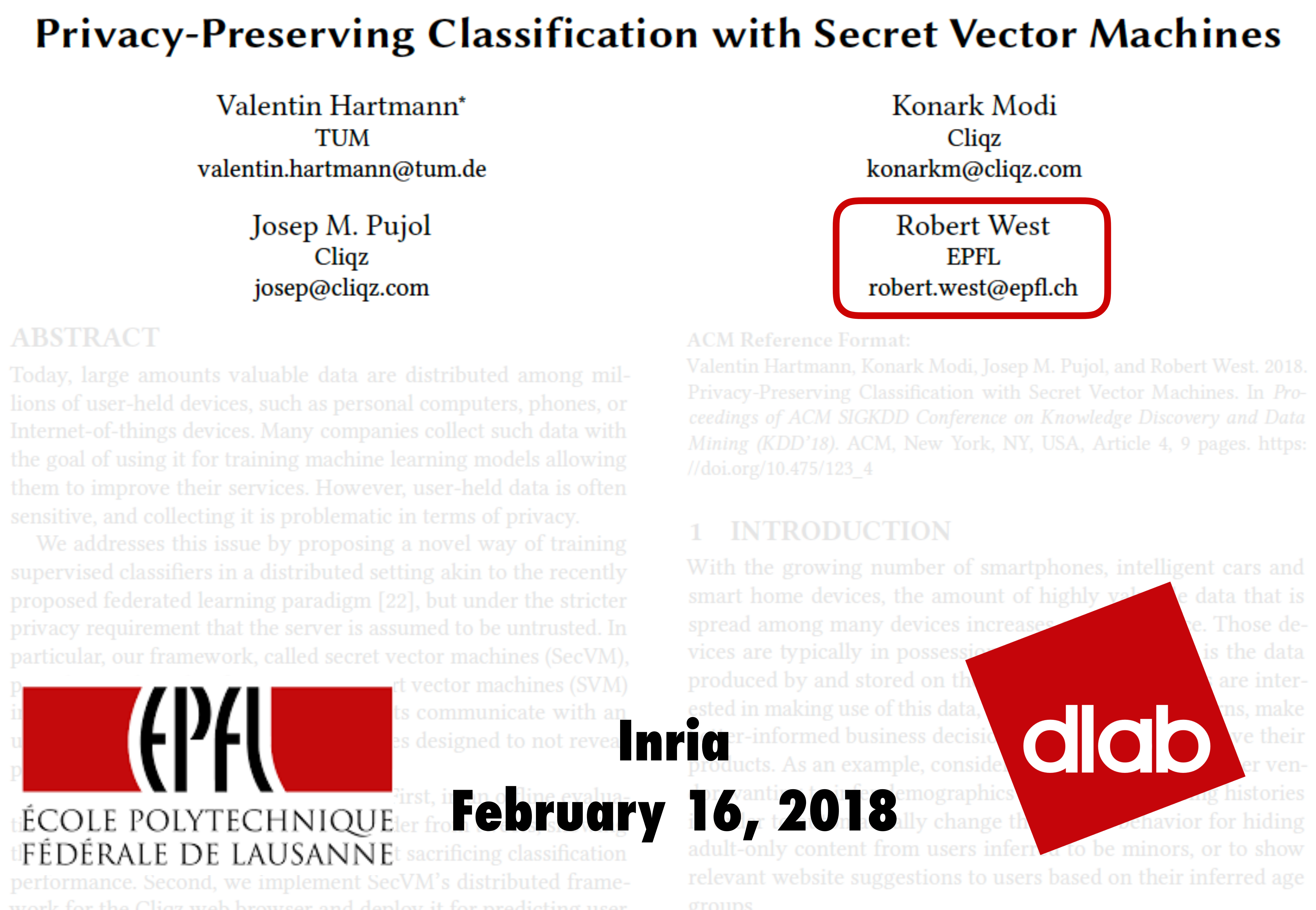


Privacy-Preserving Classification with Secret Vector Machines

Valentin Hartmann*
TUM
valentin.hartmann@tum.de

Konark Modi
Cliqz
konarkm@cliqz.com

Josep M. Pujol
Cliqz
josep@cliqz.com

Robert West
EPFL
robert.west@epfl.ch

ABSTRACT

Today, large amounts valuable data are distributed among millions of user-held devices, such as personal computers, phones, or Internet-of-things devices. Many companies collect such data with the goal of using it for training machine learning models allowing them to improve their services. However, user-held data is often sensitive, and collecting it is problematic in terms of privacy.

We addresses this issue by proposing a novel way of training supervised classifiers in a distributed setting akin to the recently proposed federated learning paradigm [22], but under the stricter privacy requirement that the server is assumed to be untrusted. In particular, our framework, called secret vector machines (SecVM), p rt vector machines (SVM) i ts communicate with an u es designed to not revea p

first, in an offline evalua- ti ler from , showing t t sacrificing classification performance. Second, we implement SecVM's distributed framework for the Cliqz web browser and deploy it for predicting user

ACM Reference Format:
Valentin Hartmann, Konark Modi, Josep M. Pujol, and Robert West. 2018. Privacy-Preserving Classification with Secret Vector Machines. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'18).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

