WeBank

FATE-12月圆桌会议

马国强
微众银行高级研究员

WeBank

01 FATE-1.6纵向算法优化

基于隐私保护的样本id匹配

H(x): x的哈希

Party A

X_A: {u1, u2, u3, u4}

public key: (n, e)

Party B

X_B: {u1, u2, u3, u5}

RSA : n, e, d

$Y_A = \{ri^e \%n * \mathrm{H(ui)} \%n \mid \mathrm{ui} \in X_A \; ri: rand\}$

Y_A

$Z_A = (ri^e \mathrm{H}(ui))^d \%n = ri * (\mathrm{H}(ui))^d \%\mathrm{n} \mid ri^e \mathrm{H}(ui) \in Y_A\}$

$D_A =$
$\{\mathrm{H(ri*(H}(ui))^d \mathrm{/ri\%n)} = \mathrm{H}((\mathrm{H}(ui))^d) \mid$
$\mathrm{ri*(H}(ui))^d \in Z_A\}$

Z_A, Z_B

$Z_B = \{ \mathrm{H}((\mathrm{H(u)})^d \%n) \mid \mathrm{u} \in X_B\}$

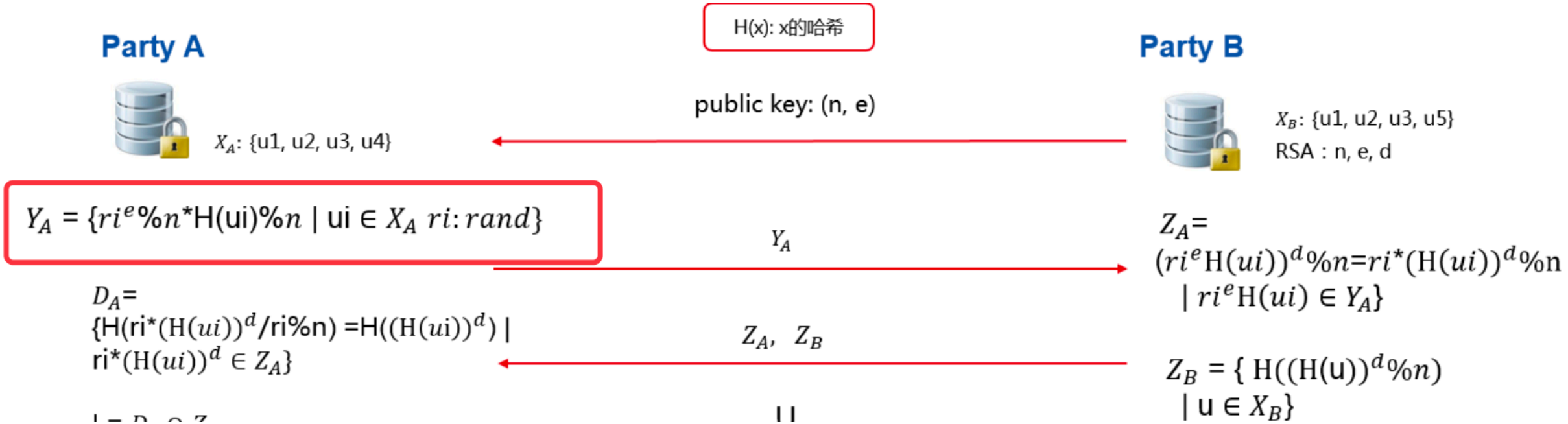
U

$\mathrm{I} = D_A \cap Z_B$
$= \{\mathrm{H}((\mathrm{H}(u1))^d), \mathrm{H}((\mathrm{H}(u2))^d), \mathrm{H}((\mathrm{H}(u3))^d)\}$

I, $\mathrm{D_A}$ => U = {u1, u2, u3}

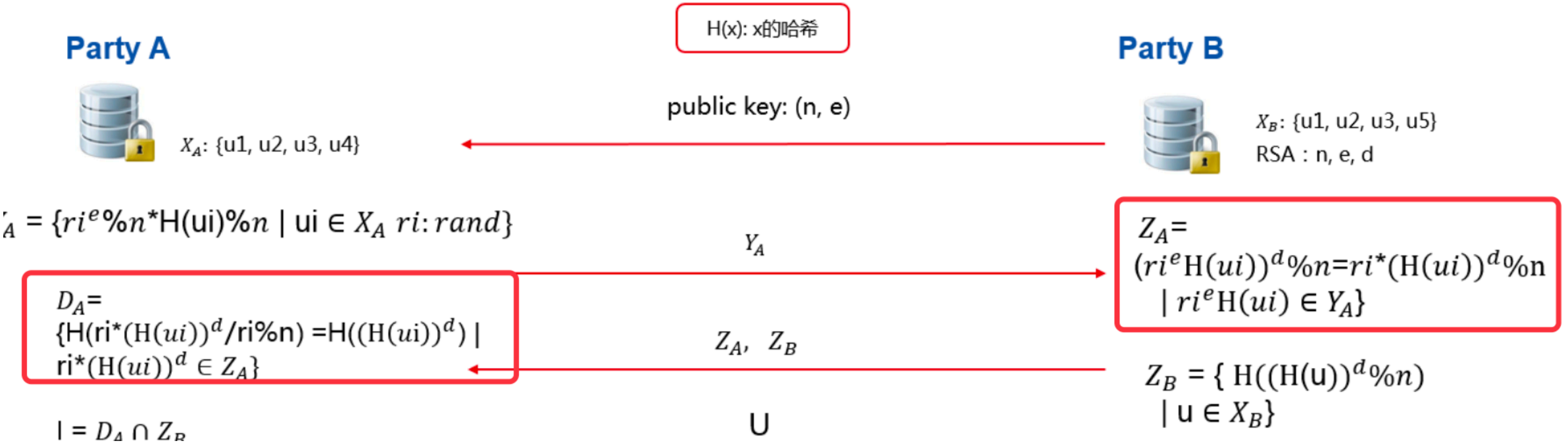
U = {u1, u2, u3}

基于隐私保护的样本id匹配



- 性能优化1:
  - ri^e: YA计算主要耗时为系统随机数生成和幂模运算，预先生成特定比例的r^e，样本随机复用

基于隐私保护的样本id匹配



- 性能优化1:
  - 算力和传输对等性：A方只需要计算YA和ZA / r = ZA * r，B方需要计算两次幂运算，耗时非常大; 同理B方数据传输为两次，接收为1次，A则相反，可通过数据分组，如对H(u) % 2进行分组，两方使用相应数据进行正反RSA交集运算，使得双方计算和传输对等

