Sistemas Multi-Agentes

Minería de Datos basada en Agentes

Dr. Alejandro Guerra-Hernández

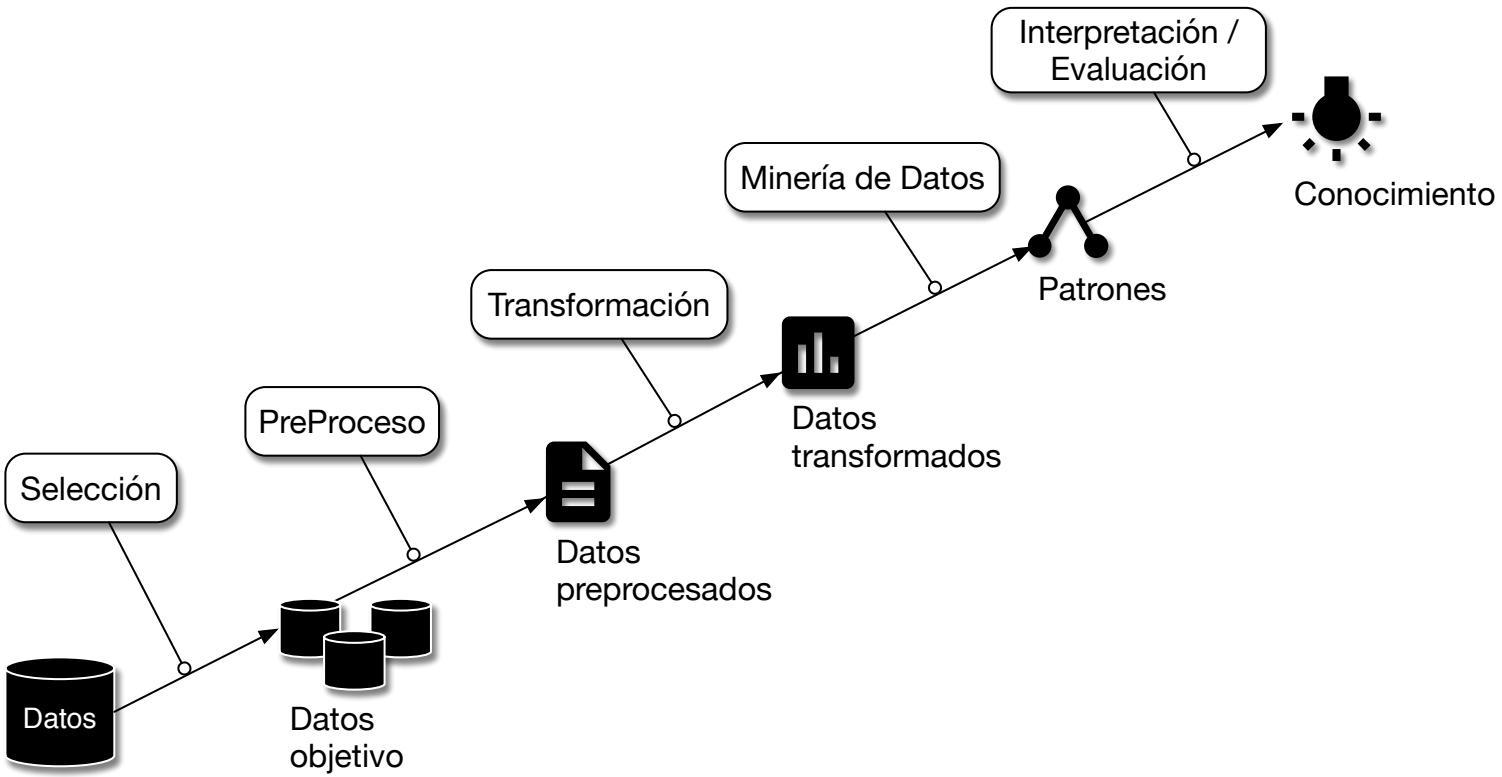
Universidad Veracruzana
Instituto de Investigaciones en Inteligencia Artificial
Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097

aguerra@uv.mx
https://www.uv.mx/personal/aguerra/

Maestría en Inteligencia Artificial 2023

Minería de datos y Descubrimiento de Conocimientos

- Fayyad, Piatetsky-Shapiro y Smyth [6] definen el proceso de extracción de conocimiento a partir de datos, como sigue:



Universidad Veracruzana

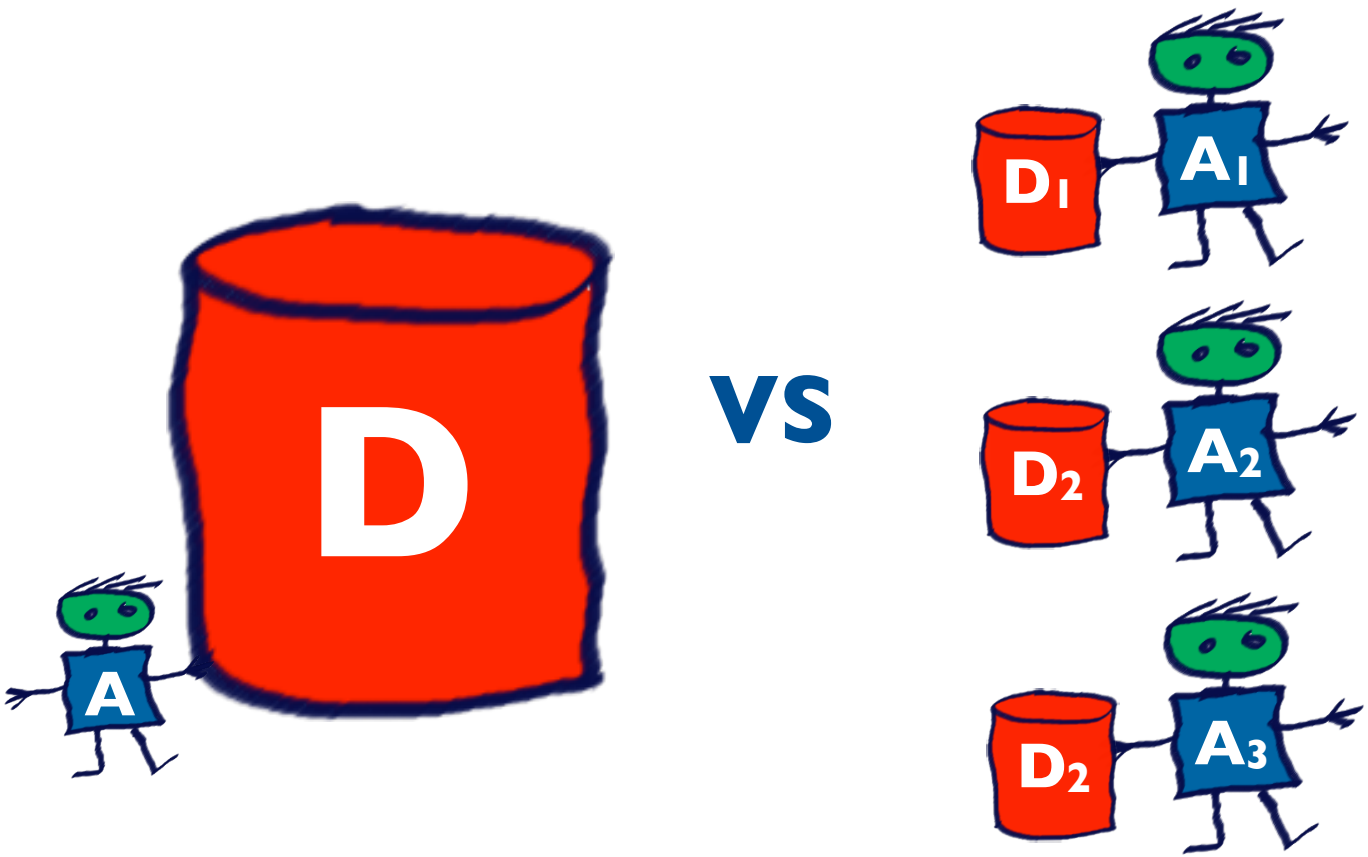
Herramientas: Weka [15]

- Representación uniforme de ejemplos, atributos, modelos, etc.
- Herramientas de evaluación de modelos.
- Soporte: Libro, documentación, comunidad activa, etc.
- Diversos algoritmos, código abierto, para:

Minería de Datos	Transformación	Meta-Aprendizaje
Árboles de decisión	Selección de atributos	Bagging
Reglas	Discretización	Boosting
Modelos lineales	Proyección	Combinación de modelos
Modelos basados en casos	Muestreo	Regresión aditiva
Predicción numérica	Calibración	Stacking
Modelos bayesianos	etc.	etc.
Redes neuronales		
Clustering		
Flujos		

Universidad Veracruzana

¿Y si los datos son demasiados?



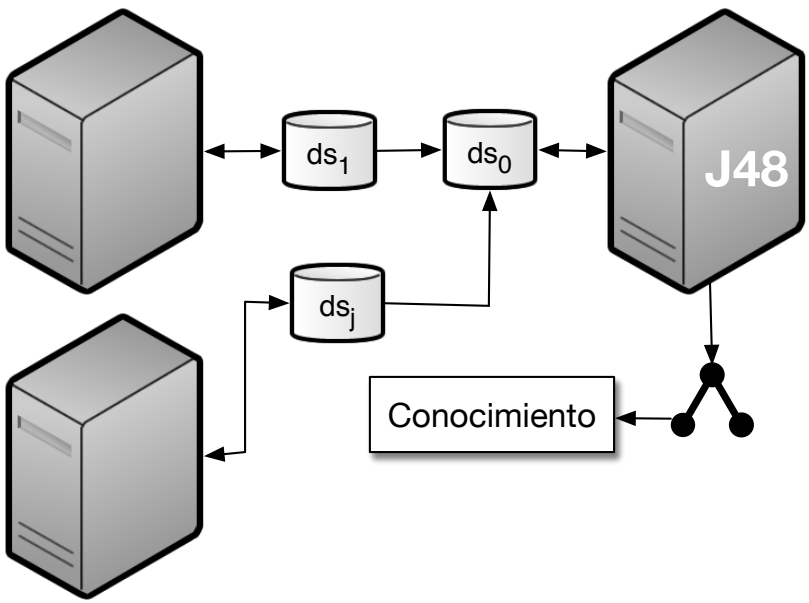
Universidad Veracruzana

¿Y si los datos están distribuidos?

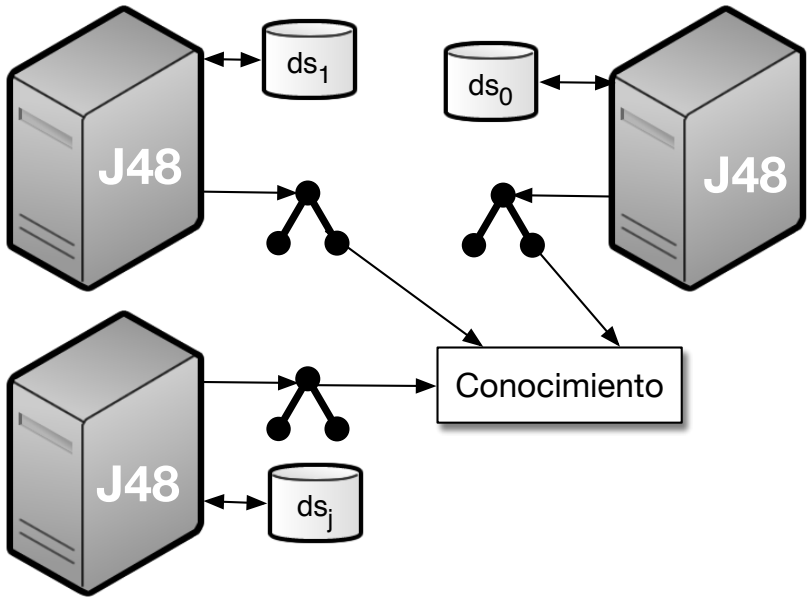
Universidad Veracruzana

Posibles soluciones

Centralización de Datos



Distribución del Aprendizaje



Universidad Veracruzana

Posibles problemas

- Distribución de los datos.
- Algoritmos centralizados.
- Datos heterogéneos.
- Comunicación.
- Privacidad.
- Cómputo concurrente y/o distribuido.
- Datos cambiantes y/o crecientes.
- Escalabilidad.

Universidad Veracruzana

Soluciones basadas en agentes

- Frameworks y Sistemas:
  - JAM [14], BODHI [7], Papyrus [3], i-Analyst [11], EMADS [2], SMAJL [16], GLS [17], etc.
- Centralizantes.
- Meta-aprendizaje.
  - Votación.
  - Arbitraje.
  - Combinación.
- Problemas existentes:
  - Modelos de agencia débil.
  - Extensibilidad.
  - Generalidad.
- Solución: JaCa-DDM [8]

Universidad Veracruzana

El Modelo JaCa-DDM

- Se ha definido un modelo abstracto JaCa-DMM, a partir de dos constructores básicos:

  Estrategia. Flujo de trabajo basado en agentes Jason y artefactos CArtAgO, basados en Weka [15] y MOA [4].

  Despliegue. La manera en que los componentes del sistema se ubican en la arquitectura de cómputo distribuida donde se ejecutará la estrategia. Basada en nodos CArtAgO.

Universidad Veracruzana

Estrategia JaCa-DDM

- Se define como $\langle Ags, Arts, Params, ag_1 \rangle$, donde:
  - $Ags = \{ag_1, \ldots, ag_n\}$ es el conjunto de programas de agente.
  - $Arts = \{art_1, \ldots, art_m\}$ es el conjunto de tipos de artefacto.
  - $Params = \{param_1 : tipo_1, \ldots param_k : tipo_k\}$ con los tipos de datos $\in \{int, bool, double, string\}$.
  - $ag_1 \in Ags$ el agente responsable de la estrategia.
- Las interacciones entre agentes se detallan usando diagramas de secuencia UML (o AUML).