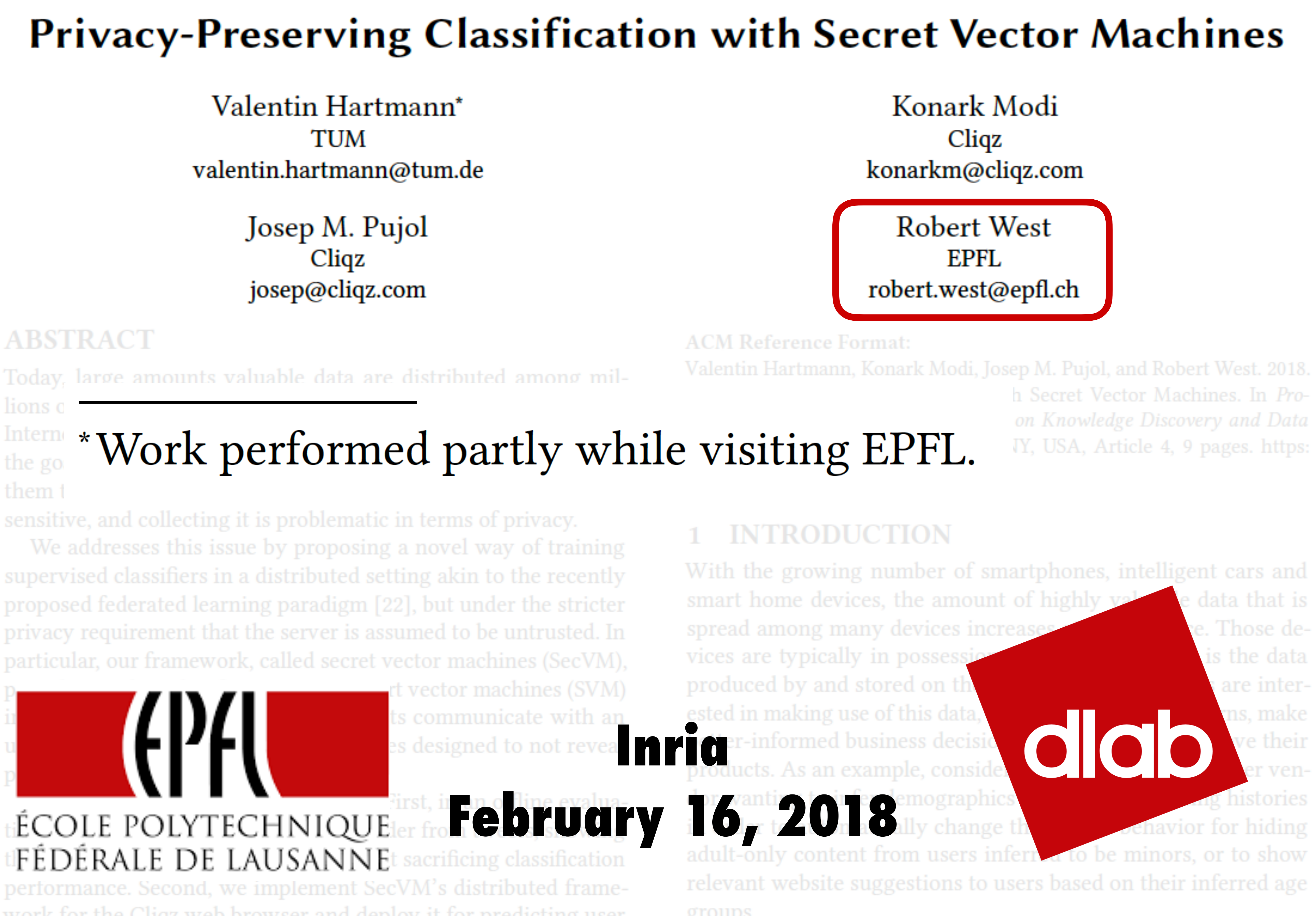
With the growing number of smartphones, intelligent cars and smart home devices, the amount of highly va e data that is spread among many devices increases e. Those devices are typically in possessi is the data produced by and stored on th are interested in making use of this data, ns, make er-informed business decisi ve their products. As an example, conside er vendor wanti demographics g histories i ally change th behavior for hiding adult-only content from users inferr to be minors, or to show relevant website suggestions to users based on their inferred age groups

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Inria

February 16, 2018

dlab



Privacy-Preserving Classification with Secret Vector Machines

Valentin Hartmann*
TUM
valentin.hartmann@tum.de

Konark Modi
Cliqz
konarkm@cliqz.com

Josep M. Pujol
Cliqz
josep@cliqz.com

Robert West
EPFL
robert.west@epfl.ch

ABSTRACT

Today, large amounts valuable data are distributed among millions o
Interne
the go
them t
sensitive, and collecting it is problematic in terms of privacy.

We addresses this issue by proposing a novel way of training supervised classifiers in a distributed setting akin to the recently proposed federated learning paradigm [22], but under the stricter privacy requirement that the server is assumed to be untrusted. In particular, our framework, called secret vector machines (SecVM), p
t vector machines (SVM)
i
ts communicate with an
u
es designed to not reveal
p
First, in an offline evalua-
ti
ler from
ti
t sacrificing classification performance. Second, we implement SecVM's distributed framework for the Cliqz web browser and deploy it for predicting user

ACM Reference Format:
Valentin Hartmann, Konark Modi, Josep M. Pujol, and Robert West. 2018.
h Secret Vector Machines. In Pro-
on Knowledge Discovery and Data
NY, USA, Article 4, 9 pages. https:

1 INTRODUCTION

With the growing number of smartphones, intelligent cars and smart home devices, the amount of highly val
e data that is spread among many devices increases
e. Those devices are typically in possessi
is the data produced by and stored on th
are interested in making use of this data,
ns, make better-informed business decisi
ve their products. As an example, conside
er vendor wanting to infer demographics
g histories in order to automatically change th
behavior for hiding adult-only content from users inferr
d to be minors, or to show relevant website suggestions to users based on their inferred age groups.

*Work performed partly while visiting EPFL.