联邦特征工程

Encry(x): x的加法同态加密,
Encode(x): 本地编码

A: Xi (特征分箱)

id_set_1 (eg: id1, id2, id5,..)
id_set_2
****
id_set_i
****
id_set_k

1. {idi , Encry(yi), Encry(1-yi)}

2. {
Encode(id_set_i),
sum(Encry(yi)),
sum(Encry(1-yi))
}

B: (X,Y)

id1	y1
id2	y2
***	***
idi	yi
***	****
idn	yn

3.
npos_i = Decry(sum(Encry(yi)));
nneg_i = Decry(sum(Encry(1-yi)))

B 方本地计算

1. distpos_i = npos_i/pos_total; distneg_i =nneg_i/neg_total
2. Woe_i = 100 * log(distpos_i / distneg_i)
3. IV = $\sum_{i=1}^{k} (dispos\_i - disneg\_i) * \log(dispos\_i/disneg\_i)$

联邦特征工程

- 性能优化
  - A计算每个bin的sum([[yi]])和sum([1-yi]]），sum([[yi]]) + sum([[1-yi]])=n为落在bin的样本数，可只计算sum([[yi]])，n - sum([[yi]])即为另一半运算结果
  - 最频繁\稀疏bin优化：对于互联网特别广告数据而言，稀疏数据为主=》可以计算非稀疏\最频繁bin的sum([[yi]])和sum([[1-yi]])，样本总数减去非稀疏\非最频繁的bin
  - 直方图压缩传输和解密：同态加密一般用1024 bit或者更高，而bin内的[[yi]]或者[[1-yi]]比较小，多个bin的密文进行压缩和传输，传输到guest后再反解压，可降低传输和解密次数，如32个bin压缩为((([[Bin_sum1]] * 2^32+[[Bin_sum2]]) * 2^32 + [[Bin_sum3]]) * 2^32 + Bin_sum4 …
  - GK-Summary Object较大，维护耗时，减少拷贝

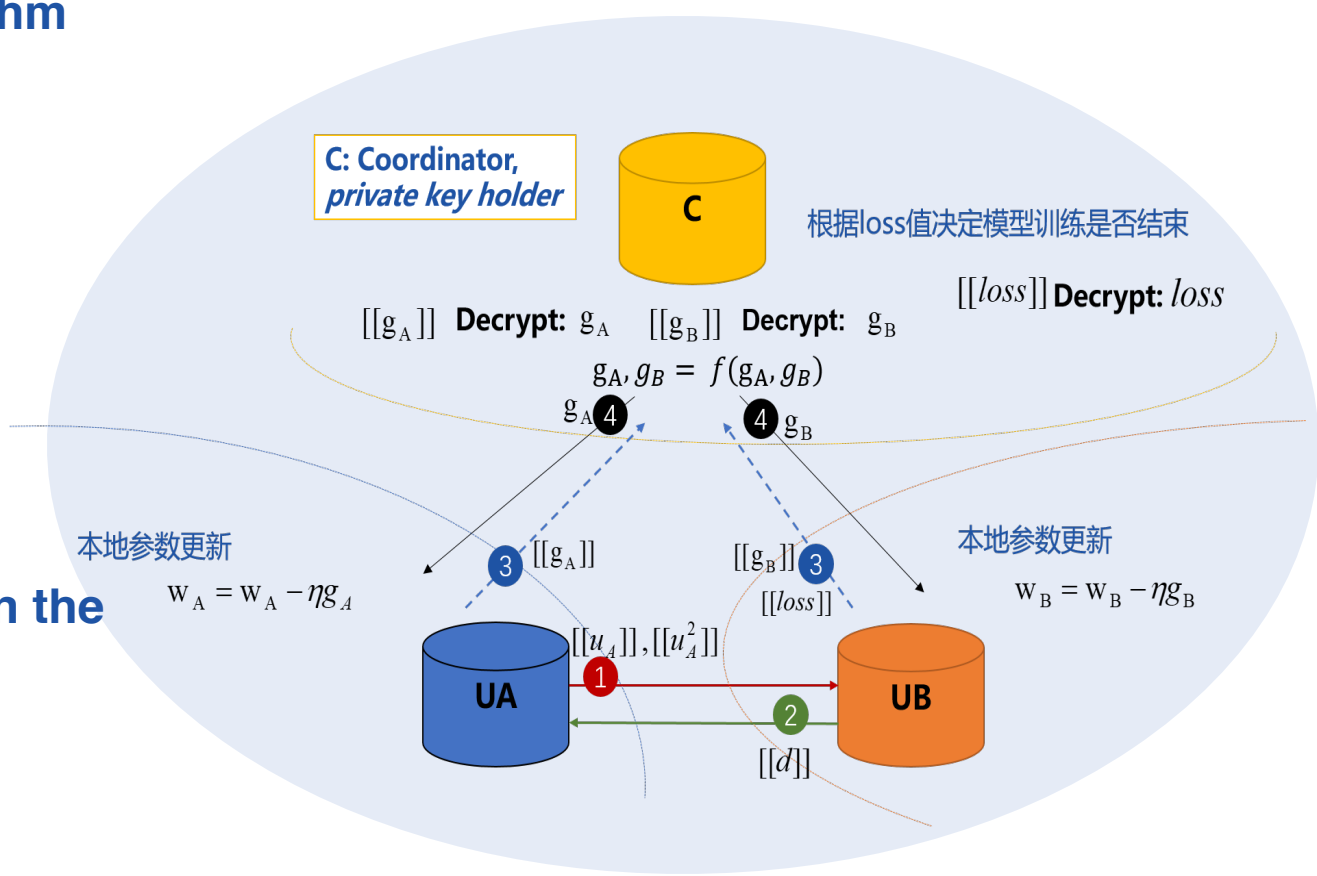
纵向Federated Logistic Regression

多项式近似 Polynomial approximation for logarithm function

$$l(w) = \log(1 + \exp(-yw^T x))$$
$$\approx \log 2 - \frac{1}{2} yw^T x + \frac{1}{8}(w^T x)^2$$
$$\nabla l(w) = (\frac{1}{1 + \exp(-yw^T x)} - 1)yx$$
$$\approx (\frac{1}{2} yw^T x - 1)\frac{1}{2} yx$$

