) в случае двухуровневой задачи разделения; значения определяются как сигнал: 1.0, 0.95, 0.9; другие значения считаются за фоновый сигнал. Многоуровневое разделение требует, чтобы длинна целевого вектора была ровна числу признанных классов. Каждое событие должно быть отмечено в целевом векторе значением сигнала помещённое на позицию соответствующего класса. Значения, возвращаемые в качестве результата классификации события X, являются оценщиками условной вероятности.

**output***C*(**x**) ≈ P(**x** ∈ *C* | **x** = **input**) = g*C*(**x**)∙p*C* / ∑[g*i*(**x**)∙p*i*] (1)

где g*i*(**x**) –вероятность функции распределения, p*i* - априорная вероятность.

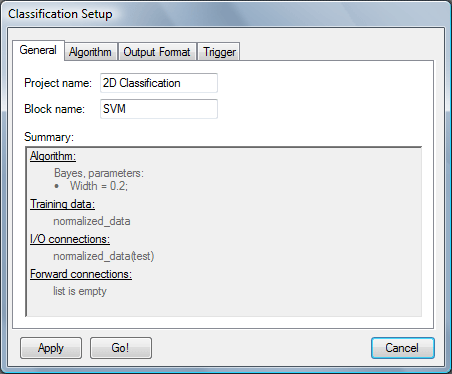
Чтобы добавить блок классификации в проект, выберете **Edit - Add Component – Classification.**

### *Настройка блока классификации (кнопка Setup):*

Кнопка  **Apply** сохраняет все параметры, но не начинает обработка; **Go!** Сохраняет параметры и начинает обработку, она появляется только при ручном способе запуска.

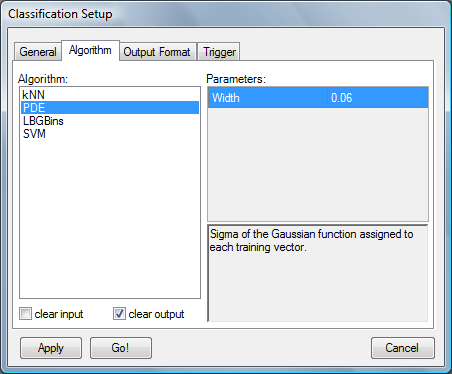
Другие контроли сгруппированы в таблицы: общая информация, формат вывода и настройка фильтров, а также настройка запуска.

*Общая информация:*



Блок и проект имена могут быть изменены тут. **Summary** показывает текущую настройку блока.

#### Настройка алгоритма:



*Классификационные алгоритмы:*

**kNN:** *k* nearest neighbors; алгоритм высчитывает вывод на основе числа сигналов событий среди k обучающих событий с наименьшим расстоянием до классифицированного события. Это простой, относительно быстрый и и известный алгоритм, но плохо работает в многомерных пространствах функций.

**PDE: Распределение плотности вероятности класса оцениваются как сумма функции Гаусса, центрированной по событиям из набора обучения. Тогда условная вероятность высчитывается прямо через формулу (1). Ширина функции Гаусса определяется пользователем. Алгоритм занимает очень много времени, но даёт более надёжный результат, чем kNN.**

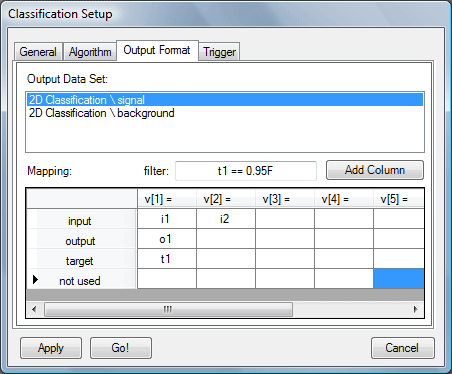
**LBGBins:** Пространство обучающих векторов квантуется LBG алгоритмом для получения N репрезентативных векторов. Эти вектора становятся центрами N секторов, при этом равномерная условная вероятность высчитывается как отношение сигналов к событиям тренировки фона в данном секторе. Классифицированные события назначаются к ближайшему центру, в результате используя его предварительно рассчитанную вероятность.

**SVM:** Поддержка векторных машин на основе библиотеки SVM.NET.

**clear input:** **DataSets** входа будут очищены после проработки.

**clear output:** **DataSets** будут очищены до обработки. Если не выбрано, то новые события будут привязаны к уже существующим событиям.

#### Формат вывода и настройка фильтра:



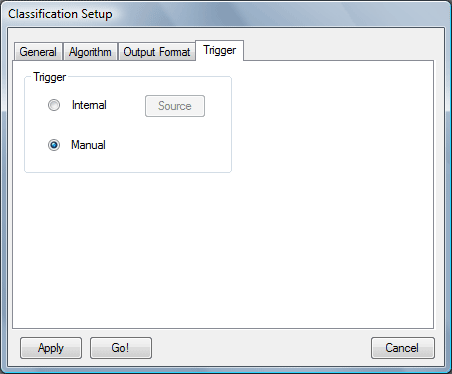
Когда классификация событий из входа **DataSet** выполнена, их содержимое отправляется на **DataSet** выхода, в соответствии со спецификацией отображения формата вывода. События назначения в каждом выходе **DataSet** могут быть составлены по-разному. Можно выбрать **DataSet**  всписке **Output Data Set**  и желаемые выражения для элементов вектора события назначения. Пример выше показывает:

* входной вектор события назначения будет состоять из первого и второго элементов входного вектора источника события;
* входной вектор события назначения будет создан с длинной = 1 и будет содержать первый элемент выходного вектора классификации;
* целевой объект назначения будет содержать копию первого элемента исходного целевого вектора.

События, хранящиеся в **DataSet** выхода, могут быть отфильтрованы (в примере выше события с t1=0.95F будут отправлены в **DataSet сигнала).**

#### 

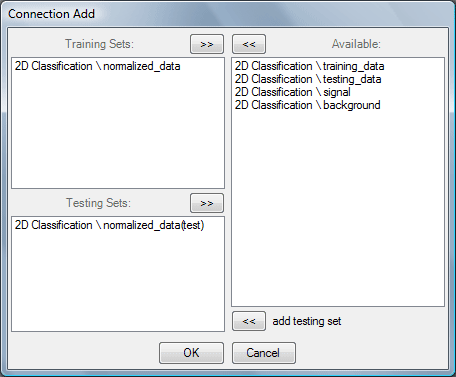
*Настройка запуска:*



Если выбран ***ручной*** способ, то кнопка **Go!** активна и выпускает классификационные вычисления. Если выбран ***внутренний*** способ, то кнопка Source активна. Она позволяет выбрать источник вызова через обычное диалоговое окно добавления соединений **Connection Add**. Обработка начинается, когда один из выбранных источников заканчивает своё задание.

### *Input list (left >> button):*

Кнопка **>>** открывает список **DataSet** обучения и тестирования – расширенное окно **Connection Add.** Двойное нажатие на элемент из списка  (**Available)**  доступных выборок добавляет новую обучающую выборку. Нижняя кнопка << добавляет новые элементы к списку **Testing sets.** Двойное нажатие на элемент из **Training Sets** или **Testing Sets** убирает связь.



*Output list (right >> button):*

Подключает вывод блока ***Classify*** к целевым наборам данных. Открывает общее диалоговое окно ***Connection Add.***