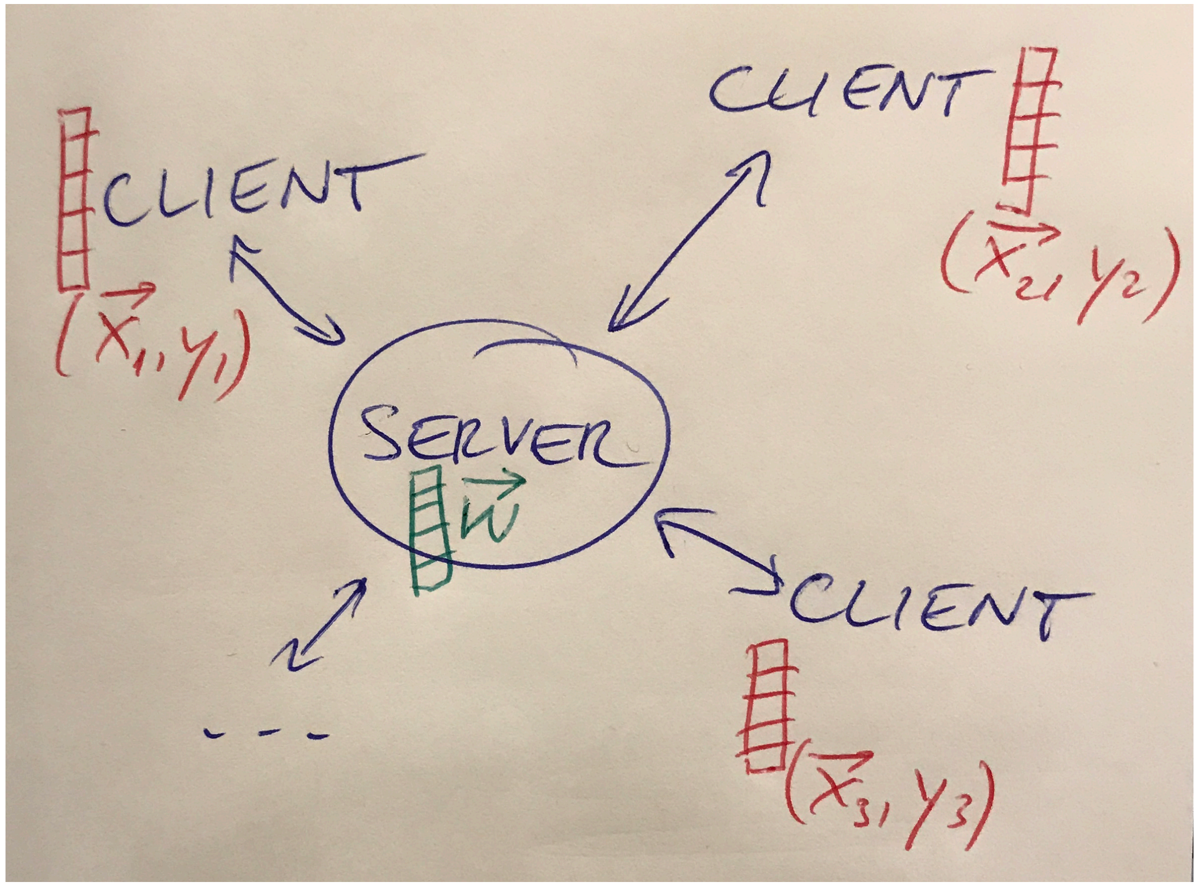
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Inria
February 16, 2018