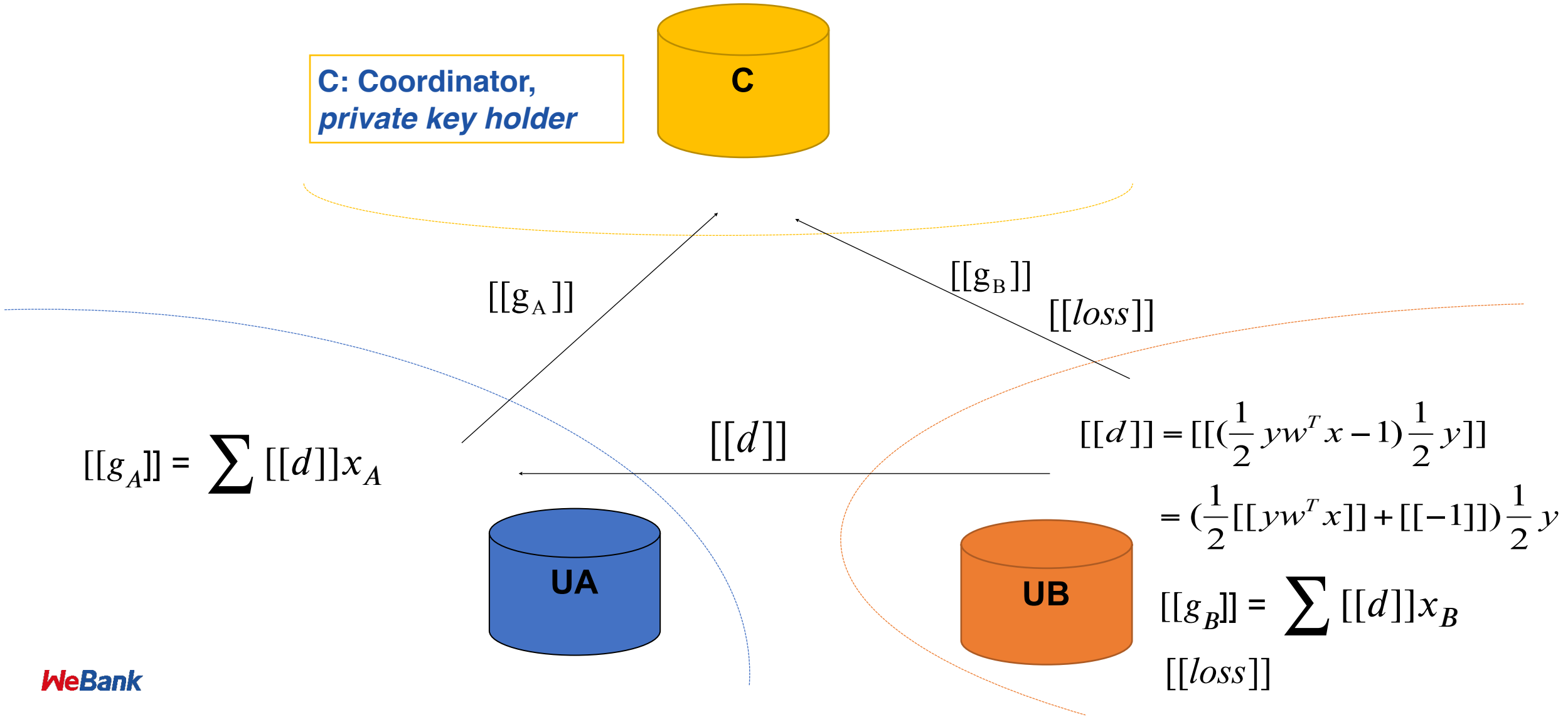
加密计算 Encrypted computation for each term in the polynomial function

$$loss = \log 2 - \frac{1}{2} yw^T x + \frac{1}{8}(w^T x)^2$$
$$[[loss]] = [[\log 2]] + (-\frac{1}{2}) * [[yw^T x]] + \frac{1}{8}[[(w^T x)^2]]$$



- Kim, M.; Song, Y.; Wang, S.; Xia, Y.; and Jiang, X. 2018. Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Med Inform 6(2)
- Y. Aono, T. Hayashi, T. P. Le, L. Wang, Scalable and secure logistic regression via homomorphic encryption, CODASPY16

纵向Federated Logistic Regression



纵向Federated Logistic Regression

$$[[d]] = ([[\frac{1}{2}]] y w^T x + [[-1]]) \frac{1}{2} y$$

$$= \frac{1}{4} w^T x - \frac{1}{2} y = (\frac{1}{4} w_A^T x_A) + (\frac{1}{4} w_B^T x_B - \frac{1}{2} y)$$