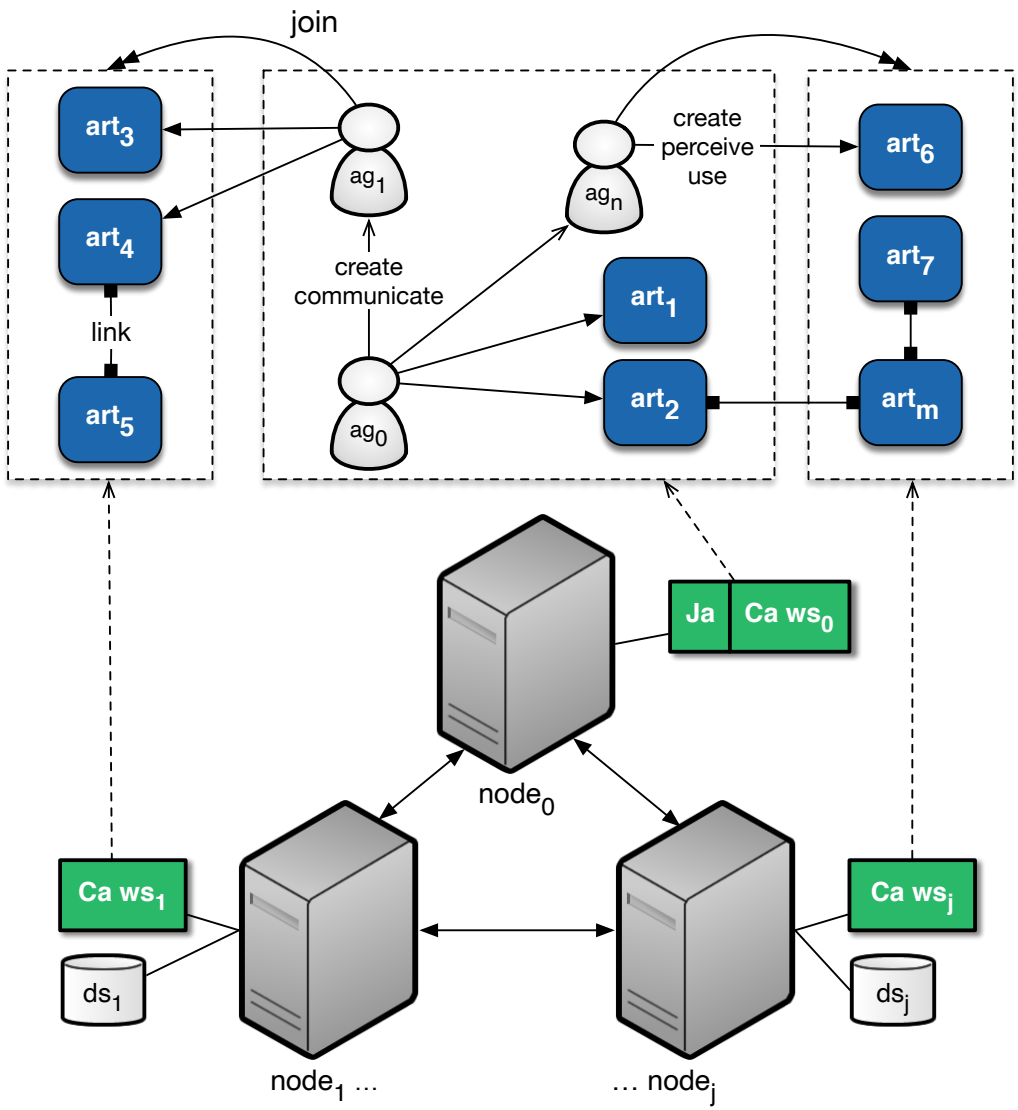
Universidad Veracruzana

Sistema de Despliegue JaCa-DDM

- Se define como $\langle Nodos, DS, Arts, Estrat, Config, ag_0 \rangle$, donde:
  - $Nodos = \{nodo_0, nodo_1, \ldots, nodo_j\}$ son los nodos CArtAgO del sistema. Cada $nodo_i = \langle nodo_i, IPAdr : Port \rangle$.
  - $DS = \{ds_1, \ldots, ds_j\}$ son las fuentes de datos para cada nodo, exceptuando el $nodo_0$.
  - $Arts = \{art_1, \ldots, art_i\}$ son los tipos de artefactos primitivos usados en el sistema de despliegue.
  - *Estrat* es una estrategia JaCa-DDM.
  - $Config = \langle \delta, \pi \rangle$ denota la configuración, donde:
    - $\delta = \langle (ag, nodo, i), \ldots \rangle$ son asignaciones de i copias de un programa de agente ag focalizando en cierto $nodo$ ($ag\#i$).
    - $\pi = \{(param : val), \ldots\}$ son pares parámetro-valor, conforme a *Estrat*.
- Se genera vía la interfaz de JaCa-DDM (formato XML).

Universidad Veracruzana

Arquitectura JaCa-DDM



Universidad Veracruzana

Tipos de artefactos primitivos (1)

Tipo de Artefacto	Descripción
`Classifier`*XXX*	Una herramienta de clasificación, capaz de aprender nuevos modelos y usarlos para clasificar ejemplos, basada en Weka/MOA. *XXX* puede substituirse por alguno de los siguientes algoritmos: J48, VFDT, Bagging, BaggingEnsemble.
`Directory`	Una herramienta de localización de servicios para agentes, artefactos y espacios de trabajo (al estilo páginas amarillas y blancas).
`Evaluator`	Una herramienta para la evaluación de modelos basada en Weka.
`GUI`	Una intefaz gráfica para configurar y lanzar experimentos.
`FileManager`	Una herramienta para escribir archivos ARFF, basado en Weka.
`InstancesBase`	Un repositorio de ejemplos de aprendizaje basado en Weka.

Universidad Veracruzana

Tipos de artefactos primitivos (2)

Tipo de Artefacto	Descripción
`LogBook`	Una herramienta para generar reportes de resultados de los experimentos.
`Oracle`	Una herramienta para distribuir conjuntos de entramiento centralizados en los diferentes nodos de la arquitectura, basado en Weka.
`TrafficMonitor`	Un sniffer para medir la carga de la red de la arquitectura, basado en utilidades de terceros. Su uso es opcional.
`Utils`	Una navaja suiza para los agentes, con multiples utilidades.

Universidad Veracruzana

Agentes obligatorios en JaCa-DDM

Responsable de Sistema de Despliegue. Es el encargado de configurar la arquitectura y evaluar los resultados obtenidos (ag_0).

Responsable de estrategia. Es el encargado de iniciar y terminar una estrategia determinada (ag_1).

Universidad Veracruzana

ag_0 responsable del Sistema de Despliegue (1)

Configurar el experimento. Interactiva, conforme a la estrategia adoptada, gracias a un artefacto de tipo `GUI`, incluye la distribución de los agentes (δ) y la inicialización de parámetros (π).

Distribuir datos dinámicamente. Cuando JaCa-DDM se usa para simular escenarios distriuidos, ag_0 puede crear un artefacto de tipo `Oracle` para distribuir un conjunto de entrenamiento centralizado, en los diferentes nodos de la arquitectura.

Monitoreo de tráfico. Es posible evaluar el tráfico de datos que genera la ejecución de un experimento opcionalmente, usando un artefacto de tipo `TrafficMonitor`.

Universidad Veracruzana

ag_0 responsable del Sistema de Despliegue (2)

Despliegue de agentes. La creación de *Ags* de la estrategia adoptada. La definición del sistema de despliegue le indica cuantos agentes de cada tipo debe crear en los nodos de la arquitectura. La inicialización de estos agentes consiste en:

- `node(NodeName).`
- `ipNode0(IPAdress:Port).`
- `data(FilePath).`
- `param(ParamId,Val).`
- Planes primitivos.

Universidad Veracruzana

ag_0 responsable del Sistema de Despliegue (3)

Evaluar modelos. A través de artefactos de tipo `Evaluator` y `LogBook`, ag_0 puede evaluar los modelos aprendidos y generar reportes sobre su desempeño. Un mensaje $\langle ag_1, tell, finish(ArtId)\rangle$, donde ag_1 es el agente responsable de la estrategia adoptada y *ArtId* es el artefacto que almacena el modelo obtenido, le avisa que puede proceder a la evaluación.

Limpieza. Si la evaluación de un experimento implica repeticiones iteradas del mismo, como en el caso de la validación cruzada, el ag_0 reinicia agentes y artefactos de acuerdo a los requerimientos del método de evaluación.

Universidad Veracruzana

ag_1 responsable de la estrategia

Iniciar el proceso de aprendizaje. Un mensaje $\langle ag_0, achieve, start \rangle$, le avisa a ag_1 que debe lanzar el proceso de aprendizaje.

Finalizar el proceso de aprendizaje. Una vez finalizado el proceso de aprendizaje, ag_1 debe enviar un mensaje $\langle ag_0, tell, finish(ArtId) \rangle$, avisando al responsable del despliegue ag_0 que el proceso ha finalizado y que el modelo obtenido se encuentra en el artefacto *ArtId*.

Universidad Veracruzana

Configuración (1)

- La estrategia *Estrat* adoptada. Al definirse una estrategia, debe generarse un descriptor XML. Ejemplos de estos descriptores pueden encontrarse en los protocolos incluídos en la distribución en el directorio `sampleProtocols`.
- La dirección IP y los puertos usados por los nodos en *Nodos*.
- La distribución de agentes (δ) y la inicialización de parámetros (π).

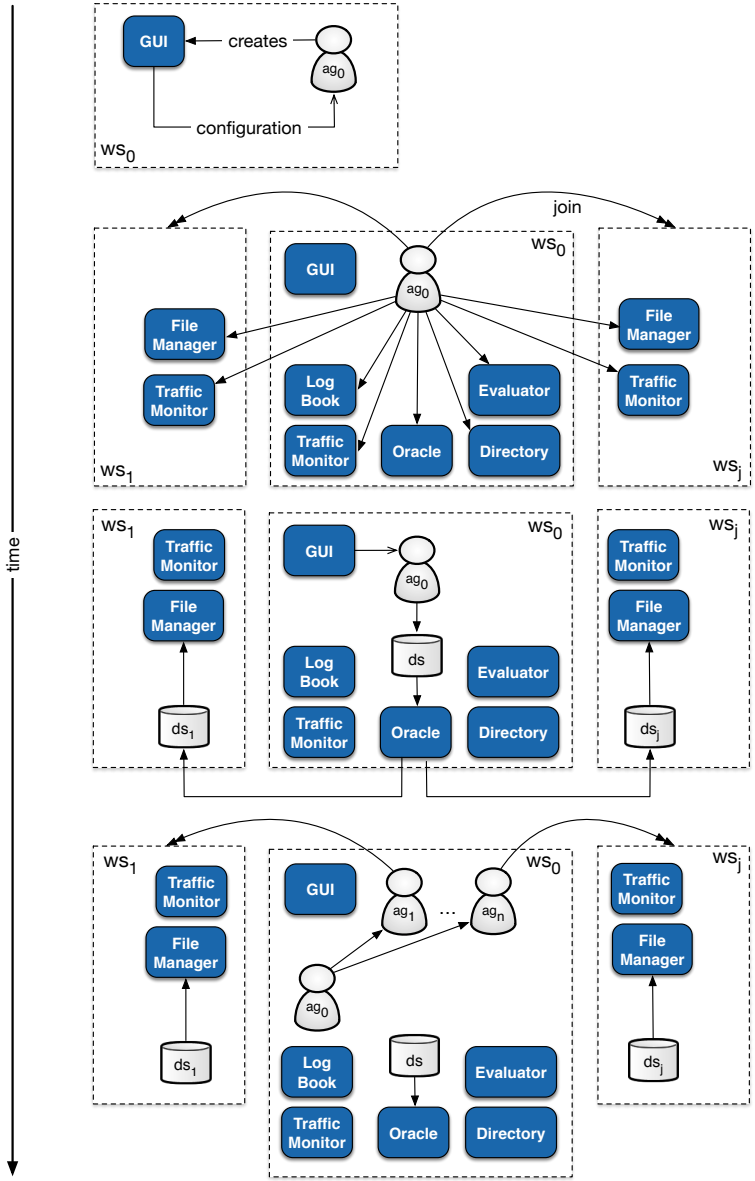
Universidad Veracruzana

Configuración (2)

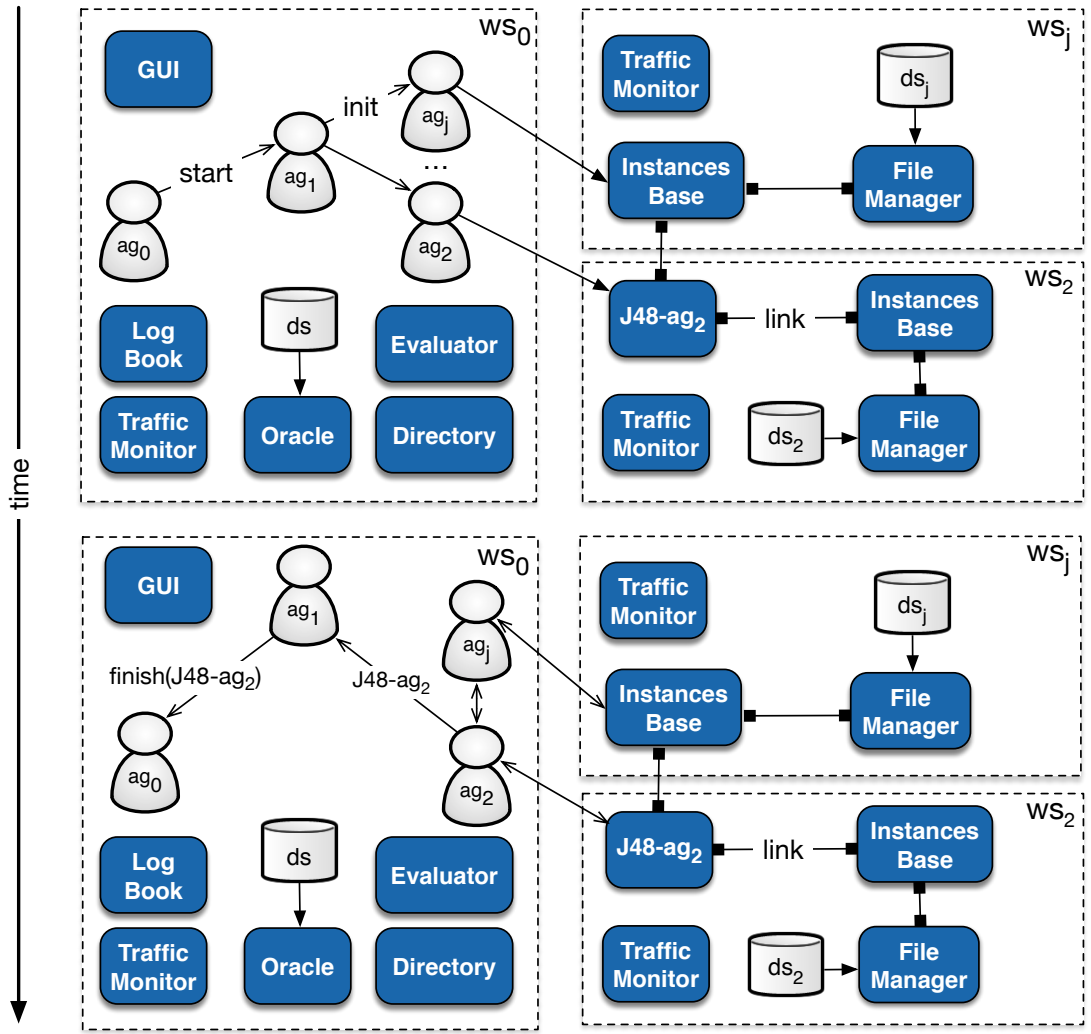
- Distribución dinámica de datos. Un conjunto de entrenamiento se encuentra en el $nodo_0$ y es distribuido en los demás nodos de la arquitectura, usando un artefacto de tipo `Oracle`:
  - `hold-out`. Conforme un parámetro que define el tamaño del conjunto de prueba, en términos de un porcentaje de ejemplos a usarse con este propósito. El resto de los ejemplos disponibles se distribuye de manera estratificada entre los nodos definidos.
  - `cross-validation`. Un parámetro indicando el número de pliegues, determina el radio de ejemplos usados para la prueba y los entrenamientos. Las particiones de entrenamiento se distribuyen en los nodos definidos en el sistema.
- Distribución estática.