dlab

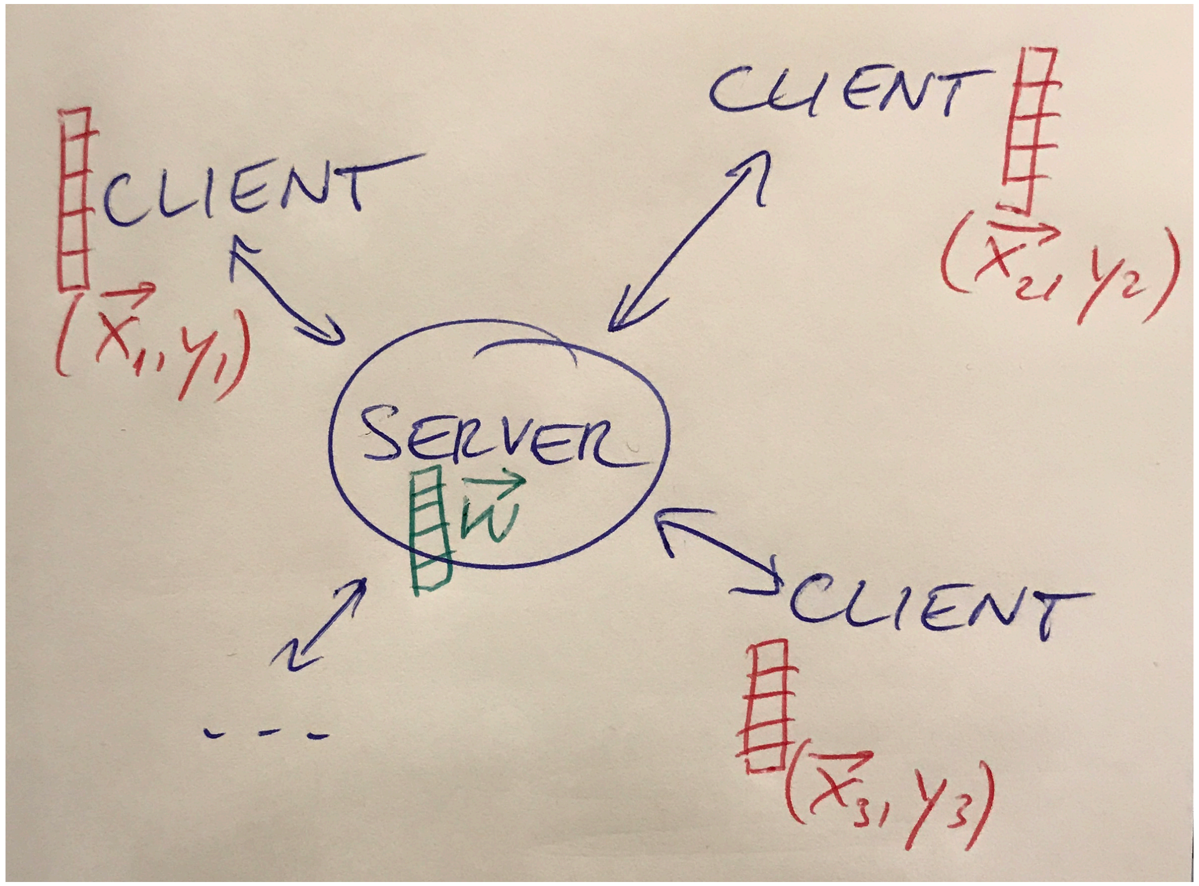
(More formal) problem statement

- Train a prediction model on distributed data



(More formal) problem statement

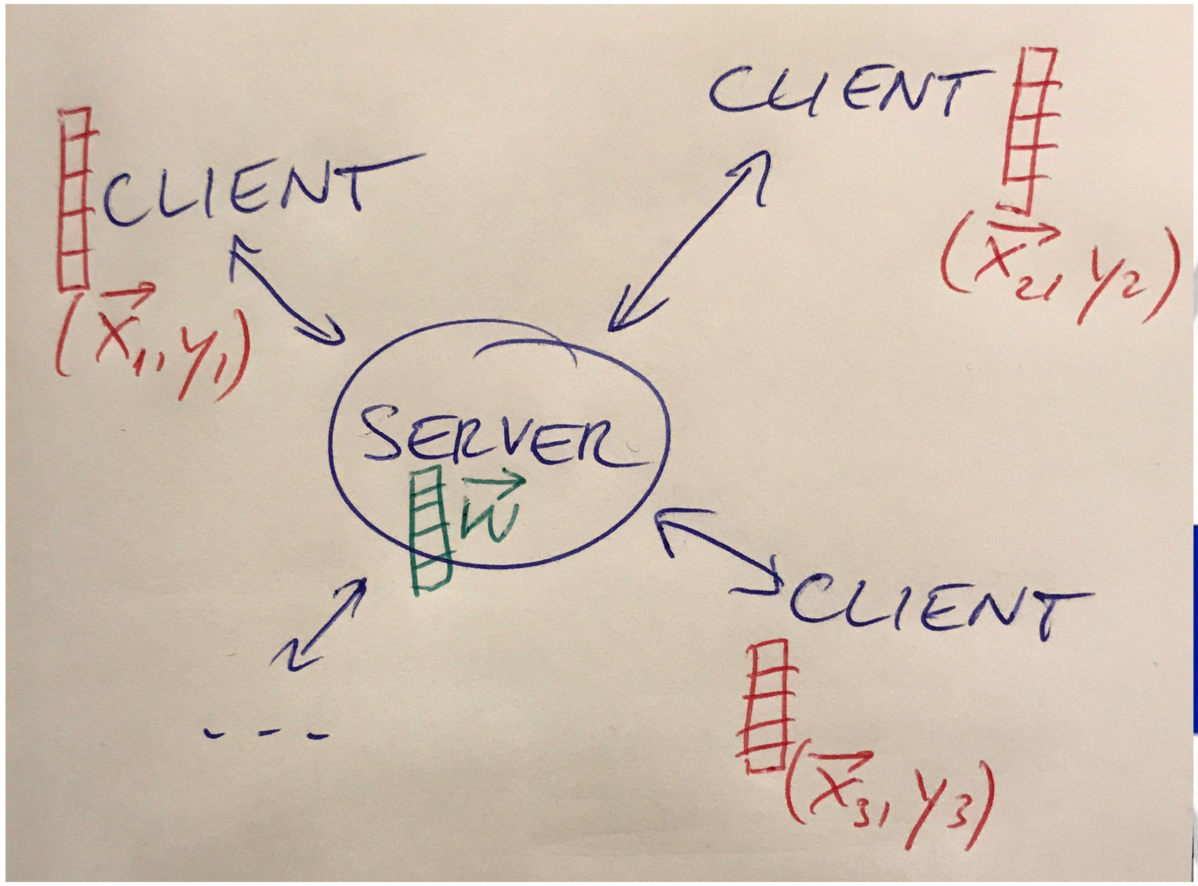
- Train a prediction model on distributed data



- Server is untrusted

(More formal) problem statement

- Train a prediction model on distributed data



I HATE THE FRENCH
VANILLA
100 PERCENT ALL-AMERICAN VANILLA ICE CREAM
ST★R SPANGLED
★ ICE CREAM ★
THE SWEET TASTE OF FREEDOM

- Server is untrusted
- ==> Vanilla machine learning not ok

sufficed to identify a user, then the other features would contain additional information. Likewise, if a single feature were unique to a certain user, the feature-label combination would give away their label. And even more, a feature itself could already contain sensitive information, e.g., if the features are strings the user types into a text box.

We therefore formalize our privacy requirement as follows:

> **Privacy requirement.** *The server must not be able to infer the label or any feature value for any individual client.*

4 PROPOSED SOLUTION

The loss function of a binary classification model often takes the form

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(w, x_i, y_i) + \frac{\lambda}{2}R(w) \qquad (1)$$

where N denotes the number of users or training samples, $x_i \in \mathbb{R}^d$ the feature vector, $y_i \in \{-1, 1\}$ the label of the i-th user, $w \in \mathbb{R}^d$ the

4 PROPOSED SOLUTION

The loss function of a binary classification model often takes the form

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(w, x_i, y_i) + \frac{\lambda}{2}R(w) \tag{1}$$

where N denotes the number of users or training samples, $x_i \in \mathbb{R}^d$ the feature vector, $y_i \in \{-1, 1\}$ the label of the i-th user, $w \in \mathbb{R}^d$ the parameter vector of the model, L the loss for an individual sample, R a regularization function that is independent from the data, and $\lambda > 0$ the regularization parameter. When using a subgradient method to train the model, the update for dimension j becomes

$$w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(w, x_i, y_i)}{\partial w_j} + \frac{\lambda}{2}\frac{\partial R(w)}{\partial w_j}\right), \tag{2}$$

where $\eta > 0$ is the learning rate parameter. The key observation that federated learning [22] and also our method rely on is that this