- 性能优化
  - [[d]]计算并行，图中第一、二步可合并

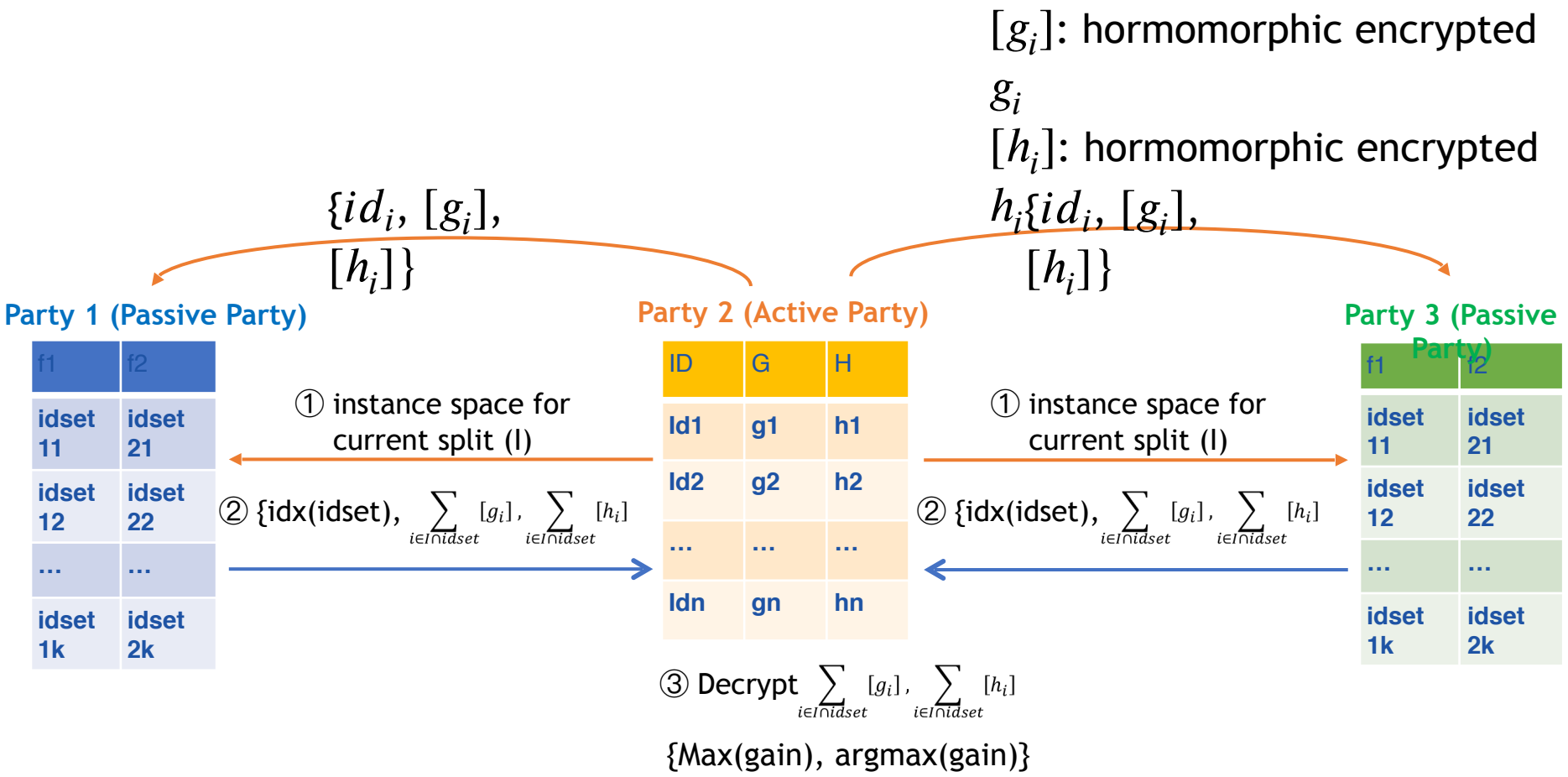
纵向Federated Logistic Regression

```
def __mul__(self, scalar):
    """return Multiply by an scalar(such as int, float)
    """

    encode = FixedPointNumber.encode(scalar, self.public_key.n, self.public_key.max_int)
    plaintext = encode.encoding

    if plaintext < 0 or plaintext >= self.public_key.n:
        raise ValueError("Scalar out of bounds: %i" % plaintext)

    if plaintext >= self.public_key.n - self.public_key.max_int:
        # Very large plaintext, play a sneaky trick using inverses
        neg_c = gmpy_math.invert(self.ciphertext(False), self.public_key.nsquare)
        neg_scalar = self.public_key.n - plaintext
        ciphertext = gmpy_math.powmod(neg_c, neg_scalar, self.public_key.nsquare)
    else:
        ciphertext = gmpy_math.powmod(self.ciphertext(False), plaintext, self.public_key.ns

    exponent = self.exponent + encode.exponent
```

- 性能优化
  - [[d]] * X计算优化，使用低精度，如X = int(round(X * 2**23))

```
@classmethod
def encode(cls, scalar, n=None, max_int=None, precision=None, max_exponent=None):
    """return an encoding of an int or float.
    """
    # Calculate the maximum exponent for desired precision
    exponent = None

    #  Too low value preprocess;
    #  avoid "OverflowError: int too large to convert to float"

    if np.abs(scalar) < 1e-200:
        scalar = 0

    if n is None:
        n = cls.Q
        max_int = cls.Q // 3 - 1

    if precision is None:
        if isinstance(scalar, int) or isinstance(scalar, np.int16) or \
                isinstance(scalar, np.int32) or isinstance(scalar, np.int64):
            exponent = 0
        elif isinstance(scalar, float) or isinstance(scalar, np.float16) \
                or isinstance(scalar, np.float32) or isinstance(scalar, np.float64):
            flt_exponent = math.frexp(scalar)[1]
            lsb_exponent = cls.FLOAT_MANTISSA_BITS - flt_exponent
            exponent = math.floor(lsb_exponent / cls.LOG2_BASE)
        else:
            raise TypeError("Don't know the precision of type %s."
                            % type(scalar))
    else:
        exponent = math.floor(math.log(precision, cls.BASE))

    if max_exponent is not None:
        exponent = max(max_exponent, exponent)

    int_fixpoint = int(round(scalar * pow(cls.BASE, exponent)))

    if abs(int_fixpoint) > max_int:
        raise ValueError('Integer needs to be within +/- %d but got %d'
                         % (max_int, int_fixpoint))

    return cls(int_fixpoint % n, exponent, n, max_int)
```