Universidad Veracruzana

Flujo de trabajo (1)

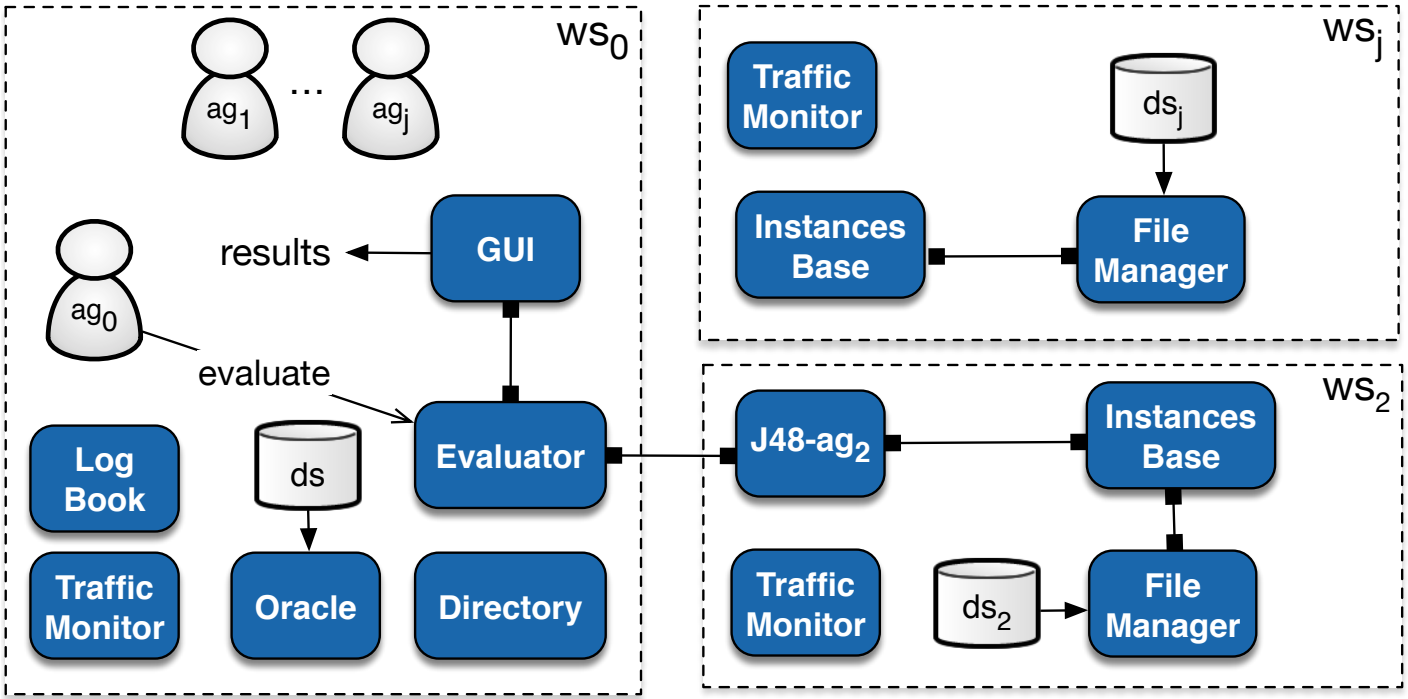


Flujo de trabajo (3–4)



Universidad Veracruzana

Flujo de trabajo (5)

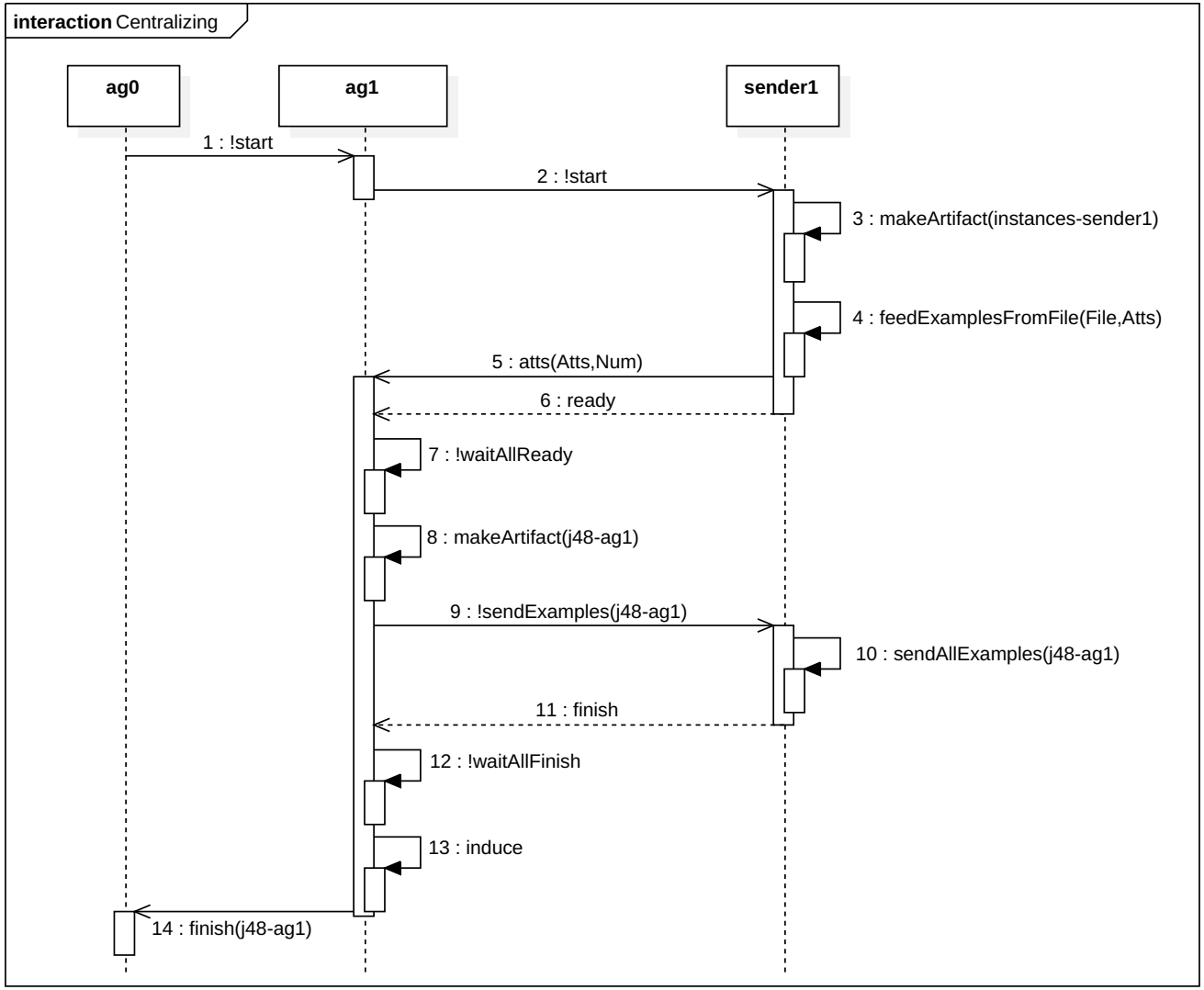


Estrategias incluídas en JaCa-DDM

- Centralizadas:
  - Centralized
  - Centralized Bagging
- Centralizantes:
  - Centralizing
  - Round
- Basadas en Meta-Aprendizaje:
  - Distributed Bagging
  - Random Forest
- Basadas en Windowing:
  - Counter
  - Round Counter
  - Parallel Round Counter
  - Counter GPU
  - Parallel Counter GPU
  - Parallel Counter GPU Extra

Universidad Veracruzana

Estrategia centralizante



Universidad Veracruzana

contactPerson I

```
29  +begin_process: true
30      <-
31
32      !consut_type_count("sender", Count);
33      for (.range(X,1,Count))
34      {
35        .concat("sender",X,NAg);
36        .send(NAg, achieve, start);
37      }.
38
39
40  +atts(N,A)[source(sender1)] : true //receives meta info about dataset
41    <-
42      !begin.
43
44
45
46  +!begin: true
47    <-
48      !consut_type_count("sender", Count);
49      +numAgents(Count);
```

Universidad Veracruzana

contactPerson II

```
50      !waitAllReady;
51      ?atts(N,A);
52      ?param("Pruning", ParamVal);
53      ?param("Classifier",  Classifier);
54      if(Classifier == "VFDT")
55      {
56          makeArtifact("coordinatorj48","artifacts.VFDT",[b(B, a(N, A))],
            ↪  IdJ48);
57      }
58      else
59      {
60          makeArtifact("coordinatorj48","artifacts.ClassifierJ48",[b(B,
            ↪  a(N, A)), ParamVal], IdJ48);
61      };
62      !register_artifact("coordinatorj48");
63      //begin
64
65      for (.range(X,1,Count))
66      {
67        .concat("sender",X,NAg);
68        .send(NAg, achieve, sendExamples("coordinatorj48"));
```

Universidad Veracruzana

contactPerson III

```
69        };
70        !waitAllFinish;
71        induce;
72        .send(experimenter, tell, finish("coordinatorj48")).
73
74
75
76  +ready(Name)[source(Ag)]: true
77    <-
78    +agent(Name). //to know the name of the agents
79
80  +!waitAllReady: numAgents(Nags) & .count(ready(Name),E) & E == Nags
81    <-
82    println("All agents are ready").
83
84  +!waitAllReady: true
85    <- .wait(50);
86    !waitAllReady.
87
88  +!waitAllFinish: numAgents(Nags) & .count(finish(Name),E) & E == Nags
89    <-
```

contactPerson IV

```
90      println("All agents have finished").
91
92    +!waitAllFinish: true
93      <- .wait(50);
94      !waitAllFinish.
```

Universidad Veracruzana

sender I

```
30  +!start: true
31    <-
32      .my_name(Name);
33      .concat("base", Name, ArName);
34      makeArtifact(ArName,"artifacts.InstancesBase",[],IdBase);
35
36      ?data(File);
37      feedExamplesFromFile(File, Atts)[artifact_id(IdBase)];
38      .term2string(Attt, Atts); //meta info
39      -+Attt;
40      ?atts(N, A);
41
42      if(Name == sender1)
43      {
44        .send(contactPerson1, tell, atts(N, A));
45      };
46      .send(contactPerson1, tell, ready(Name)).
47
48
49  +!sendExamples(To): true
50    <-
```

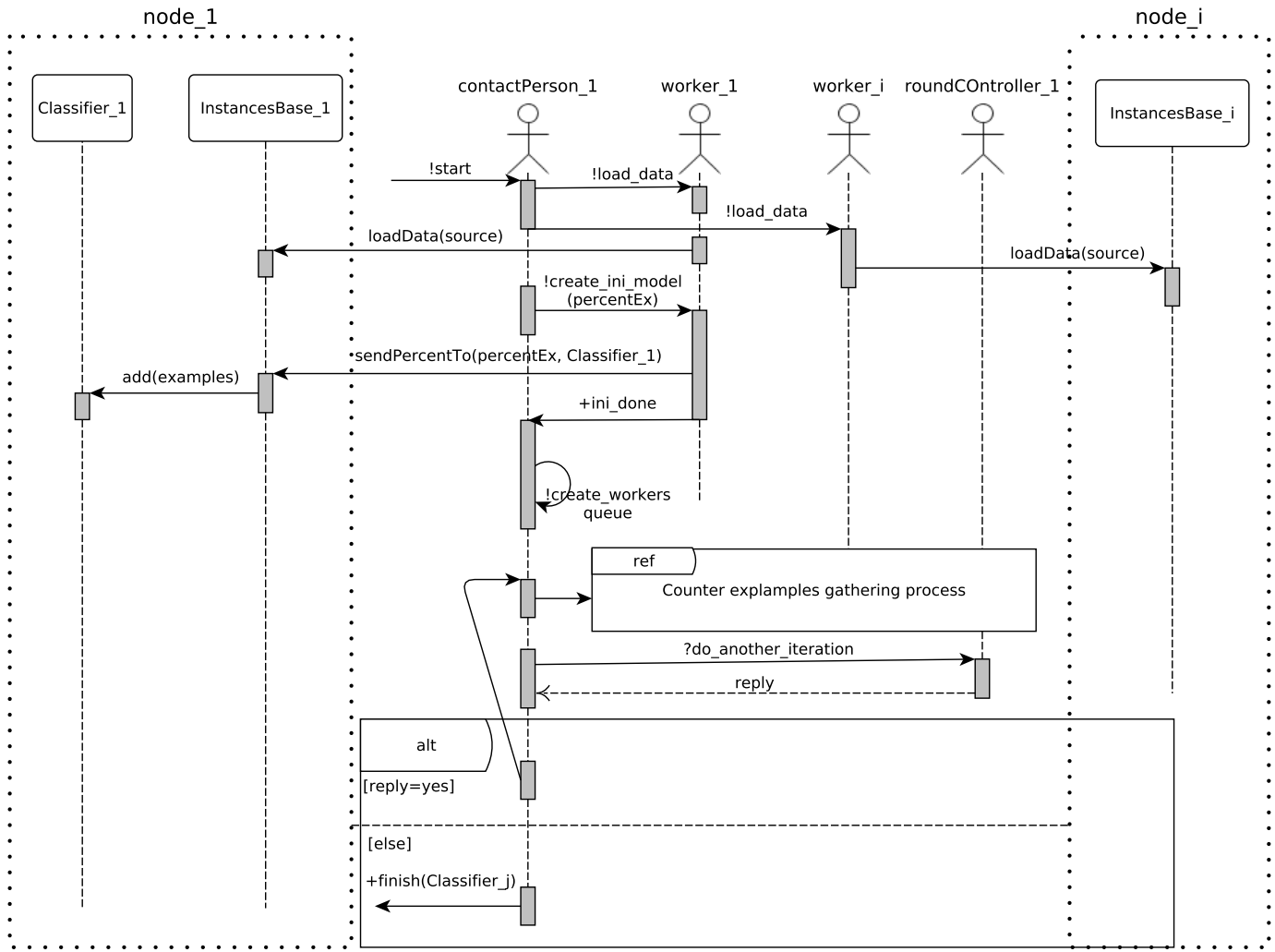
Universidad Veracruzana

sender II

```
51                ?ipServer(IPSer); //all agents have this believe
52        joinRemoteWorkspace("default",IPSer,_); //go to the server
53              getArtifactId(To, IdArt); //use directory
54              quitWorkspace; //return to previous workspace
55        sendAllExamples(IdArt);
56        .my_name(Name);
57        .send(contactPerson1, tell, finish(Name)).
```