4 PROPOSED SOLUTION

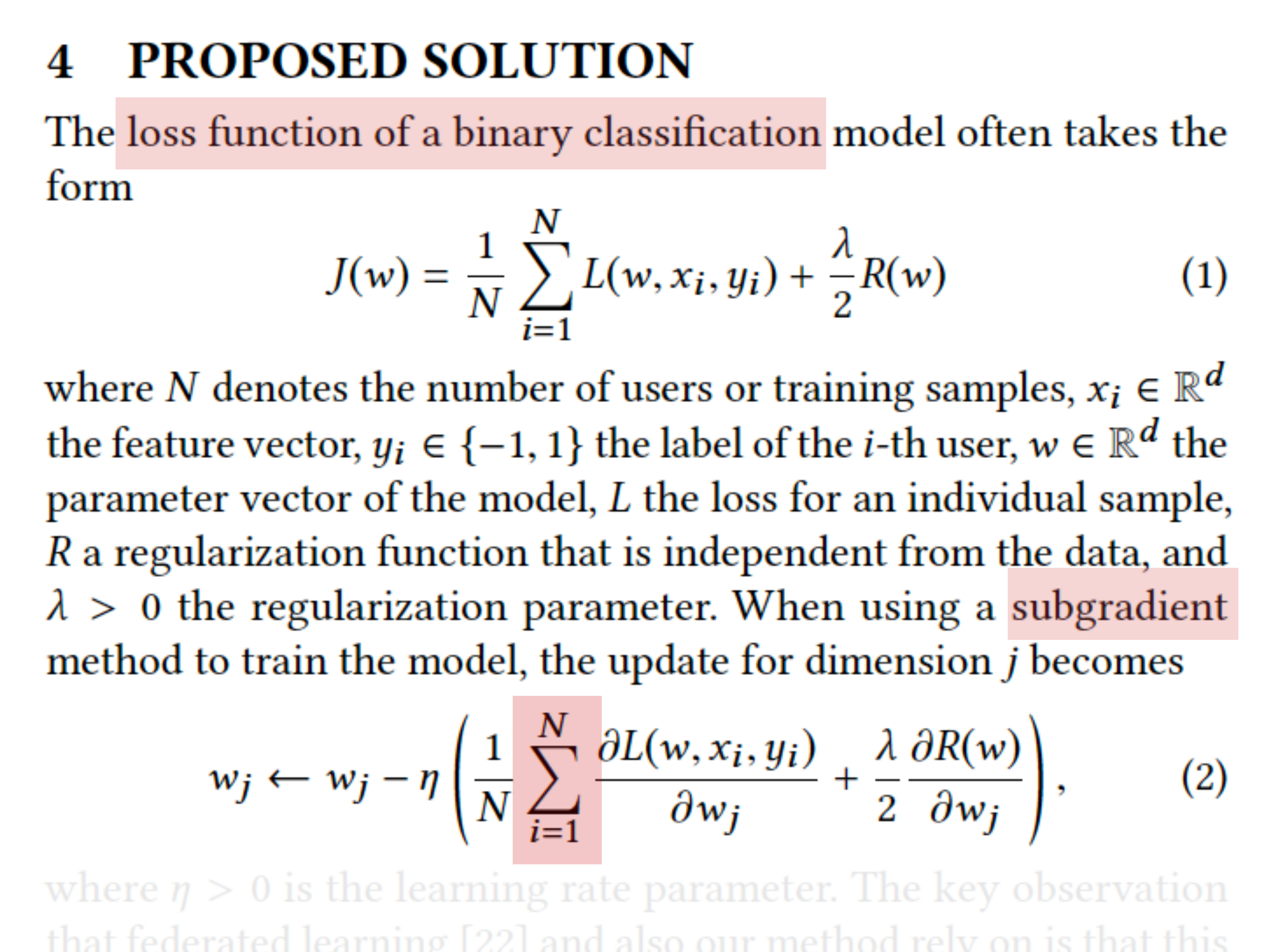
The loss function of a binary classification model often takes the form

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} L(w, x_i, y_i) + \frac{\lambda}{2} R(w) \tag{1}$$

where N denotes the number of users or training samples, $x_i \in \mathbb{R}^d$ the feature vector, $y_i \in \{-1, 1\}$ the label of the i-th user, $w \in \mathbb{R}^d$ the parameter vector of the model, L the loss for an individual sample, R a regularization function that is independent from the data, and $\lambda > 0$ the regularization parameter. When using a subgradient method to train the model, the update for dimension j becomes

$$w_j \leftarrow w_j - \eta \left(\frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(w, x_i, y_i)}{\partial w_j} + \frac{\lambda}{2} \frac{\partial R(w)}{\partial w_j} \right), \tag{2}$$

where $\eta > 0$ is the learning rate parameter. The key observation that federated learning [22] and also our method rely on is that this



4 PROPOSED SOLUTION

The loss function of a binary classification model often takes the form

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(w, x_i, y_i) + \frac{\lambda}{2}R(w) \qquad (1)$$

where N denotes the number of users or training samples, $x_i \in \mathbb{R}^d$ the feature vector, $y_i \in \{-1, 1\}$ the label of the i-th user, $w \in \mathbb{R}^d$ the parameter vector of the model, L the loss for an individual sample, R a regularization function that is independent from the data, and $\lambda > 0$ the regularization parameter. When using a subgradient method to train the model, the update for dimension j becomes

$$w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(w, x_i, y_i)}{\partial w_j} + \frac{\lambda}{2}\frac{\partial R(w)}{\partial w_j}\right), \qquad (2)$$

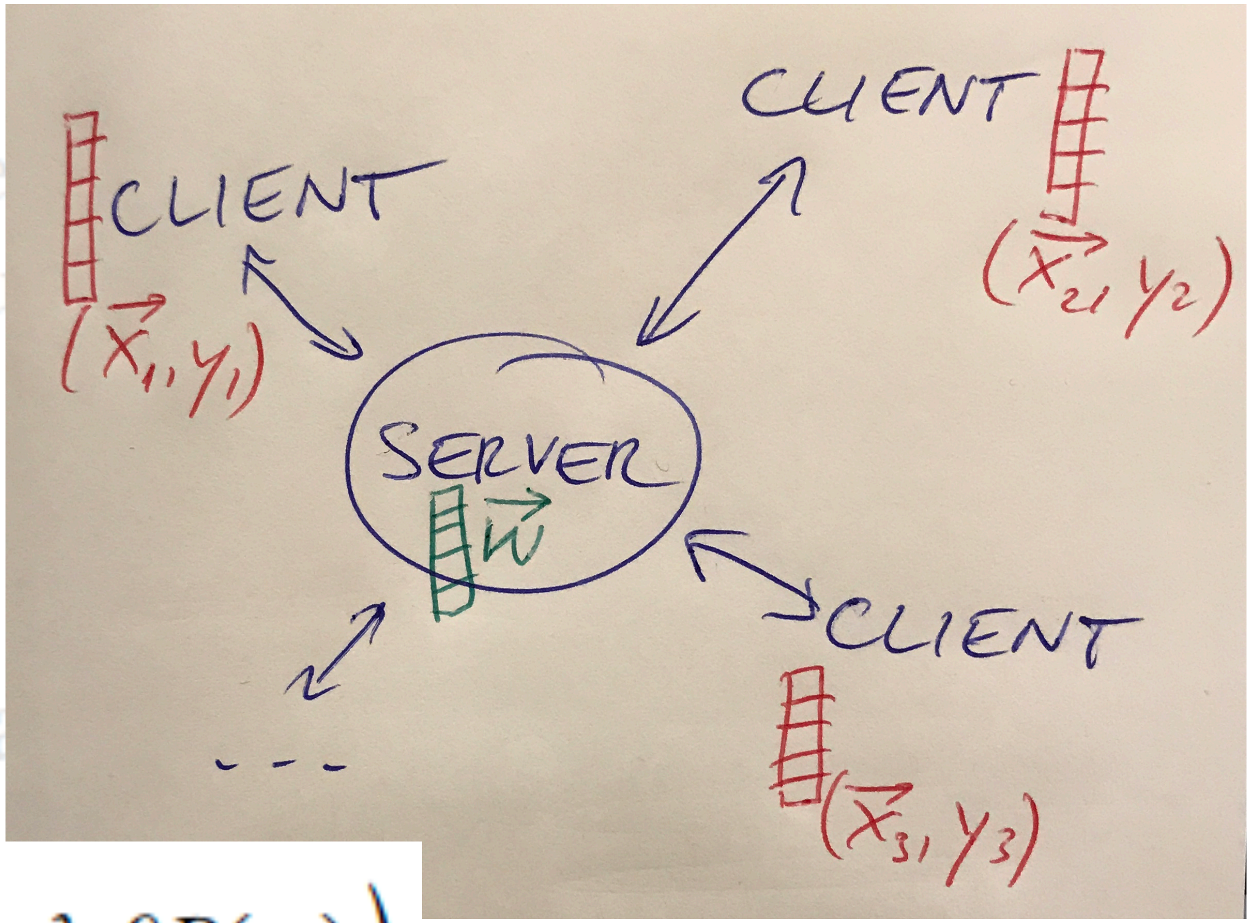
where $\eta > 0$ is the learning rate parameter. The key observation that federated learning [22] and also our method rely on is that this