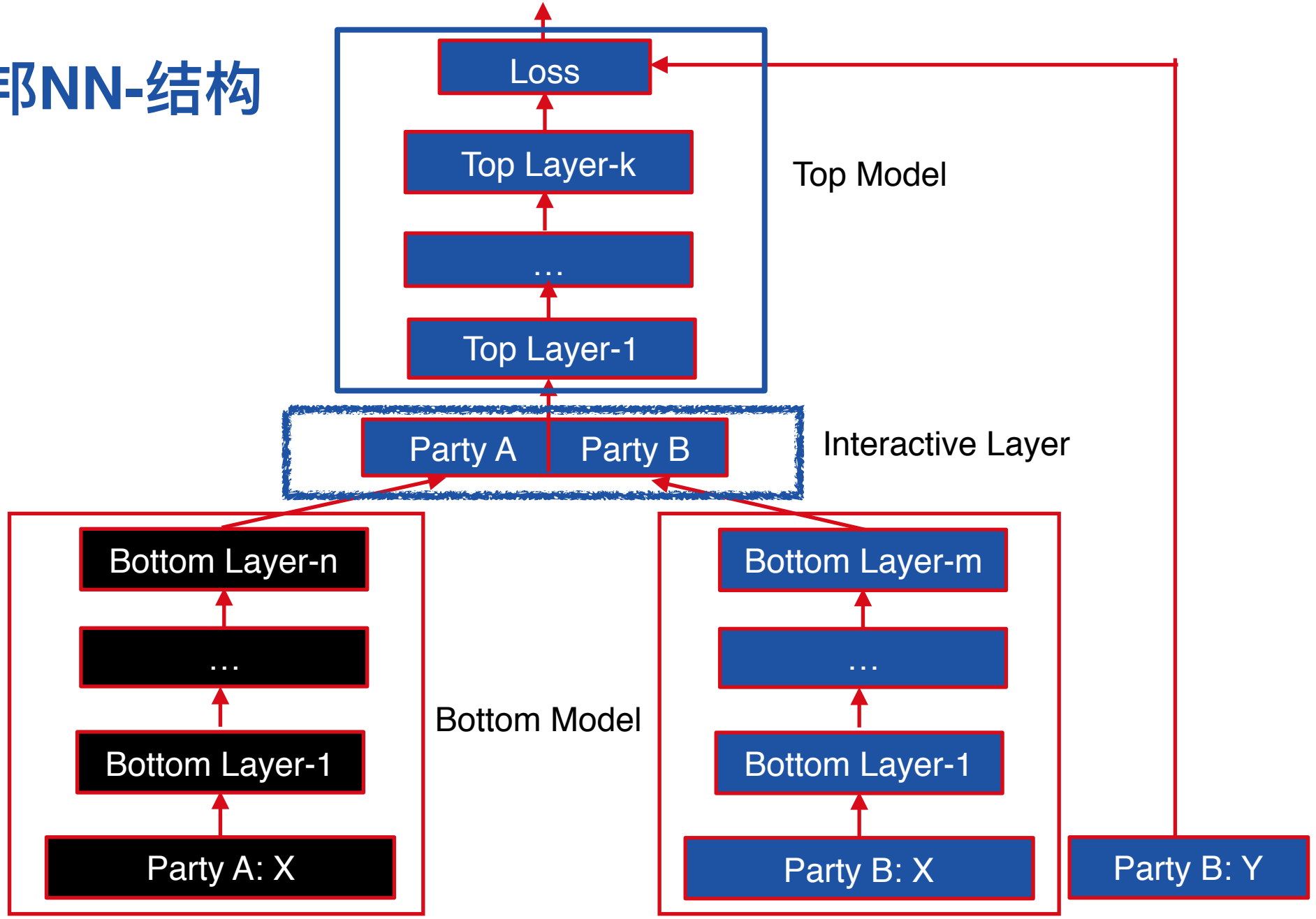
Secureboost 联邦计算信息增益



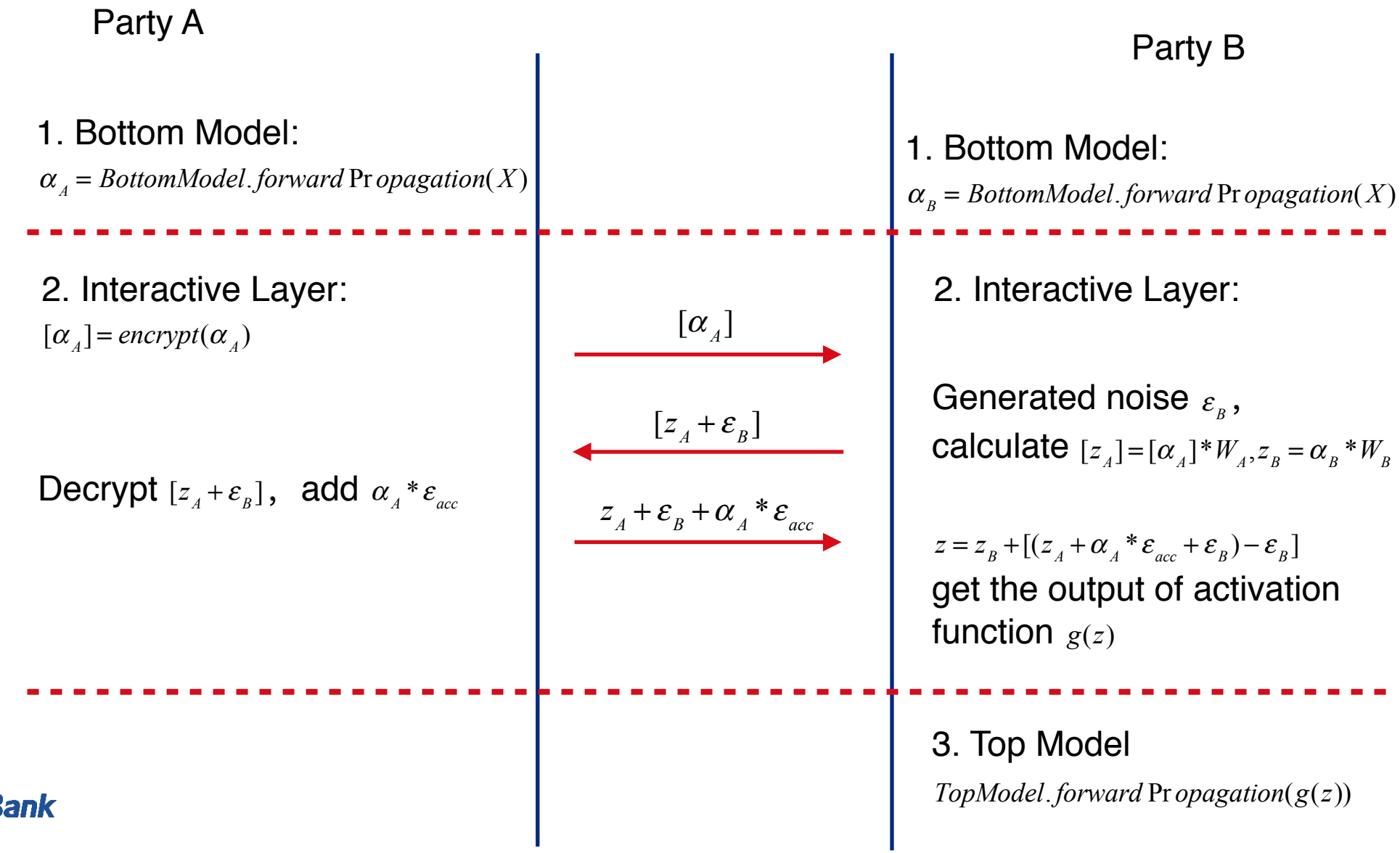
Secureboost

- 性能优化：
  - [[左儿子梯度直方图]] + [[右儿子梯度直方图]] = [[父亲节点梯度直方图]]，每次统计直方图时，只需统计儿子样本数少的一方，另一方可直接得到
  - GOSS: 梯度更大的样本代表欠拟合，优先训练
  - 直方图压缩传输和解密：类似binning，加密直方图压缩传输
  - 频繁桶优化