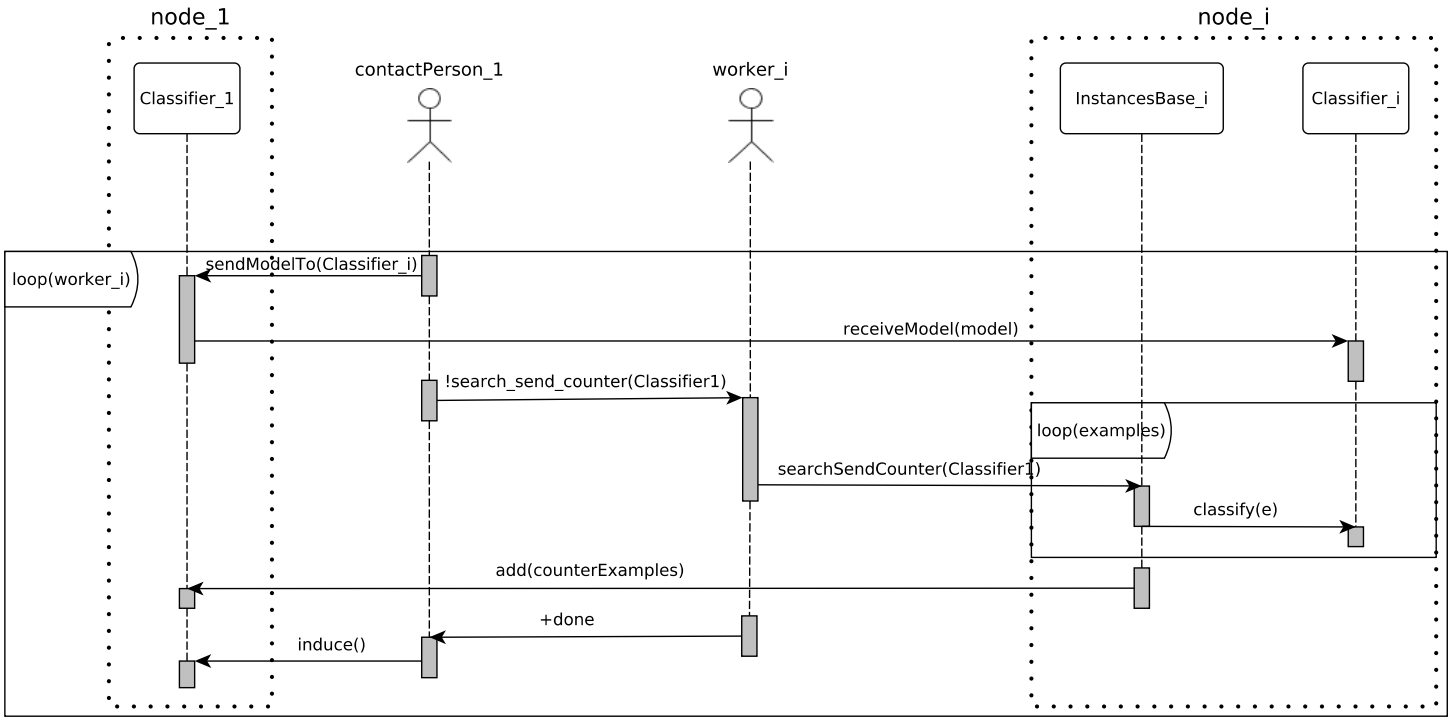
Universidad Veracruzana

Estrategia contra ejemplos



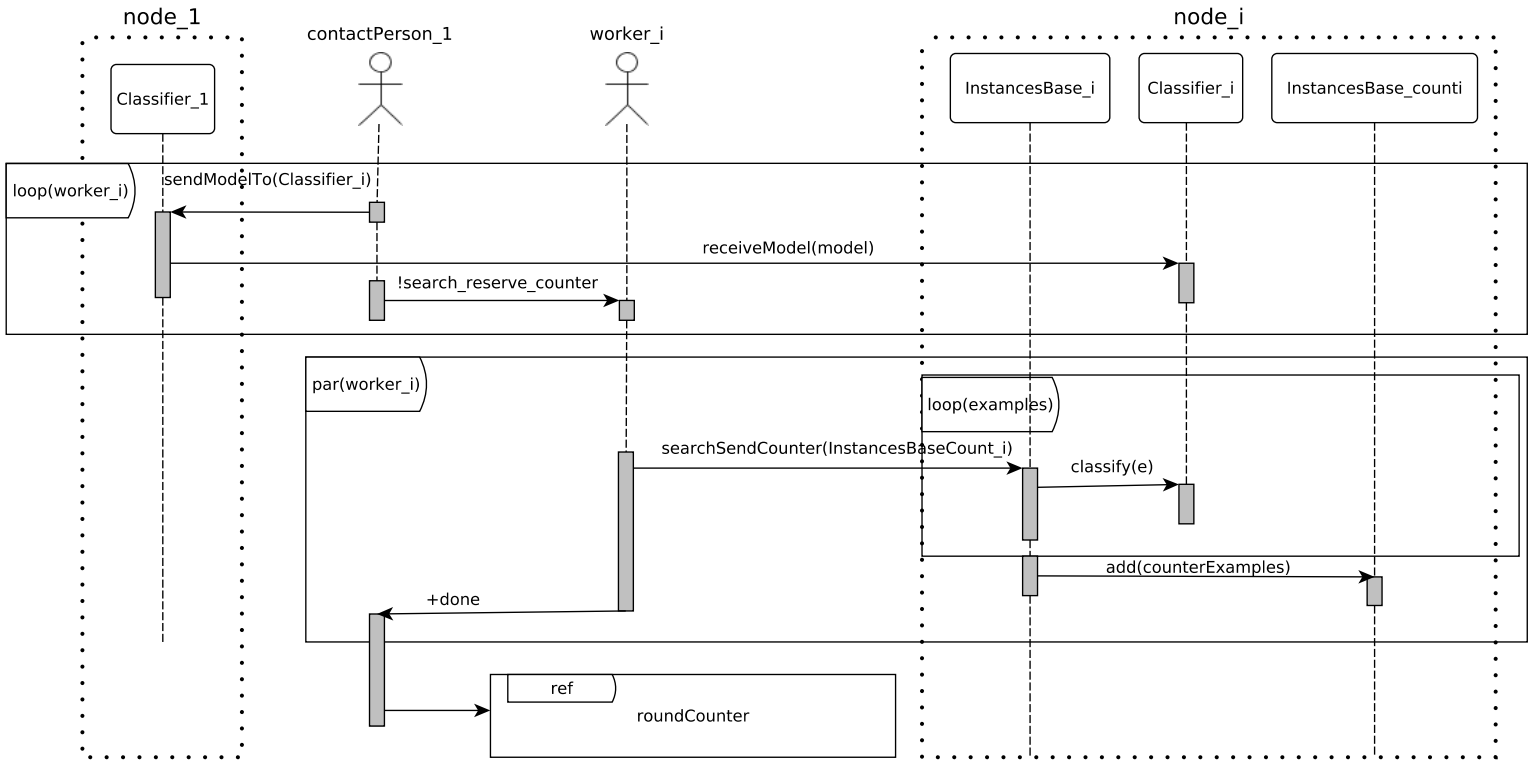
Universidad Veracruzana

Búsqueda de contra ejemplos



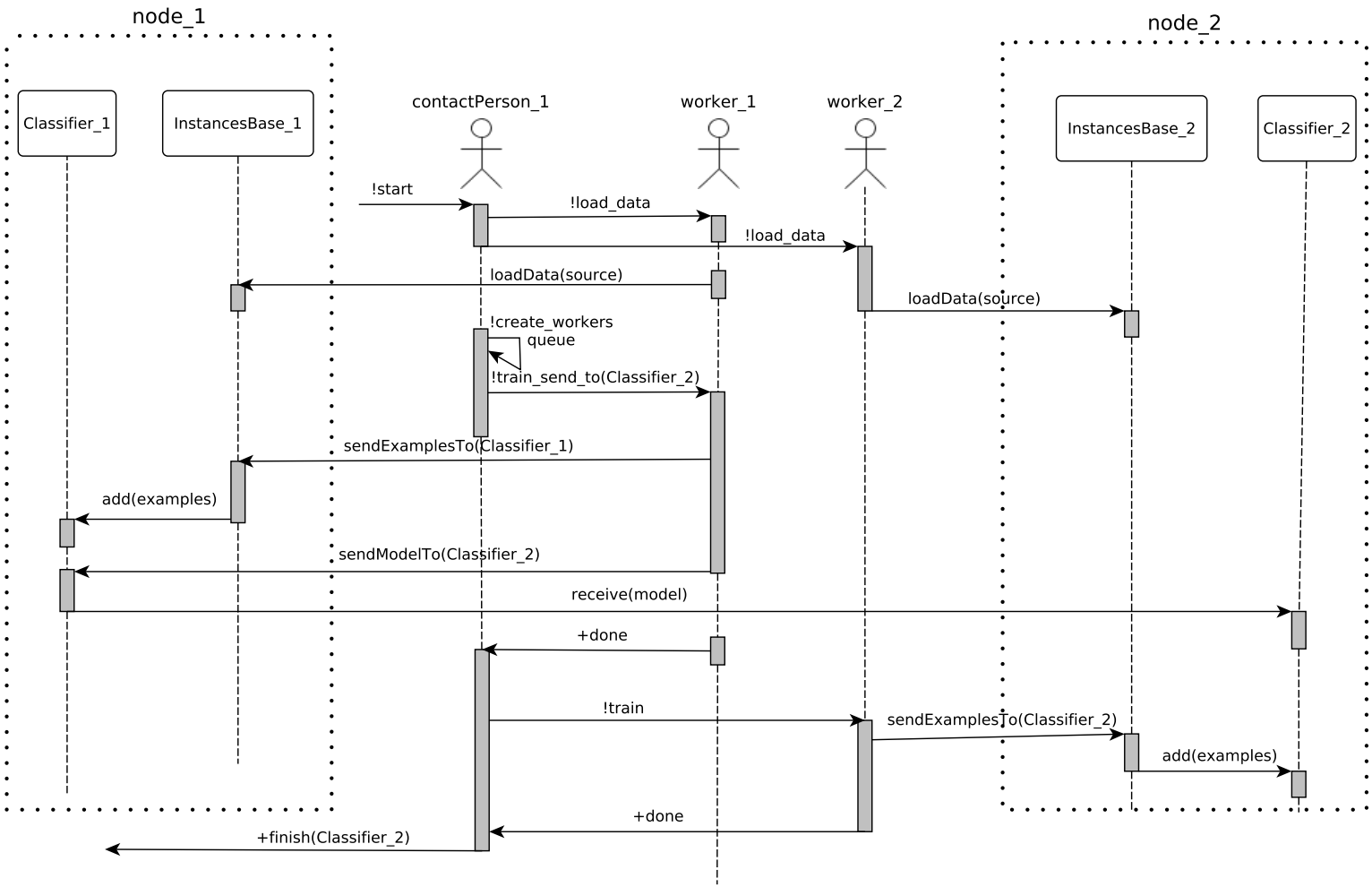
Universidad Veracruzana

Estrategia contra ejemplos paralela



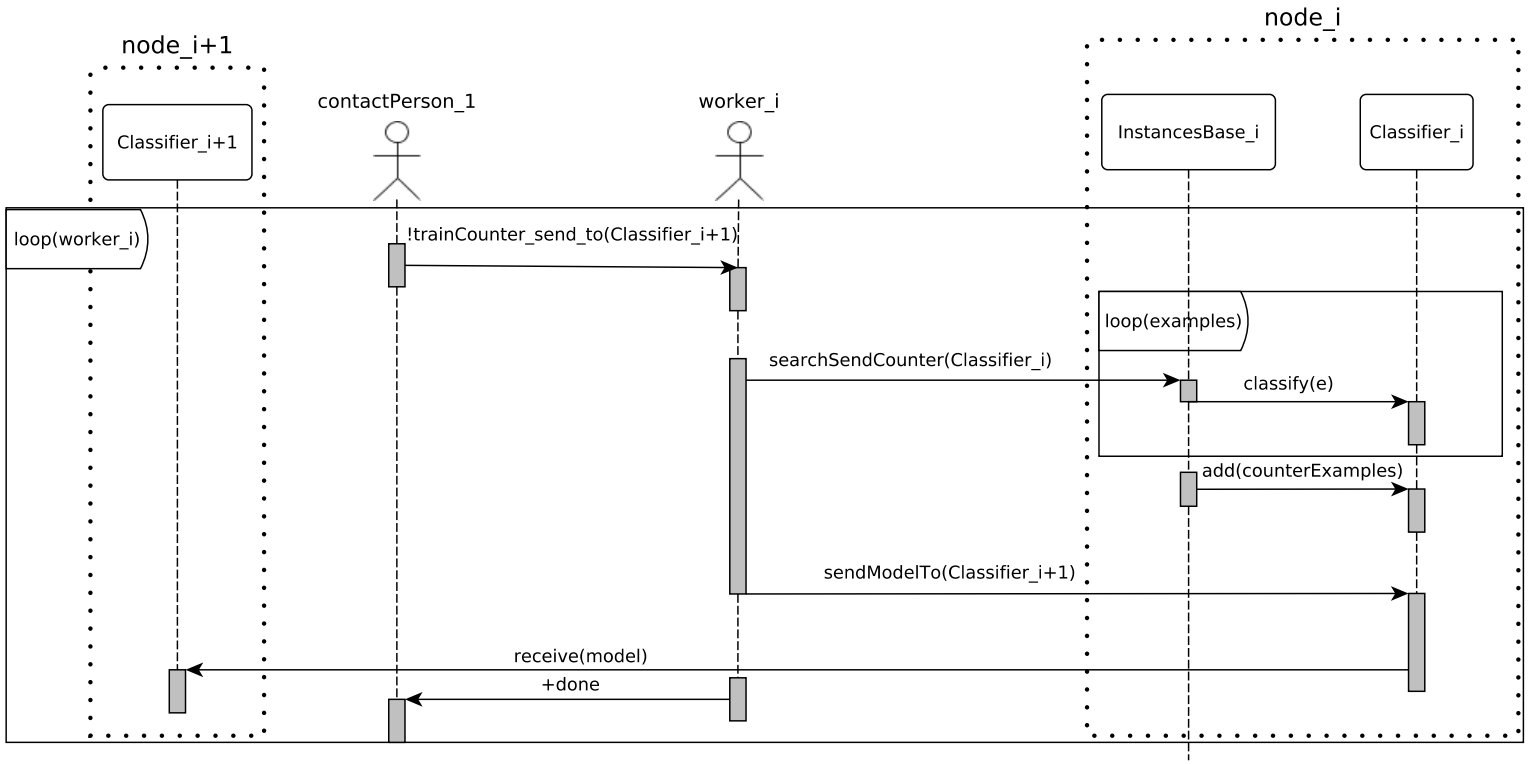
Universidad Veracruzana

Estrategia ronda



ersidad Veracruzana

Estrategia ronda por contra ejemplos



Universidad Veracruzana

Windowing [12]

Algoritmo

```
1:  function Windowing(Exs)
2:      Window ← sample(Exs)
3:      Exs ← Exs − Window
4:      repeat
5:          stopCond ← true
6:          model ← induce(Window)
7:          for ex ∈ Exs do
8:              if classify(model, ex) ≠ class(ex) then
9:                  Window ← Window ∪ {ex}
10:                 Exs ← Exs − {ex}
11:                 stopCond ← false
12:             end if
13:         end for
14:     until stopCond
15:     return model
16: end function
```

Ventajas

- Reduce ejemplos.
- Mantiene precisión.

Desventajas

- Costo inducción.
- Costo clasificación.

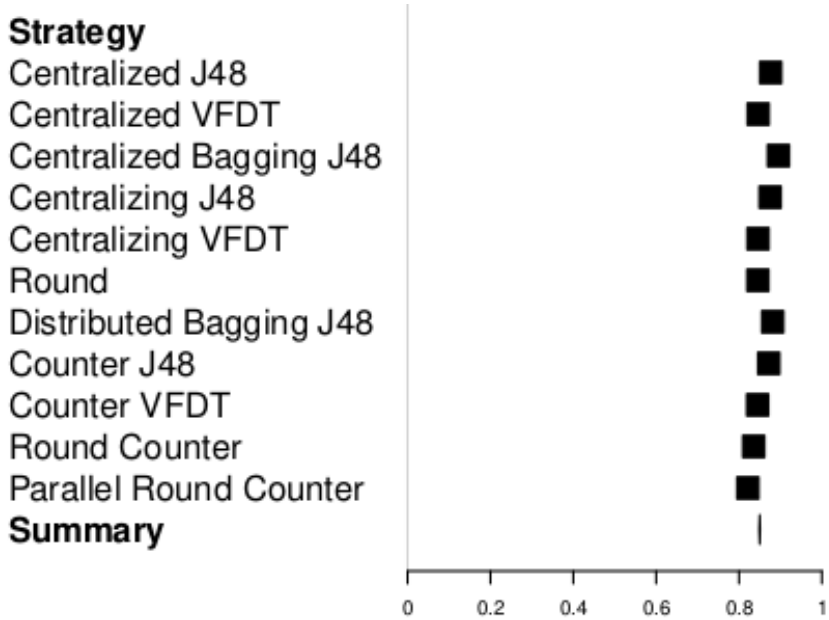
Universidad Veracruzana

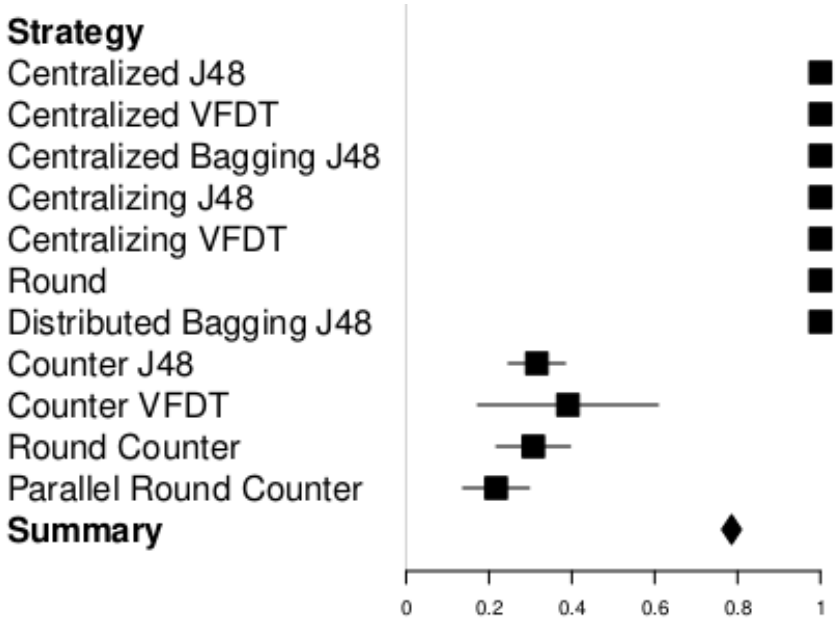
Segundos resultados [10]