update is the sum over values that each only depend on the data of a single user. We follow the general three-step process of federated learning:

(1) The server sends out the current model to the clients.
(2) The clients compute their local updates based on their data x_i, y_i and send it back to the server.
(3) The server sums up the individual updates and updates the model.

In its basic form, however, this doe
ment. In the following we will ou
could compromise the privacy and
to the final SecVM method.

4.1 Attack 1: Linkage vi



$$w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(w, x_i, y_i)}{\partial w_j} + \frac{\lambda}{2}\frac{\partial R(w)}{\partial w_j}\right)$$

an not only be done
s, building up a trace

learning:

(1) The server sends out the current model to the clients.
(2) The clients compute their local updates based on their c
x_i, y_i and send it back to the server.
(3) The server sums up the individual updates and updates
model.

In its basic form, however, this does not fulfill our privacy requ
ment. In the following we will outline the different attacks t
could compromise the privacy and our counter measures, that l
to the final SecVM method.

4.1 Attack 1: Linkage via IP addresses

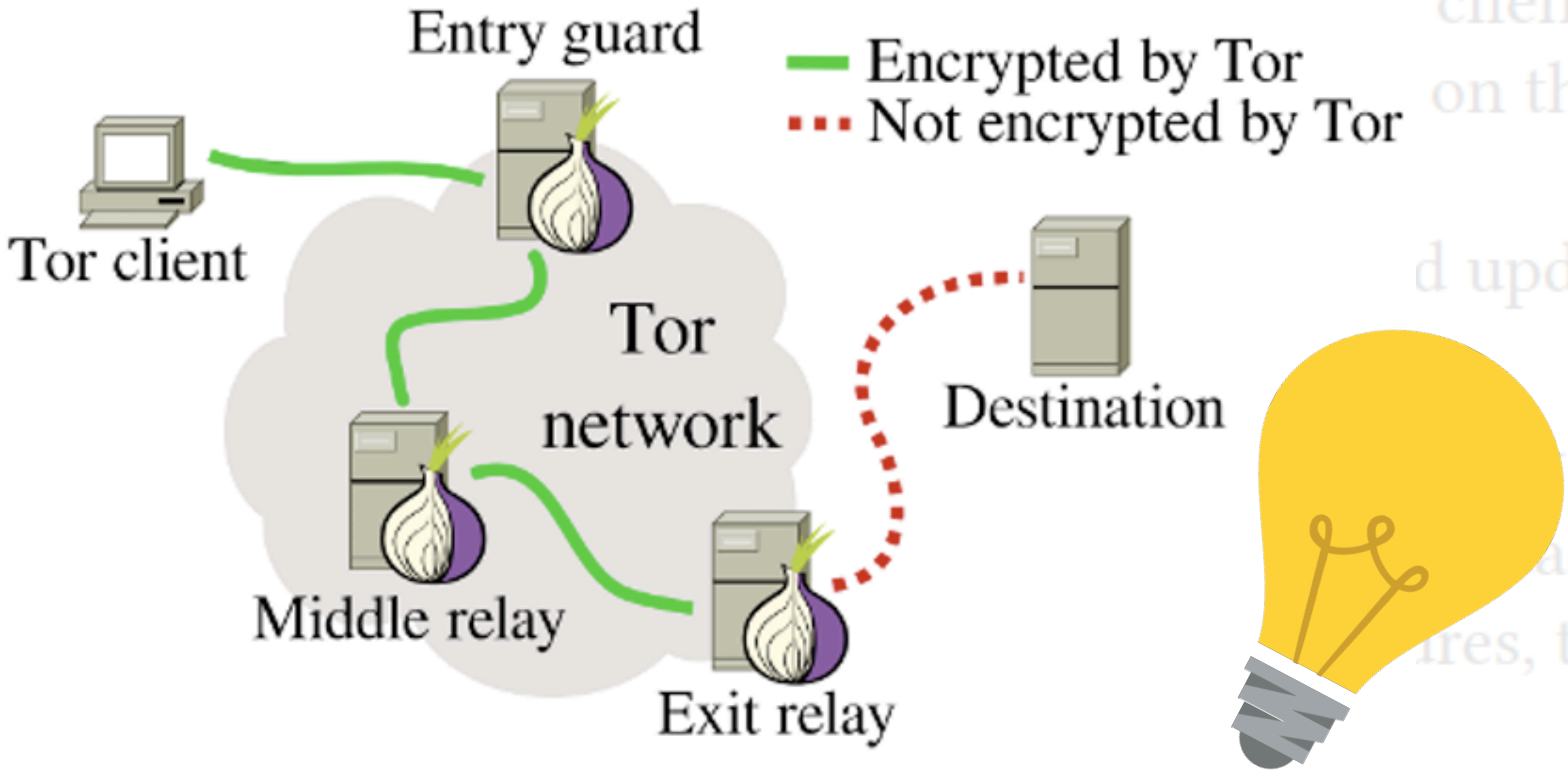
The most direct way to link data to a client is via the IP addr
attached to the packages it arrives in. This can not only be d
isolatedly, but even with subsequent messages, building up a tr
of data from a client over time. Thus the first step is to remove

learning:

(1) The server sends out the current model to the clients.