纵向联邦NN-结构


Loss
Top Layer-k
…
Top Layer-1
Top Model
Party A
Party B
Interactive Layer
Bottom Layer-n
…
Bottom Layer-1
Party A: X
Bottom Model
Bottom Layer-m
…
Bottom Layer-1
Party B: X
Party B: Y

纵向联邦NN-前向传播


Party A
Party B
1. Bottom Model:
$\alpha_A = BottomModel.forward\,\mathrm{Pr}opagation(X)$
1. Bottom Model:
$\alpha_B = BottomModel.forward\,\mathrm{Pr}opagation(X)$
2. Interactive Layer:
$[\alpha_A] = encrypt(\alpha_A)$
$[\alpha_A]$
2. Interactive Layer:
Generated noise ε_B,
calculate $[z_A] = [\alpha_A] * W_A, z_B = \alpha_B * W_B$
$[z_A + \varepsilon_B]$
Decrypt $[z_A + \varepsilon_B]$, add $\alpha_A * \varepsilon_{acc}$
$z_A + \varepsilon_B + \alpha_A * \varepsilon_{acc}$
$z = z_B + [(z_A + \alpha_A * \varepsilon_{acc} + \varepsilon_B) - \varepsilon_B]$
get the output of activation function $g(z)$
3. Top Model
$TopModel.forward\,\mathrm{Pr}opagation(g(z))$

纵向联邦NN-后向传播

Party A

Party B

1. Top Model:

Use label Y, forward output to generate interactive layer's Output error δ, and update top model

2. Interactive Layer:

Get error of activation-fun δ_{act},

calculate $\delta_{act} * \alpha_B$ to update W_B,

gen bottom model's output error δ_{bottom_B}

gen noise ε_B and

calculate $\delta_{act} * [\alpha_A] + \varepsilon_B = [\delta_{act} * \alpha_A + \varepsilon_B]$

$[\delta_{act} * \alpha_A + \varepsilon_B]$

$\delta_{act} * \alpha_A + \varepsilon_B + \frac{\varepsilon_A}{\eta}, [\varepsilon_{acc}]$

$[\delta_{bottom\_A}]$

2. Interactive Layer:

Encrypt ε_{acc}, gen ε_A

decrypt $[\delta_{act} * \alpha_A + \varepsilon_B]$,

calculate $\delta_{act} * \alpha_A + \varepsilon_B + \frac{\varepsilon_A}{\eta}$

$\varepsilon_{acc} += \varepsilon_A$

Decrypt $[\delta_{bottom\_A}]$

$[\delta_{bottom\_A}] = \delta_{act} * (W_A + [\varepsilon_{acc}]) = [\delta_{act} * (W_A + \varepsilon_{acc})]$

$W_A -= \eta(\delta_{act} * \alpha_A + \varepsilon_B + \frac{\varepsilon_A}{\eta} - \varepsilon_B)$

$\Rightarrow W_A = W_A^{True} - \varepsilon_{acc}$

3. Bottom Model

Use δ_{bottom_A} to update bottom model

3. Bottom Model

Use δ_{bottom_B} to update bottom model

WeBank

纵向联邦NN

- 性能优化：
  - SelectiveBP: 损失大样本优先训练，优化后向传播
  - 前向W * [[alpha]]， 后向delta * [[alpha]]低精度优化同态加密数乘和加法运算

WeBank

02 FATE-Client介绍

FATE Client

- PipeLine：类keras的用户建模编程界面，后端无缝对接tensorflow、keras、pytorch, 快速搭建联邦建模流程
- 全新FLOW CLI v2，独立安装、语法简洁、智能提示
- 提供FLOW Python语言SDK
- 支持PyPI

https://github.com/FederatedAI/FATE/tree/master/python/fate_client

CLI V2

- Enable
  - pip install fate-client (pip install -e python/fate_client
  - 初始化：flow init —ip 127.0.0.1 —port 9380 (ip: flow服务端所在的主机地址，port: flow服务端口)
    - flow -h

```
(venv) [redacted] fate]$ flow -h
Usage: flow [OPTIONS] COMMAND [ARGS]...

  Fate Flow Client

Options:
  -h, --help  Show this message and exit.

Commands:
  component  Component Operations
  data       Data Operations
  init       Flow CLI Init Command
  job        Job Operations
  model      Model Operations
  queue      Queue Operations
  table      Table Operations
  tag        Tag Operations
  task       Task Operations
```

```
(venv) [redacted] fate]$ flow job
Usage: flow job [OPTIONS] COMMAND [ARGS]...

  Provides numbers of job operational commands, including submit, stop, query and etc.
  For more details, please check out the help text.

Options:
  -h, --help  Show this message and exit.

Commands:
  config  Config Job Command
  dsl     Generate Predict DSL Command
  list    List Job Command
  log     Log Job Command
  query   Query Job Command
  stop    Stop Job Command
  submit  Submit Job Command
  view    Query Job Data View Command
```

PipeLine

```
pipeline = PipeLine() \
               .set_initiator(role='guest', party_id=guest) \
               .set_roles(guest=guest, host=host, arbiter=arbiter)
pipeline.add_component(reader_0)
pipeline.add_component(dataio_0, data=Data(data=reader_0.output.data))
pipeline.add_component(intersection_0, data=Data(data=dataio_0.output.data))
pipeline.add_component(hetero_poisson_0, data=Data(train_data=intersection_0.output.data))
pipeline.add_component(evaluation_0, data=Data(data=hetero_poisson_0.output.data))
pipeline.compile()
pipeline.fit(backend=Backend.EGGROLL, work_mode=WorkMode.CLUSTER)
```