Datos	Ejemplos	Atributos	Clases
adult	48842	15	2
australian	690	15	2
breast	683	10	2
credit-g	1000	21	2
covtypeNorm	581012	55	7
diabetes	768	9	2
ecoli	336	8	8
german	1000	21	2
hypothyroid	3772	30	4
imdb-D	120919	1002	2
kr-vs-kp	3196	37	2
letter	20000	17	26
mushroom	8124	23	2
poker	829201	11	10
segment	2310	20	7
sick	3772	30	2
splice	3190	61	3
waveform-5000	5000	41	3

Universidad Veracruzana

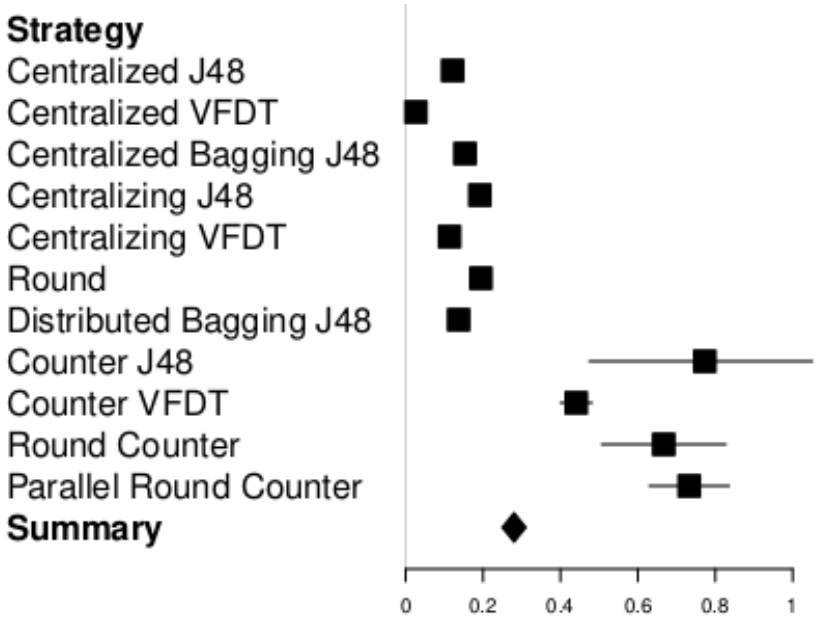
Precisión vs Ejemplos usados (Forest Plots)

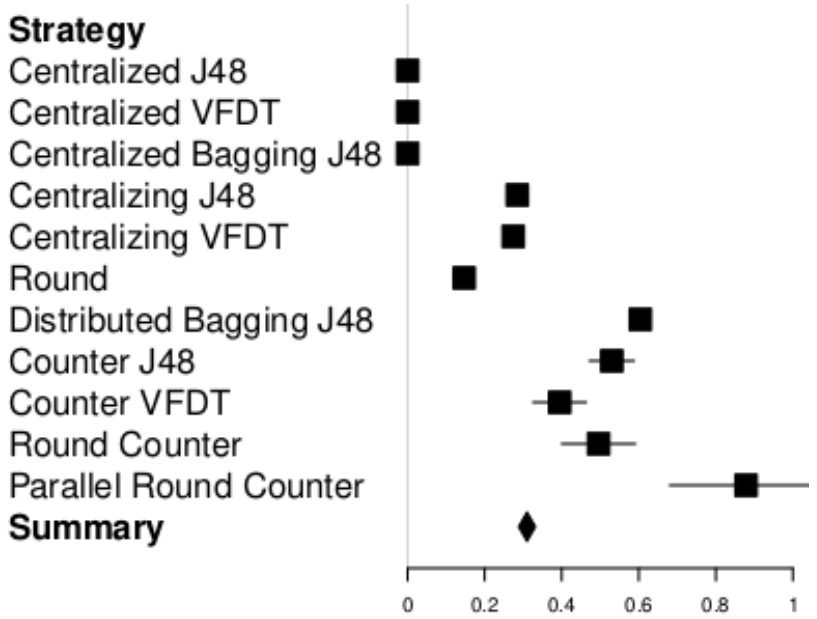




Universidad Veracruzana

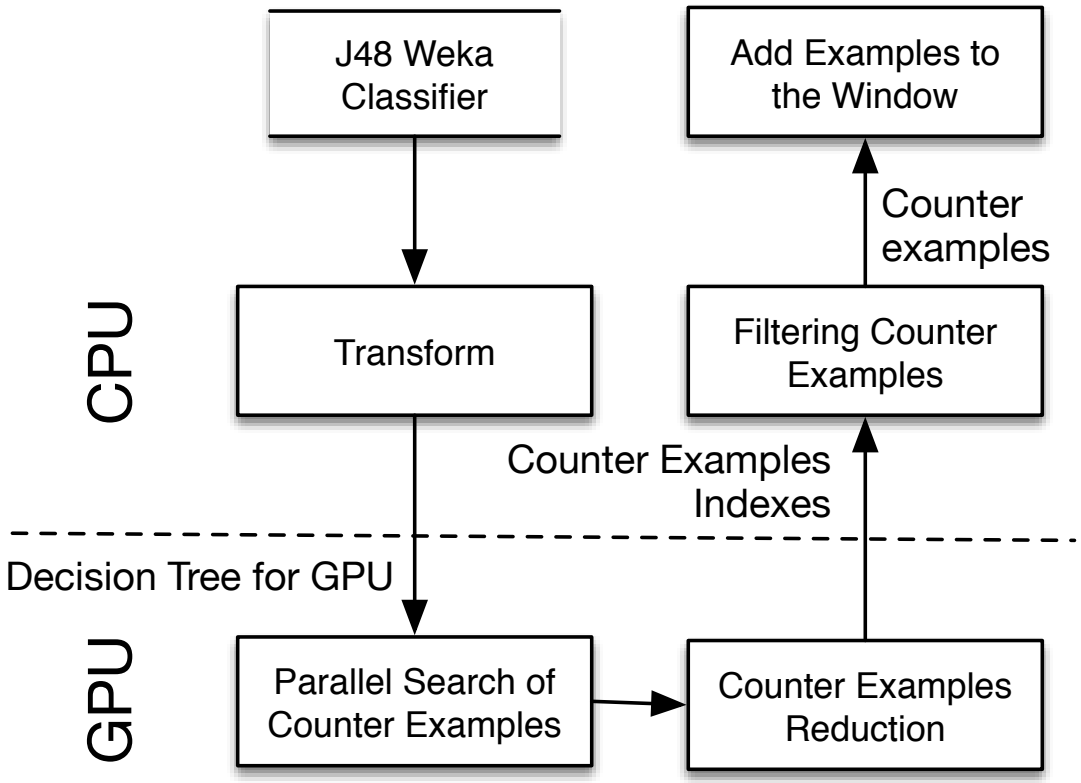
Tiempo vs Tráfico





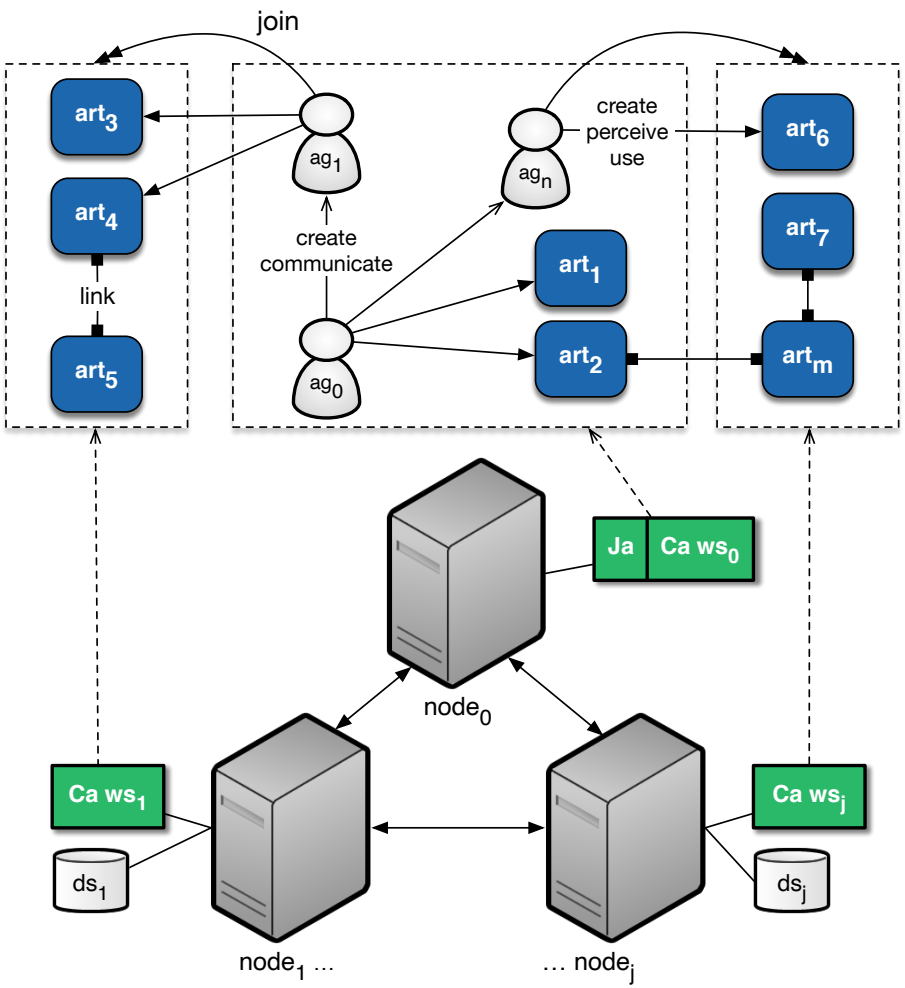
Universidad Veracruzana

Primera revisión: GPU [9]



Universidad Veracruzana

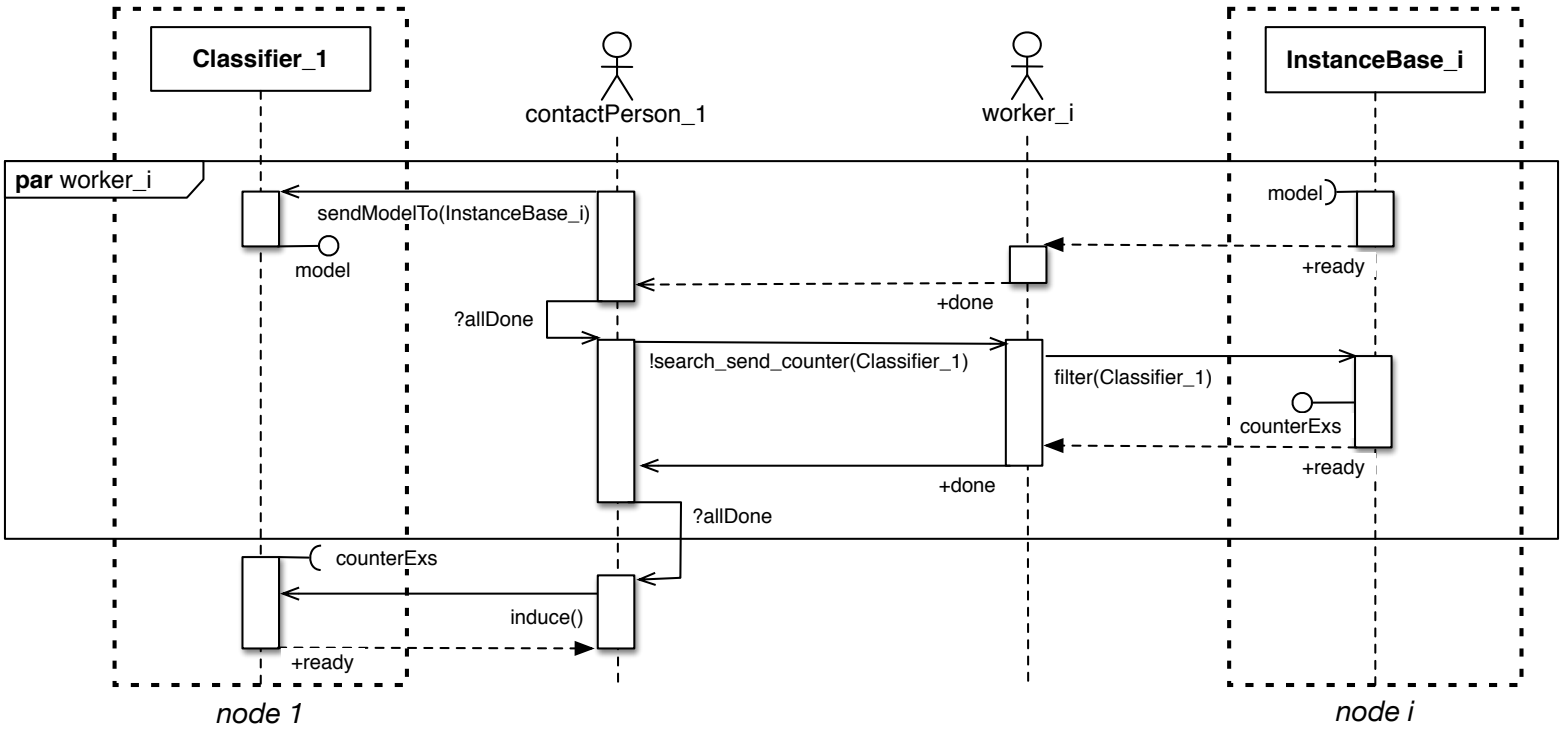
Cluster



- 2 procesadores Xeon a 2.40 GHz, con 4 cores, dos hilos c/u.
- 24 GB of RAM.
- GPU Tesla C2050 con 448 cores, 6 GB de memoria.