(2 … on their c

(3 … d updates



In its … requ
men… acks t
coul… res, that l
to th

4.1 Attack 1: Linkage via IP addresses

The most direct way to link data to a client is via the IP add attached to the packages it arrives in. This can not only be d isolatedly, but even with subsequent messages, building up a t of data from a client over time. Thus the first step is to remov

4.2 Attack 2: Identity inference via feature vectors

For linear models, (1) typically becomes

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i w^T x_i) + \frac{\lambda}{2} R(w). \tag{3}$$

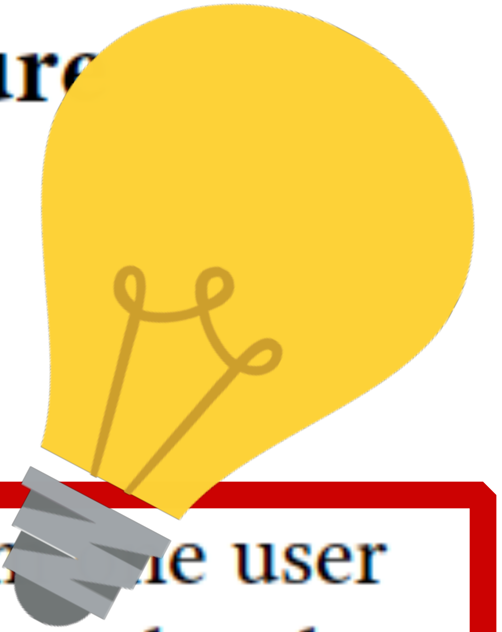
Taking the derivative with respect to dimension j, we get

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} L'(y_i w^T x_i)\, y_i x_{ij} + \frac{\lambda}{2} \frac{\partial R(w)}{\partial w_j}, \tag{4}$$

giving rise to the subgradient update rule

$$w_j \leftarrow w_j - \eta \left(\frac{1}{N} \sum_{i=1}^{N} L'(y_i w^T x_i)\, y_i x_{ij} + \frac{\lambda}{2} \frac{\partial R(w)}{\partial w_j} \right). \tag{5}$$

4.2 Attack 2: Identity inference via feature vectors



For linear models, (1) typically becomes

can use to prevent this: Not only is the local update from the user independent from the local updates of the other users, but also the update for one entry w_j does not rely on the update for any other entries. That means we can update each entry individually. We exploit this by splitting the update vector $\left(\frac{\partial L(w,x_i,y_i)}{\partial w_1},\ldots,\frac{\partial L(w,x_i,y_i)}{\partial w_d}\right)$ up into its individual entries and sending each entry together with its index as a separate package.

$$w_j \leftarrow w_j - \eta\left(\frac{1}{N}\sum_{i=1}^{N} L'(y_i w^T x_i)\, y_i x_{ij} + \frac{\lambda}{2}\frac{\partial R(w)}{\partial w_j}\right). \qquad (5)$$

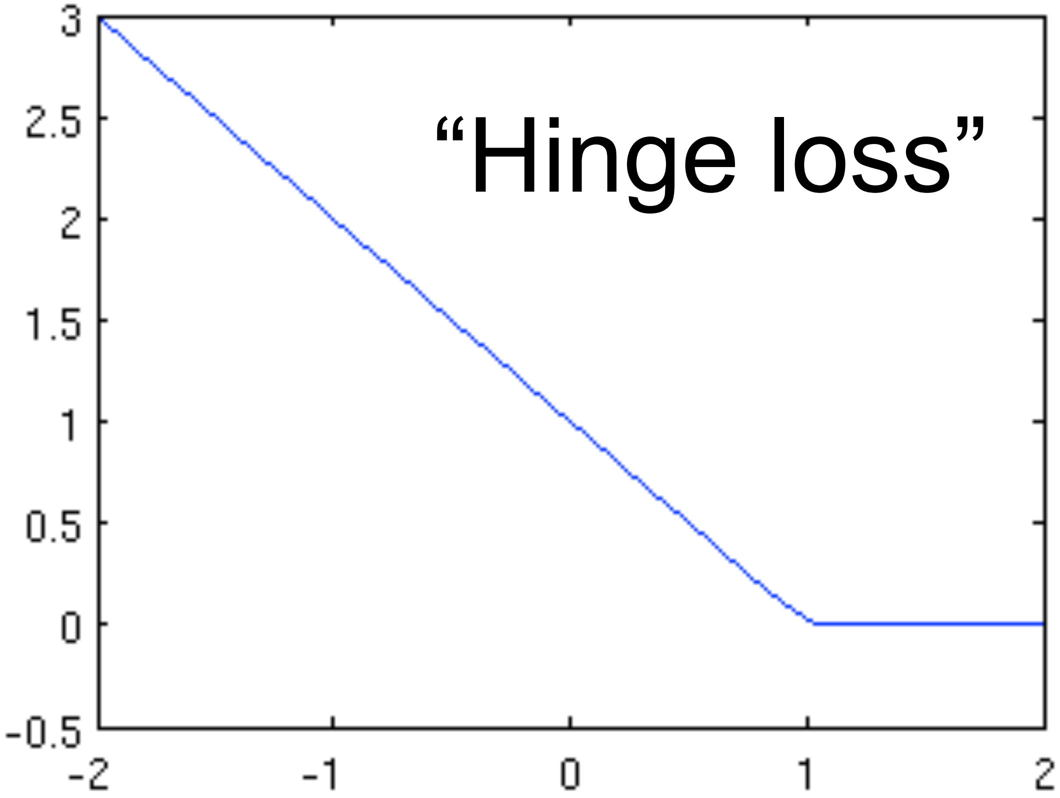
independent from the local updates of the other users, but also the update for one entry w_j does not rely on the update for any other entries. That means we can update each entry individually. We exploit this by splitting the update vector $\left(\frac{\partial L(w,x_i,y_i)}{\partial w_1}, \dots, \frac{\partial L(w,x_i,y_i)}{\partial w_d}\right)$ up into its individual entries and sending each entry together with its index as a separate package.