- 类keras的用户建模编程界面，接口简单
- 与旧的通过配置json DSL方式比，对开发者友好，搭建LR等经典应用核心代码十行内

PipeLine

- PipeLine Enable
  - pip install fate-client (pip install -e python/fate_client
  - 初始化：pipeline init —ip 127.0.0.1 —port 9380 (ip: flow服务端所在的主机地址，port: flow服务端口)
    - pipeline init —help 可以打印指令详情介绍

PipeLine

- Components Pipeline Support:
  - Reference: https://github.com/FederatedAI/FATE/blob/master/python/fate_client/pipeline/component/README.rst#component-list
  - Or: python/fate_client/pipeline/component

PipeLine

```
from pipeline.component import DataIO
from pipeline.component import HeteroPoisson
from pipeline.component import Intersection
from pipeline.component import Reader
```

- Step1: import using component
- Components Support:
  - Reference: https://github.com/FederatedAI/FATE/blob/master/python/fate_client/pipeline/component/README.rst#component-list
  - Or: python/fate_client/pipeline/component

PipeLine

```
dataio_0 = DataIO(name="dataio_0", data_type='float', output_format="dense")
dataio_0.get_party_instance(role='guest', party_id=9999).component_param(with_label=True,
                                                                         label_name="doctorco",
                                                                         label_type="float")
dataio_0.get_party_instance(role='host', party_id=10000).component_param(with_label=False)
```

- Step2: Init Component Setting
- Param Support: python/federatedml/param or python/fate_client/pipeline/param
- Component init:
  - name: 标记这个组件的别名，如不设置，则由系统自动分配
  - Object(param1=, param2=, param3….)：此类参数代表所有的参与方使用同样的参数配置
  - get_party_instance(role, party_id=int or list).component_param，表示单独设置某个参与方的参数

PipeLine

```
from pipeline.backend.pipeline import PipeLine
```

```
pipeline = PipeLine().set_initiator(role='guest', party_id=9999). \
    set_roles(guest=9999, host=10000, arbiter=10001)
```

- Step3: 设置PipeLine驱动
- set_initator(role, party_id): 设置任务发起方
- set_roles: 设置任务参与方及partyid

PipeLine

```
from pipeline.interface import Data
from pipeline.interface import Model
```

```
pipeline.add_component(reader_0)
pipeline.add_component(reader_1)
pipeline.add_component(dataio_0, data=Data(data=reader_0.output.data))
pipeline.add_component(dataio_1,
                       data=Data(data=reader_1.output.data),
                       model=Model(dataio_0.output.model))
pipeline.add_component(intersection_0, data=Data(data=dataio_0.output.data))
pipeline.add_component(intersect_1, data=Data(data=dataio_1.output.data))
pipeline.add_component(hetero_poisson_0, data=Data(train_data=intersection_0.output.data,
                                                   validate_data=intersect_1.output.data))
```

- Step4: 构建运行的DAG图
- DAG图通过Data和Model进行关联，其中Data表示输入数据依赖，Model表示模型依赖
- Pipeline().add_component(component, data=Data(), model=Model()): 表示增加组件，数据依赖关系、模型依赖关系

PipeLine

```
from pipeline.runtime.entity import JobParameters
```

```
pipeline.compile()
job_parameters = JobParameters(backend=0, work_mode=1)
pipeline.fit(job_parameters)
```

- Step5：编译&运行
  - Pipeline.compile(): 调用pipeline编译接口，构造训练的dsl以及构造任务运行时的参数配置
  - JobParameters: 设置运行时的环境参数，包括后端执行运行、执行方式（单机、并行）、并发度等
  - pipeline.fit(): 执行训练流程

PipeLine

```
(venv) [a[redacted]fate]$ python pipeline-hetero-poisson-validate.py
2020-12-23 12:00:31.785 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:121 - Job id is 202012231200310995902 9
Job is still waiting, time elapse: 0:00:01
Running component reader_1, time elapse: 0:00:17
Running component reader_0, time elapse: 0:00:35
Running component dataio_0, time elapse: 0:00:48
Running component intersection_0, time elapse: 0:01:04
Running component dataio_1, time elapse: 0:01:17
Running component intersection_1, time elapse: 0:01:33
Running component hetero_poisson_0, time elapse: 0:02:10
2020-12-23 12:02:46.364 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:129 - Job is success!!! Job id is 20201223120031099
59029
2020-12-23 12:02:46.364 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:130 - Total time: 0:02:14
```

PipeLine

```
summary = pipeline.get_component("hetero_poisson_0").get_summary()
print(summary)
model_param = pipeline.get_component("hetero_poisson_0").get_model_param()
print(model_param)
```


{'best_iteration': -1, 'coef': {'chcond1': -0.0499457442441531, 'chcond2': -0.04774131444036287, 'hscore': -0.05004757054342704}, 'intercept': -0.12006708385233443, 'is_converged': False}
{'bestIteration': -1, 'header': ['hscore', 'chcond1', 'chcond2'], 'intercept': -0.12006708385233443, 'isConverged': False, 'iters': 2, 'lossHistory': [], 'weight': {'chcond1': -0.0499457442441531, 'chcond2': -0.04774131444036287, 'hscore': -0.05004757054342704}}

- Step6: 获取训练相关信息
  - pipeline.get_component(component_name): 获取某个组件的任务信息task_info
  - task_info.get_summary(): 获取组件的汇总信息
  - task_info.get_model_param(): 获取组件的模型输出
  - task_info.get_output_data(): 获取组件的数据输出

PipeLine

```
pipeline.deploy_component(["dataio_0", "intersection_0", "hetero_poisson_0"])
pipeline.dump("pipeline_saved.pkl")
```

- Step7: 模型部署和保存
  - pipeline.deploy_component(component_list): 通过该接口指定参与离线预测的组件
  - pipeline.dump(pipeline_model_path): 将训练的pipeline保存到指定的模型文件