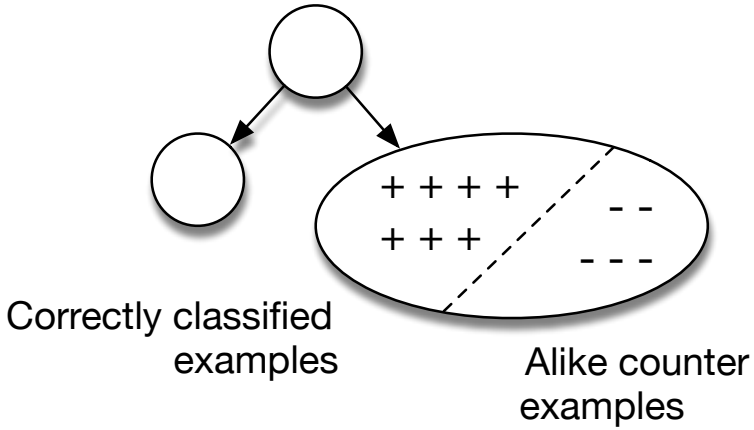
Universidad Veracruzana

Parallel Counter GPU



Universidad Veracruzana

Segunda revisión: Parallel Counter GPU Extra



Universidad Veracruzana

Datos

Datos	Ejemplos	Atributos	Clases
airlines	539383	8	2
covtypeNorm	581012	55	7
KDDCup99	4898431	42	23
poker-lsn	829201	11	10
pixelSegCancer [1]	1434060	31	2

Universidad Veracruzana

Precisión

Dataset	Estrategia	Precisión		Test de Wilcoxon
airlines	**Parallel Counter GPU Extra**	65.36 ±	0.25	–
airlines	Weka Centralized	66.34 ±	0.11	Won
airlines	Parallel Counter GPU	66.26 ±	0.12	Won
airlines	Centralizing VFDT	65.24 ±	0.27	Tie
airlines	Bagging	66.45 ±	0.13	Won
airlines	Random Forest	66.76 ±	0.11	Won
covtypeNorm	**Parallel Counter GPU Extra**	92.17 ±	0.52	–
covtypeNorm	Weka Centralized	94.59 ±	0.04	Won
covtypeNorm	Parallel Counter GPU	93.10 ±	0.34	Won
covtypeNorm	Centralizing VFDT	76.83 ±	0.35	Lost
covtypeNorm	Bagging	94.99 ±	0.10	Won
covtypeNorm	Random Forest	78.34 ±	0.39	Lost
KDDCup99	**Parallel Counter GPU Extra**	99.98 ±	0.01	–
KDDCup99	Weka Centralized	99.99 ±	0.01	Tie
KDDCup99	Parallel Counter GPU	99.96 ±	0.01	Lost
KDDCup99	Centralizing VFDT	99.97 ±	0.01	Lost
KDDCup99	Bagging	99.99 ±	0.01	Tie
KDDCup99	Random Forest	99.97 ±	0.01	Lost
poker-lsn	**Parallel Counter GPU Extra**	99.53 ±	0.59	–
poker-lsn	Weka Centralized	99.78 ±	0.01	Tie
poker-lsn	Parallel Counter GPU	98.67 ±	0.46	Lost
poker-lsn	Centralizing VFDT	87.78 ±	1.92	Lost
poker-lsn	Bagging	99.71 ±	0.01	Tie
poker-lsn	Random Forest	96.73 ±	0.25	Lost

Memoria y Tiempo

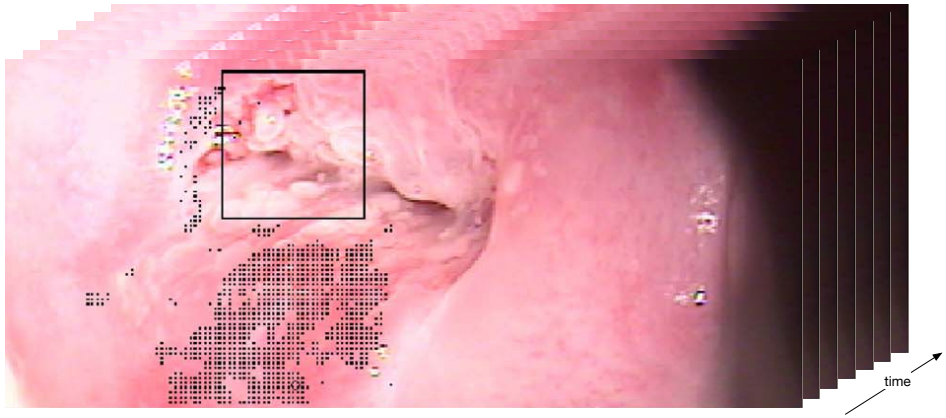
Dataset	Estrategia	%Ejemplos	Tiempo(Segs.)
airlines	**Parallel Counter GPU Extra**	51.21 ± 0.03	435.04 ± 106.28
airlines	Weka Centralized	100.00 ± 0.00	1164.66 ± 211.76
airlines	Parallel Counter GPU	94.95 ± 0.01	1810.78 ± 446.47
airlines	Centralizing VFDT	100.00 ± 0.00	4.67 ± 0.53
airlines	Bagging	100.00 ± 0.00	144.67 ± 4.47
airlines	Random Forest	100.00 ± 0.00	123.82 ± 3.89
covtypeNorm	**Parallel Counter GPU Extra**	43.28 ± 0.04	817.30 ± 253.27
covtypeNorm	Weka Centralized	100.00 ± 0.00	855.41 ± 97.88
covtypeNorm	Parallel Counter GPU	48.44 ± 0.01	1089.03 ± 277.06
covtypeNorm	Centralizing VFDT	100.00 ± 0.00	8.96 ± 0.56
covtypeNorm	Bagging	100.00 ± 0.00	149.35 ± 5.37
covtypeNorm	Random Forest	100.00 ± 0.00	44.47 ± 4.38
KDDCup99	**Parallel Counter GPU Extra**	4.29 ± 0.01	93.23 ± 6.671
KDDCup99	Weka Centralized	100.00 ± 0.00	1688.91 ± 363.89
KDDCup99	Parallel Counter GPU	9.28 ± 0.01	199.72 ± 45.62
KDDCup99	Centralizing VFDT	100.00 ± 0.00	56.17 ± 1.307
KDDCup99	Bagging	100.00 ± 0.00	371.51 ± 19.39
KDDCup99	Random Forest	100.00 ± 0.00	132.43 ± 21.23
poker-lsn	**Parallel Counter GPU Extra**	8.80 ± 0.01	22.93 ± 3.51
poker-lsn	Weka Centralized	100.00 ± 0.00	174.26 ± 28.55
poker-lsn	Parallel Counter GPU	9.56 ± 0.01	24.90 ± 8.05
poker-lsn	Centralizing VFDT	100.00 ± 0.00	4.25 ± 0.47
poker-lsn	Bagging	100.00 ± 0.00	64.09 ± 5.49
poker-lsn	Random Forest	100.00 ± 0.00	236.34 ± 14.37

Universidad Veracruzana

Segmentación basada en pixeles

- Secuencias de imágenes de colposcopia, presentando posibles lesiones cervicales pre-cancerosas:

- 38 pacientes.
- Pre-procesamiento con FIJI [13].
- 1,434,060 pixeles de entrenamiento.
- 30 atributos.
- 6 clases, agrupadas en 2.



Universidad Veracruzana

Precisión, sensibilidad y especificidad

Estrategia	Precisión		Wilcoxon test	Sen	Esp
Parallel Counter GPU Extra	67.61 ±	19.32	–	60.96	64.83
Weka Centralized	63.68 ±	18.44	Lost	60.80	61.60
Centralizing VFDT	53.34 ±	20.58	Lost	53.10	58.51
Bagging	64.25 ±	21.78	Lost	65.40	59.16
Random Forest	58.88 ±	23.71	Lost	68.78	49.34
Acosta2009	67.00 ±	n/a	n/a	71.00	59.00

Universidad Veracruzana

Número de ejemplos y tiempo

Estrategia	% Ejemplos		Tiempo (Seg.)	
Parallel Counter GPU Extra	37.00 ±	3.52	3782.26 ±	1094.21
Weka Centralized	100.00 ±	0.00	6436.64 ±	923.16
Centralizing VFDT	100.00 ±	0.00	32.03 ±	2.61
Bagging	100.00 ±	0.00	1138.83 ±	108.83
Random Forest	100.00 ±	0.00	1817.10 ±	179.18
Acosta2009	n/a		n/a	

Universidad Veracruzana

Trabajo Futuro

- Mejorar la facilidad de uso:
  - Interfaz como un servicio Web
  - Lenguaje DSL / Gráfico basado en AUML
  - Mejorar transparencia de CArtAgO
- Analizar la relación precisión vs ejemplos.
- Analizar generalización del Windowing.
- Buscar Otras Aplicaciones:
  - Análisis de tendencias en twitter
  - Simulación Social [5]

Universidad Veracruzana

El sistema

- Repositorio: `http://sourceforge.net/projects/jacaddm/`
- Pág web: `http://jacaddm.sourceforge.net`
- Ayuda: `http://sourceforge.net/p/jacaddm/wiki/Home/`
- Requisitos:
  - Jason >= 1.4.X
  - Java >= 1.7
- Opcionalmente:
  - CUDA
  - jcuda (`http://www.jcuda.org`)

Universidad Veracruzana

Distribución

- La distribución contiene tres directorios a saber:
  - node0 código del sitio principal, a ejecutar el la computadroa que ejecutará los agentes Jason. Correr `run.sh`
  - defaultNode código a ejecutar en las demás computadoras del sistema. Correr `run.sh IPAddress Port`
  - sampleProtocols Los protocolos discutidos en clase.

Universidad Veracruzana

Ejecución en una computadora

- Dar de alta al menos un nodo local (desde el directorio defaultNode):

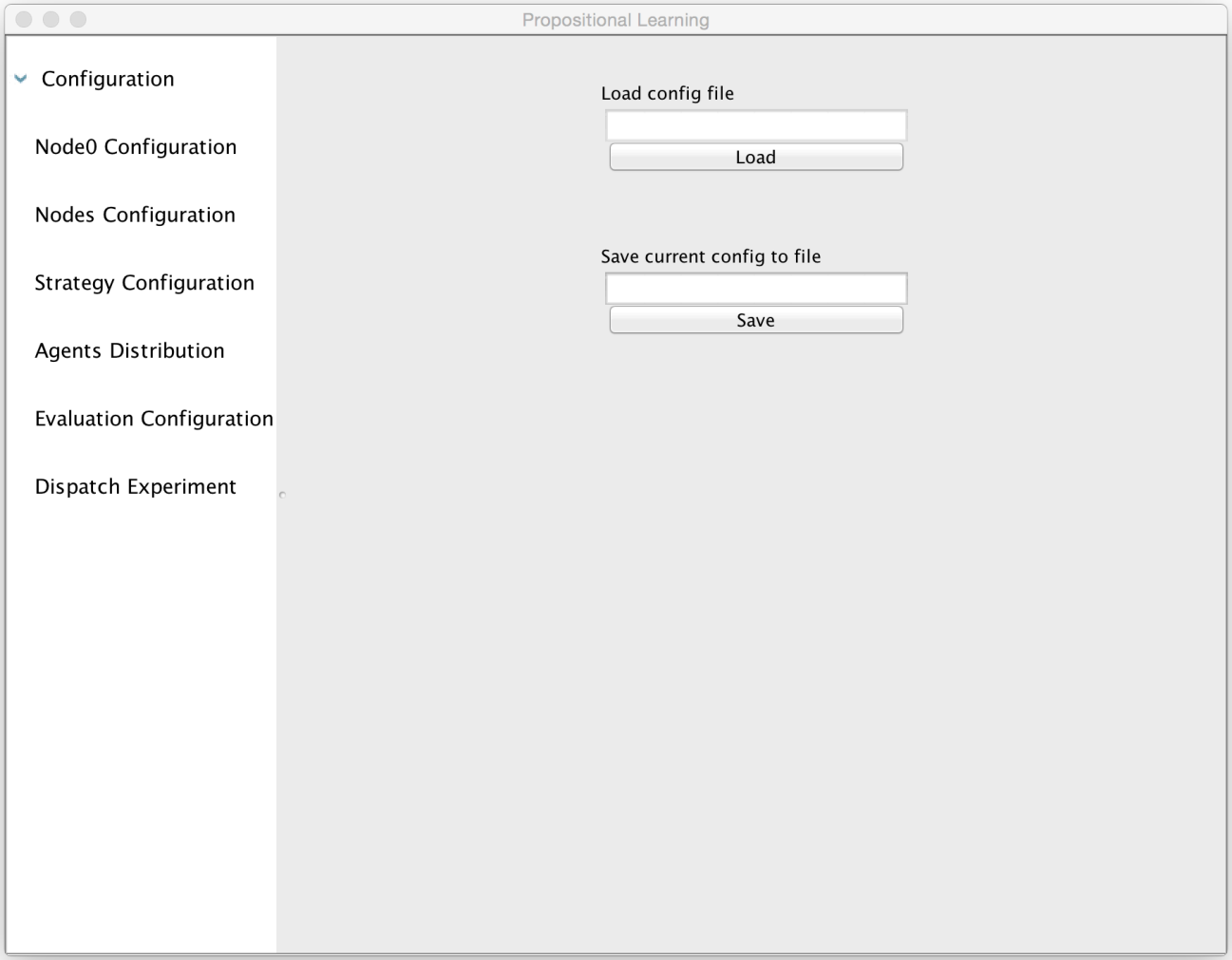
```
1  > ./run.sh localhost 1024 CArtAgO Node Ready on localhost:1024
```

- Ejecutar el sistema principal (desde el directorio node0):

```
1  >./run.sh
```

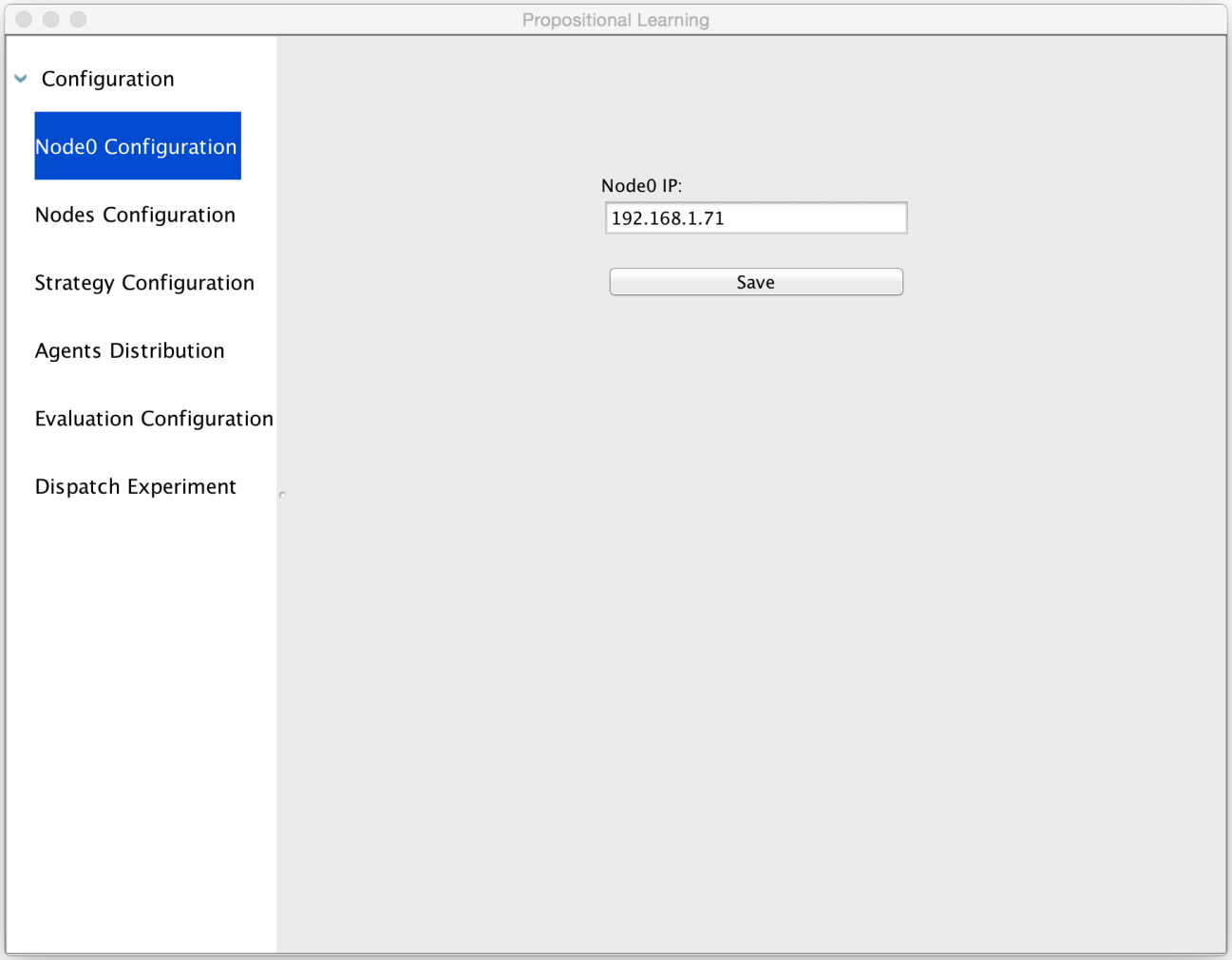
Universidad Veracruzana

Interfaz gráfica



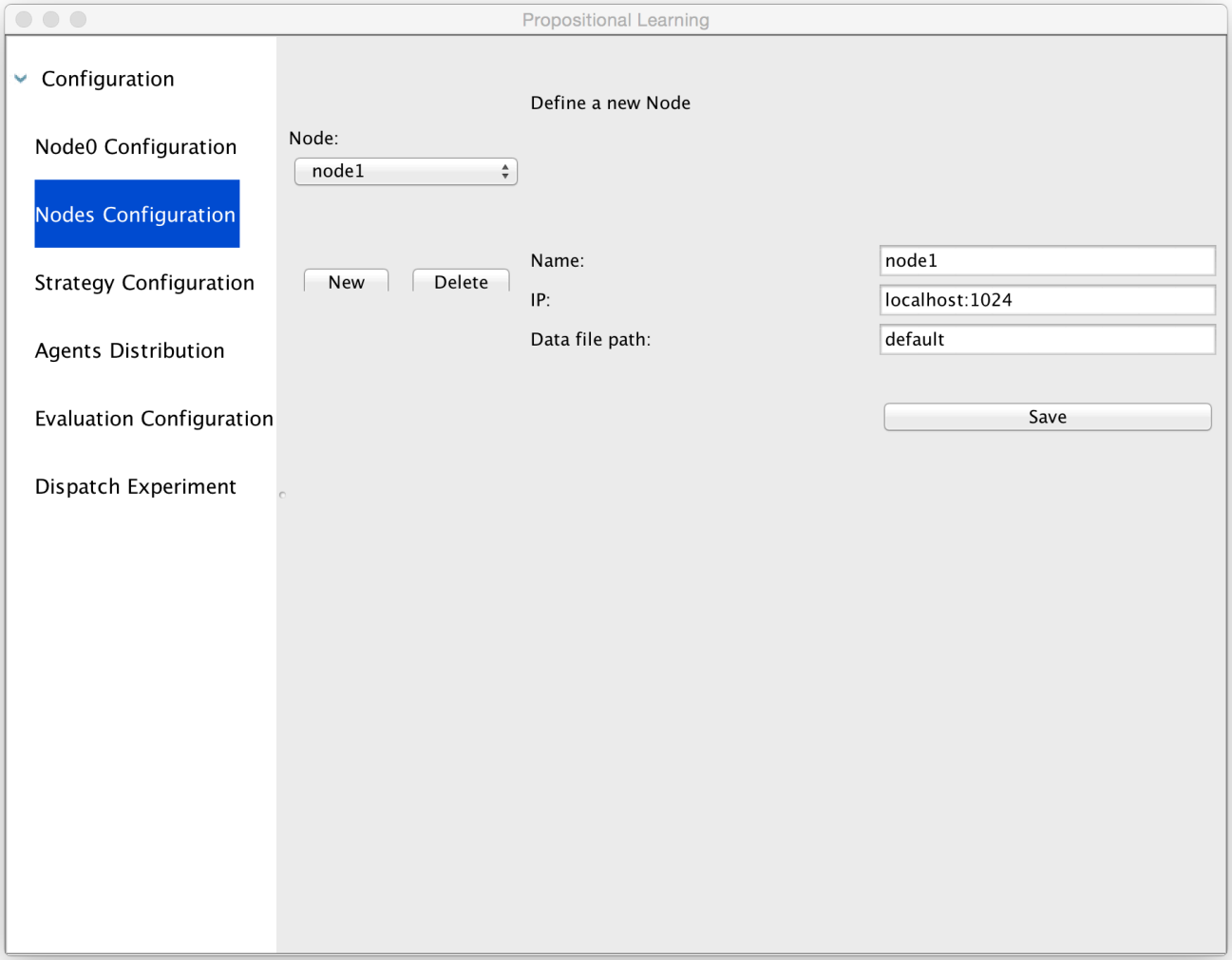
Universidad Veracruzana

Verificando al nodo node0



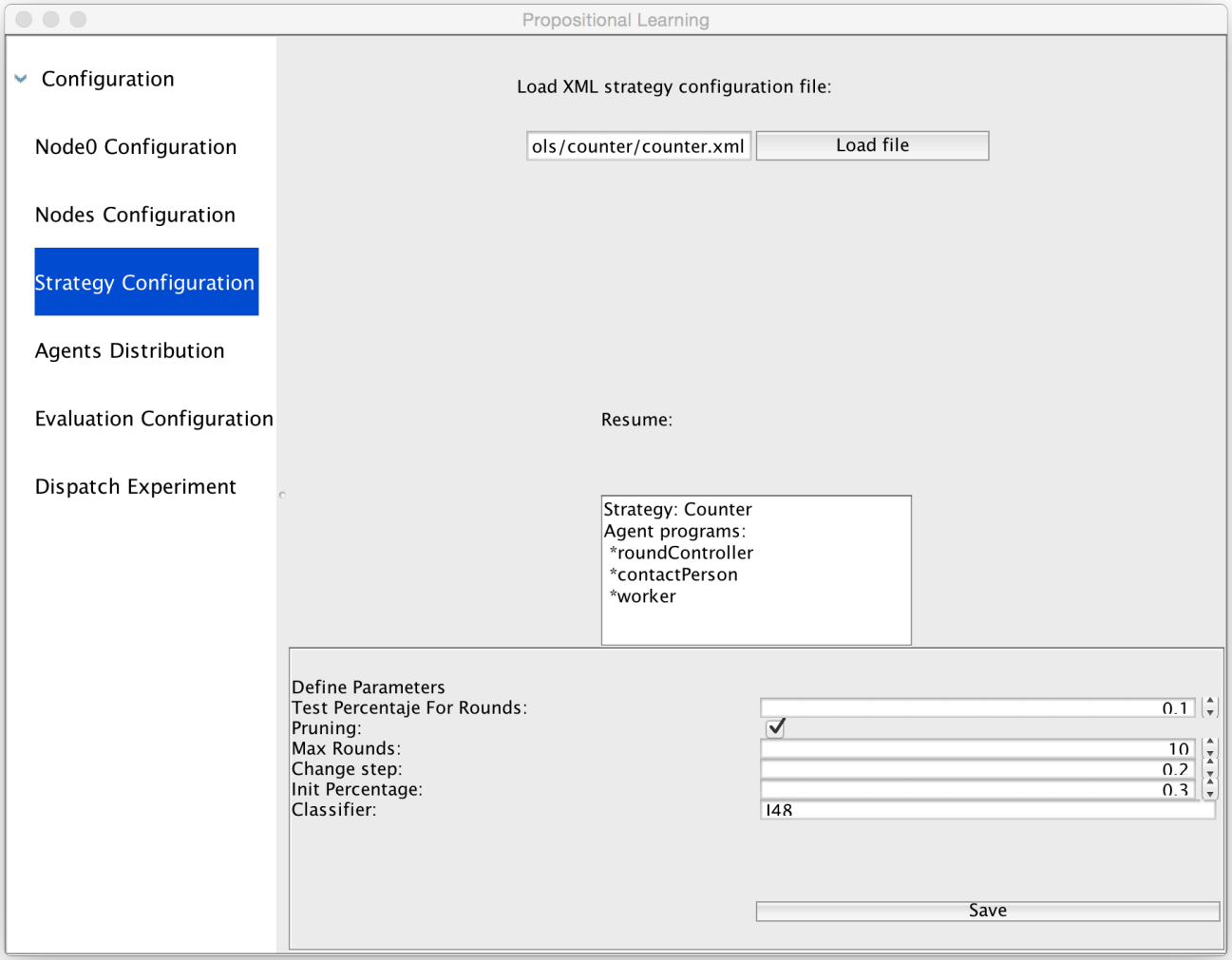
Universidad Veracruzana

Agregando al nodo node1



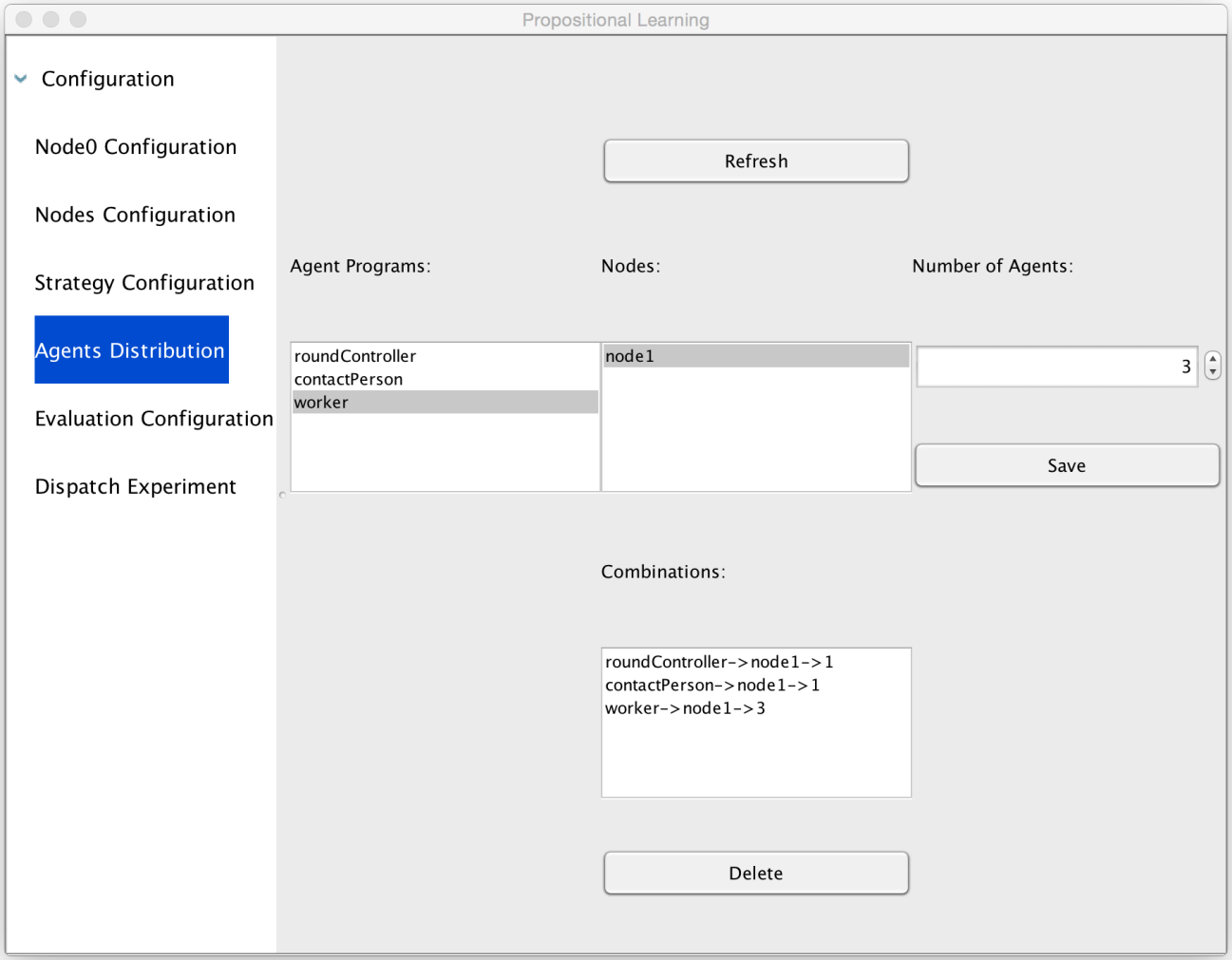
Universidad Veracruzana

Parámetros de la estrategia (π)



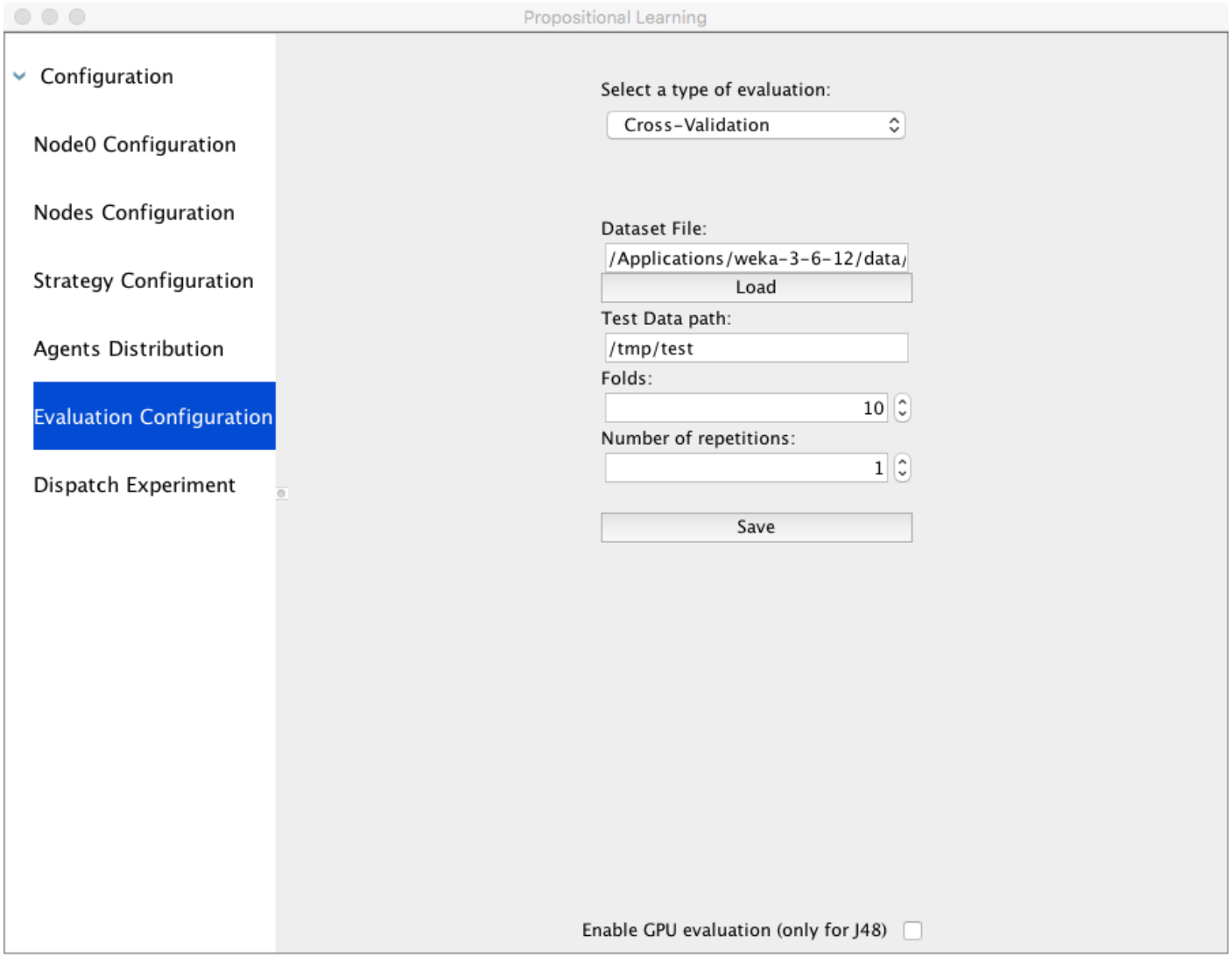
Universidad Veracruzana

Distribución de agentes (δ)