PipeLine

```
train_pipeline = PipeLine.load_model_from_file("pipeline_saved.pkl")
train_pipeline.describe()
```

```
2020-12-23 13:06:36.846 | INFO     | pipeline.backend.pipeline:describe:314 - Pipeline Stage is fit
2020-12-23 13:06:36.846 | INFO     | pipeline.backend.pipeline:describe:315 - DSL is:
2020-12-23 13:06:36.846 | INFO     | pipeline.backend.pipeline:describe:317 - {'components': {'reader_0': {'module': 'Reader', 'output': {'data'
: ['data']}}, 'reader_1': {'module': 'Reader', 'output': {'data': ['data']}}, 'dataio_0': {'module': 'DataIO', 'input': {'data': {'data': ['read
er_0.data']}}, 'output': {'data': ['data'], 'model': ['model']}}, 'dataio_1': {'module': 'DataIO', 'input': {'data': {'data': ['reader_1.data']}
, 'model': ['dataio_0.model']}, 'output': {'data': ['data'], 'model': ['model']}}, 'intersection_0': {'module': 'Intersection', 'input': {'data'
: {'data': ['dataio_0.data']}}, 'output': {'data': ['data']}}, 'intersection_1': {'module': 'Intersection', 'input': {'data': {'data': ['dataio_
1.data']}}, 'output': {'data': ['data']}}, 'hetero_poisson_0': {'module': 'HeteroPoisson', 'input': {'data': {'train_data': ['intersection_0.dat
a'], 'validate_data': ['intersection_1.data']}}, 'output': {'data': ['data'], 'model': ['model']}}}}
2020-12-23 13:06:36.847 | INFO     | pipeline.backend.pipeline:describe:321 - Pipeline Create Time: Wed Dec 23 13:01:11 2020
```

- Step8: 训练模型的恢复和查看
  - Pipeline.load_model_from_file(restore_model_path): 恢复保存的pipeline
  - pipeline.describe(): 获取pipeline的相关描述，包括阶段、创建时间、构造的dsl等

PipeLine

```
predict_pipeline = PipeLine()
predict_pipeline.add_component(reader_predict)
predict_pipeline.add_component(train_pipeline,
                               data=Data(predict_input={train_pipeline.dataio_0.input.data:
                                                        reader_predict.output.data}))
predict_pipeline.add_component(evaluation_0, data=Data(data=train_pipeline.hetero_poisson_0.output.data))
predict_pipeline.predict(JobParameters(backend=0, work_mode=1))
```

```
2020-12-23 14:28:07.136 | INFO     | pipeline.backend.pipeline:describe:321 - Pipeline Create Time: Wed Dec 23 13:01:11 2020
2020-12-23 14:28:07.777 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:121 - Job id is 202012231428071449 0836
Job is still waiting, time elapse: 0:00:07
Running component reader_predict, time elapse: 0:00:24
Running component dataio_0, time elapse: 0:00:39
Running component intersection_0, time elapse: 0:00:54
Running component hetero_poisson_0, time elapse: 0:01:12
Running component evaluation_0, time elapse: 0:01:28
2020-12-23 14:29:41.401 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:129 - Job is success!!! Job id is 20201223142807144
90836
2020-12-23 14:29:41.401 | INFO     | pipeline.utils.invoker.job_submitter:monitor_job_status:130 - Total time: 0:01:33
```

- Step9: 模型预测
  - add_component: 支持将训练pipeline部署后的模型作为一个整体进行增加
    - pipeline作为整体输入时，使用predict_input=dict来进行数据传递（可能有多组件需要数据）
    - 支持在训练部署模型前后增加新组件
  - pipeline.predict(): 执行预测流程

WeBank

03 DSL V2使用介绍

DSL V2

- 增加多项语法异常检测，V1兼容
- 优化提交任务配置格式
- 支持按需生成预测配置
- 预测阶段可实时编辑增加新组件

DSL V2

- submit_conf需要添加 “dsl_version”: 2
- “job_parameters”: 运行时的任务参数变量
  - “common”: 表示每一方都用同样的任务参数设置
  - “role”: 指定某一方特定的运行参数
    - 图例为 { “role”: {“arbiter”: {“0”: {}}}}，arbiter: “0
      代表arbiter方的第0个party

```
"dsl_version": 2,
"job_parameters": {
    "common": {
        "job_type": "train",
        "backend": 0,
        "work_mode": 0
    },
    "role": {
        "arbiter": {
            "0": {
                "eggroll_run": {
                    "eggroll.session.processors.per.node": 1
                }
            }
        }
    }
},
```

DSL V2

- “component_parameters”: 运行时组件的参数配置
  - “common”: 配置在下面的组件，代表所有方都是用该参数设置
  - “role”: 指定某一方特定的组件运行参数
    - 图例 { “role”: {“host”: {“0”: {}}}}，host: “0” 代表host方的第0个party
    - 图例 { “role”: {“guest”: {“0”: {}}}}，guest: “0” 代表guest方的第0个party

```
"component_parameters": {
    "common": {
        "hetero_secure_boost_0": {
            "task_type": "classification",
            "num_trees": 3
        }
    },
    "role": {
        "guest": {
            "0": {
                "reader_0": {
                    "table": {
                        "name": "breast_hetero_guest",
                        "namespace": "experiment"
                    }
                },
                "dataio_0": {
                    "with_label": true
                }
            }
        },
        "host": {
            "0": {
                "reader_0": {
                    "table": {
                        "name": "breast_hetero_host",
                        "namespace": "experiment"
                    }
                },
                "dataio_0": {
                    "with_label": false
                }
            }
        }
    }
}
```

DSL V2

- 支持训练结束后按需多次预测DSL
  - flow job dsl —train-dsl-path $job_dsl —cpn-list “dataio_0, intersection_0, hetero_lr_0” —version 2 -o ./
- 支持预测阶段实时增加DSL
- 任务提交方式flow job submit -c $conf -d $dsl

```
"evaluation_0": {
    "module": "Evaluation",
    "input": {
        "data": {
            "data": [
                "hetero_lr_0.data"
            ]
        }
    },
    "output": {
        "data": [
            "data"
        ]
    }
}
```

关注FATE

FATE GitHub主页

欢迎来GitHub 加入FATE建设
star我们，第一时间接收项目进展
官网：**https://www.fedai.org/**
邮箱：**contact@fedai.org**

FATE 微信小助手

国内首个联邦学习官方社区，这里有
- 高价值贡献者激励计划
- 10+顶尖算法工程师实时答疑解惑
- 超300家企业机构开发者共同交流学习
- 国内最新联邦学习产品资讯抢先获取