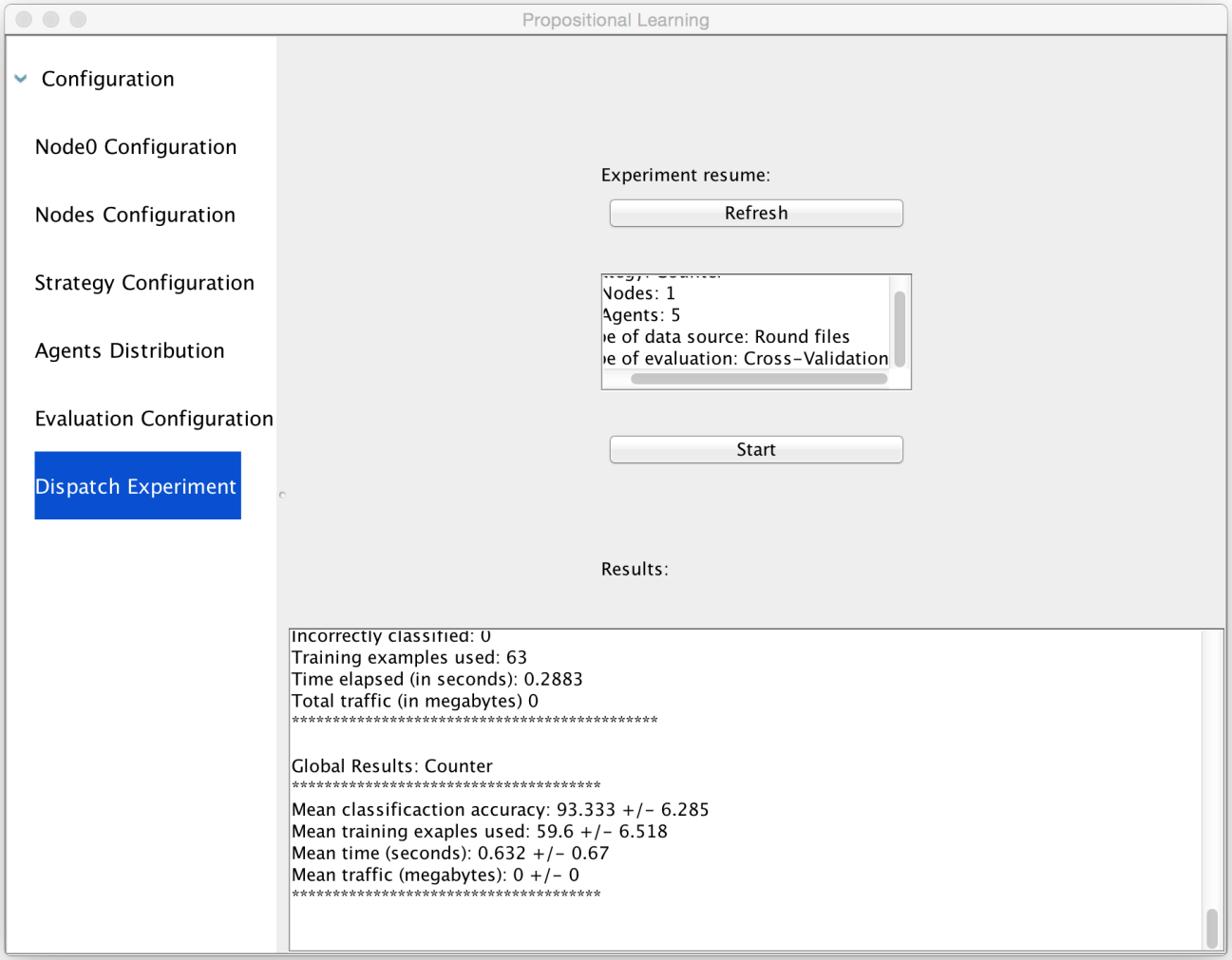
Universidad Veracruzana

Evaluación experimento



Universidad Veracruzana

Ejecutando el experimento



Universidad Veracruzana

Inicialización I

```
1  package jacaDDM;
2
3  import cartago.*;
4  import jason.asSyntax.*;
5  import weka.core.*;
6  import weka.core.converters.ConverterUtils.DataSource;
7
8  public class InstancesBase extends Artifact {
9    private DataSource source;
10   private Instances instances;
11   private Instances test;
12   private Instances train;
13
14   void init() {
15     Atom none = new Atom("none");
16     java.util.List<Term> attributesNames = null;
17     // Observable properties
18     defineObsProperty("numberOfAttributes",0);
19     defineObsProperty("class",none);
20     defineObsProperty("attributes",attributesNames);
```

Universidad Veracruzana

Inicialización II

```
21        defineObsProperty("arffSource","none");
22        defineObsProperty("stratified",0);
23    }
```

Universidad Veracruzana

Carga de ejemplos de entrenamiento I

```
1   @OPERATION void loadARFF(String arffPath) throws Exception {
2       // Read all instances in the file arffPath
3       source = new DataSource(arffPath);
4       instances = source.getDataSet();
5       // Make the last attribute be the class
6       int numAttributes = instances.numAttributes();
7       instances.setClassIndex(numAttributes-1);
8       // Upgrade observable properties
9       ObsProperty numberOfAttributesProp =
        ↪  getObsProperty("numberOfAttributes");
10      ObsProperty classAttributeProp = getObsProperty("class");
11      ObsProperty attributesProp = getObsProperty("attributes");
12      ObsProperty arffSourceProp = getObsProperty("arffSource");
13      numberOfAttributesProp.updateValue(numAttributes-1);
14      Atom className = new Atom(instances.classAttribute().name());
15      classAttributeProp.updateValue(className);
16      ListTerm attributesNamesList = new ListTermImpl();
17      for(int i=0;i<numAttributes-1;i++){
18        Atom attributeName = new Atom(instances.attribute(i).name());
19        attributesNamesList.add(attributeName);
```

Universidad Veracruzana

Carga de ejemplos de entrenamiento II

```
20        }
21        attributesProp.updateValue(attributesNamesList.getAsList());
22        arffSourceProp.updateValue(arffPath);
23    }
```

Universidad Veracruzana

Imprimiendo contenido

```
1  @OPERATION void printARFF(){
2      System.out.println(instances);
3    }
4
5  @OPERATION void printTest(){
6      System.out.println(test);
7    }
8
9  @OPERATION void printTrain(){
10     System.out.println(train);
11   }
```

Universidad Veracruzana

Obteniendo meta-información de los datos I

```
1  @OPERATION void getAttributeName(int i, OpFeedbackParam<Atom> attrib){
2      Atom attributeName = new Atom(instances.attribute(i-1).name());
3      attrib.set(attributeName);
4    }
5
6  @OPERATION void getAttributesNames(OpFeedbackParam<java.util.List<Term>>
   ↪  attributeNames){
7      int numAttributes = instances.numAttributes();
8      ListTerm attributesNamesList = new ListTermImpl();
9      for(int i=0;i<numAttributes-1;i++){
10       Atom attributeName = new Atom(instances.attribute(i).name());
11       attributesNamesList.add(attributeName);
12     }
13     attributeNames.set(attributesNamesList.getAsList());
14   }
15 @OPERATION void getNumberOfAttributes(OpFeedbackParam<Integer>
   ↪  numAttributes){
16     numAttributes.set(instances.numAttributes()-1);
17   }
18
```

Universidad Veracruzana

Obteniendo meta-información de los datos II

```
19  @OPERATION void getNumberOfInstances(OpFeedbackParam<Integer>
    ↪  numInstances){
20      numInstances.set(instances.numInstances());
21    }
```

Universidad Veracruzana

Particiones para entrenar y evaluar

```
1   @OPERATION void setTestCV(int numFolds, int numFold){
2       test = instances.testCV(numFolds, numFold);
3     }
4
5   @OPERATION void setTrainCV(int numFolds, int numFold){
6       train = instances.trainCV(numFolds, numFold);
7     }
8
9   @OPERATION void setTrainCV(int numFolds, int numFold, java.util.Random
    ↪  random){
10      train = instances.trainCV(numFolds,numFold,random);
11    }
12
13  @OPERATION void stratify(int numFolds){
14      instances.stratify(numFolds);
15      ObsProperty stratifiedProp = getObsProperty("stratified");
16      stratifiedProp.updateValue(numFolds);
17    }
```

Universidad Veracruzana

Pasando ejemplos a otro artefacto

```
1  @LINK void getTrainData(OpFeedbackParam<Instances> data){
2      data.set(train);
3    }
```

Universidad Veracruzana

Inicialización

```
1  package jacaDDM;
2
3  import cartago.*;
4  import jason.asSyntax.*;
5  import weka.classifiers.trees.J48;
6  import weka.core.*;
7
8  @ARTIFACT_INFO(
9      outports = { @OUTPORT(name="portInstancesBase") }
10 )
11
12 public class J48Artifact extends Artifact {
13 private J48 tree = new J48();
14 private OpFeedbackParam<Instances> data = new
   ↪  OpFeedbackParam<Instances>();
15 Instances trainData;
16 Instances inconsistentTrainData;
17 private int numAttributes;
18 private Attribute[] attributes;
```

Universidad Veracruzana

Obteniendo ejemplos

```
1   @OPERATION void getTrainData(){
2       try {
3         execLinkedOp("portInstancesBase","getTrainData",data);
4         trainData = data.get();
5         numAttributes = trainData.numAttributes();
6         attributes = new Attribute[numAttributes];
7         for(int i=0;i<numAttributes-1;i++){
8           attributes[i] = trainData.attribute(i);
9         }
10        signal("newTrainData");
11      } catch (Exception e) {
12        // TODO Auto-generated catch block
13        e.printStackTrace();
14      }
15    }
```

Universidad Veracruzana

Induciendo una hipótesis

```
1   @OPERATION void induce() {
2       try {
3         tree.buildClassifier(trainData);
4         signal("NewModel");
5       } catch (Exception e) {
6         // TODO Auto-generated catch block
7         e.printStackTrace();
8       }
9     }
10
11  @OPERATION void printModel() {
12      System.out.println(tree);
13    }
```

Universidad Veracruzana

Usando la hipótesis I

```
1  @OPERATION void classify(String instanceStr,OpFeedbackParam<Term>
   ↪  classVal){
2      Instance i = new Instance(numAttributes);
3      String[] vals = instanceStr.split(",");
4
5      int j=0;
6      for(String val:vals){
7        if(attributes[j].isNumeric()){
8          double valDouble = Double.parseDouble(val);
9          i.setValue(attributes[j],valDouble);
10       } else {
11         i.setValue(attributes[j],val);
12       };
13       j++;
14     }
15     try {
16       i.setDataset(trainData);
17       i.setClassValue(tree.classifyInstance(i));
18       Atom classTerm = new Atom(i.stringValue(numAttributes-1));
19       classVal.set(classTerm);
```

Universidad Veracruzana

Usando la hipótesis II

```
20        } catch (Exception e) {
21          // TODO Auto-generated catch block
22          e.printStackTrace();
23        }
24      }
```

Universidad Veracruzana

Beto aprende y clasifica I

```
1  +!start : true <-
2    makeArtifact("timer","jacaDDM.Timer",[],TimerId);
3    .my_name(Me);
4    .concat(Me,"-instancesBase",InstancesBaseName);
5    makeArtifact(InstancesBaseName,"jacaDDM.InstancesBase",[],InstancesBaseId);
6    focus(InstancesBaseId);
7    .concat(Me,"-j48",J48Name);
8    makeArtifact(J48Name,"jacaDDM.J48Artifact",[],J48Id);
9    focus(J48Id);
10   loadARFF("./arff/iris.arff"); // An instance base operation
11   setTrainCV(10,1);
12   linkArtifacts(J48Id, "portInstancesBase", InstancesBaseId);
13   println("Cargando conjunto de entrenamiento...");
14   startTime; // A timer operation to measure loading time
15   getTrainData;
16   endTime(LoadTime);
17   println("Tiempo de carga de datos: ",LoadTime," ms.");
18   println("Construyendo un árbol de decisión por inducción...");
19   startTime; // A timer operation to measure induction time
20   induce;
```

Universidad Veracruzana

Beto aprende y clasifica II

```
21      endTime(InduceTime);
22      println("Tiempo de inducción: ",InduceTime," ms.");
23      println("El árbol inducido es: \n");
24      printModel; // A j48 operation
25      println("A partir del conjunto en entrenamiento: \n");
26      printTrain;
27      I = "5.1,3.5,1.4,0.2";
28      classify(I,C);
29      println("La clase de ", I," es : ",C,".").
```

Universidad Veracruzana

Cronómetro

```
1  package jacaDDM;
2
3  import cartago.*;
4
5  public class Timer extends Artifact {
6    private long startTime;
7    private long endTime;
8
9  @OPERATION
10     void startTime(){
11      startTime= System.nanoTime();
12    }
13
14  @OPERATION
15     void endTime(OpFeedbackParam<java.lang.Long> totalTime){
16     endTime = System.nanoTime();
17     totalTime.set((endTime - startTime)/1000000); // milliseconds
18    }
19
20 }
```

Universidad Veracruzana

Referencias I

[1] HG Acosta-Mesa, N Cruz-Ramírez y R Hernández-Jiménez. "Aceto-white temporal pattern classification using k-NN to identify precancerous cervical lesion in colposcopic images". En: *Computers in biology and medicine* 39.9 (2009), págs. 778-784.

[2] KA Albashiri y F Coenen. "Agent-enriched data mining using an extendable framework". En: *Agents and Data Mining Interaction*. Springer, 2009, págs. 53-68.

[3] S Bailey et al. "Papyrus: a system for data mining over local and wide area clusters and super-clusters". En: *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 1999, pág. 63.

[4] A Bifet et al. "MOA: Massive online analysis". En: *The Journal of Machine Learning Research* 11 (2010), págs. 1601-1604.

[5] JC Díaz-Preciado, A Guerra-Hernández y N Cruz-Ramí'rez. "Un modelo de red bayesiana de la informalidad laboral en Veracruz orientado a una simulación social basada en agentes". En: *Research in Computing Science* 113.2016 (2016), págs. 157-170. issn: 1870-4069.

[6] U Fayyad, G Piatetsky-Shapiro y P Smyth. "From Data Mining to Knowledge Discovery in Databases". En: *AI Magazine* 17.3 (1996), págs. 37-54.

Universidad Veracruzana

Referencias II

[7] H Kargupta, DH Byung-Hoon y E Johnson. "Collective data mining: A new perspective toward distributed data analysis". En: *Advances in Distributed and Parallel Knowledge Discovery*. Citeseer. 1999.

[8] X Limón et al. "An Agents & Artifacts approach to Distributed Data Mining". En: *MICAI 2013: Eleventh Mexican International Conference on Artificial Intelligence*. Ed. por F Castro, A Gelbukh y MG Mendoza. Vol. 8266. Lecture Notes in Artificial Intelligence. Berlin, Germany: Springer Verlag, 2013, págs. 338-349.

[9] X Limón et al. "A windowing based GPU optimized strategy for the induction of Decision Trees". En: *Arfificial Intelligence Research and Development*. Ed. por E Armengol, D Boixader y F Grimaldo. Vol. 277. Frontiers in Artificial Intelligence and Applications. Amsterdam, NL: IOS Press, 2015, págs. 100-109.

[10] X Limón et al. "Modeling and implementing distributed data mining strategies in JaCa-DDM". En: *Knowledge and Information Systems* 60.1 (2019), págs. 99-143. issn: 0219-1377.

[11] C Moemeng et al. "i-Analyst: An agent-based distributed data mining platform". En: *2010 IEEE International Conference on Data Mining Workshops*. New York, NY, USA: IEEE, 2010, págs. 1404-1406.

Universidad Veracruzana

Referencias III

[12] JR Quinlan. *C4. 5: programs for machine learning*. Vol. 1. San Mateo, CA, USA: Morgan kaufmann, 1993.

[13] J Schindelin et al. "Fiji: an open-source platform for biological-image analysis". En: *Nature methods* 9.7 (2012), págs. 676-682.

[14] SJ Stolfo et al. "JAM: Java Agents for Meta-Learning over Distributed Databases.". En: *KDD*. Vol. 97. 1997, págs. 74-81.

[15] IH Witten, E Frank y MA Hall. *Data Mining: Practical Machine Learning Toools and Techniques*. Burlington, MA, USA: Morgan Kaufmann Publishers, 2011.

[16] J Xu et al. "Sampling based multi-agent joint learning for association rule mining". En: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents y Multiagent Systems. 2014, págs. 1469-1470.

[17] N Zhong et al. "Framework of a Multi-agent KDD System.". En: *IDEAL 2002*. Ed. por H Yin. Vol. 2412. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2002, págs. 337-346.

Universidad Veracruzana