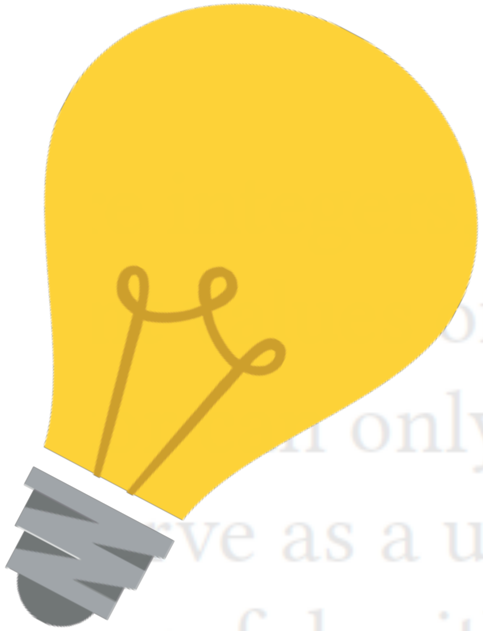
4.3 Attack 3: Package linkage via update values

The method proposed in the last paragraph ensures that no two updates for different entries of w are sent in the same package. Nevertheless, there is still the risk of identifying packages belonging to the same update vector post hoc. When using, e.g., L_2-regularized logisitic regression, we have $L(w, x_i, y_i) = \log(1 + \exp(-y_i w^T x_i))$ and

$$\frac{\partial L(w, x_i, y_i)}{\partial w_j} = \frac{y_i x_{ij}}{1 + \exp(y_i w^T x_i)}. \tag{6}$$

Attack 3: Package linkage via update values





it can easily happen that of j can be associated with only take integer values, the ve as a unique identifier.

careful with the choice of the con-

1. Support vector machines (SVMs) ave the loss function $L(w, x_i, y_i) = \max\{0, 1 - y_i w^T x_i\}$ with derivative

$$\frac{\partial L(w, x_i, y_i)}{\partial w_j} = \delta(1 - y_i w^T x_i)\, y_i x_{ij}, \qquad (7)$$

where $\delta(x)$ is 1 if $x > 0$ is true, and 0 otherwise. If we restrict ourselves to binary x_{ij}, we are in the lucky situation that the updates can only take the values -1, 0 and 1. The case of binary features can easily extended to integer features: Instead of sending one can only take the values −1, 0 and 1. The case of binary features can easily extended to integer features: Instead of sending one package containing $y_i x_{ij}$, the user sends $|x_{ij}|$ packages containing $y_i \operatorname{sgn}(x_{ij})$.

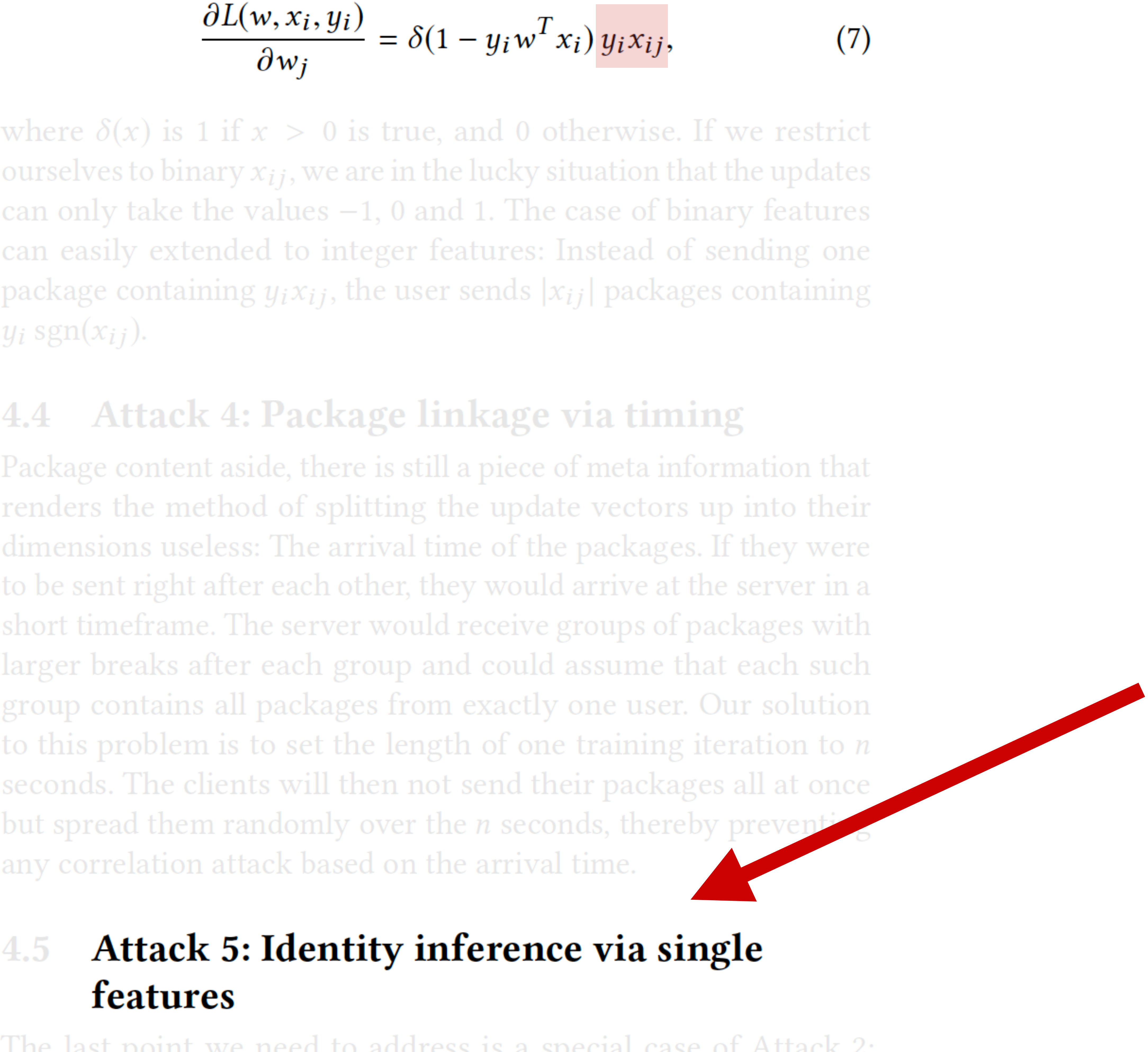
4.4 Attack 4: Package linkage via timing

Package content aside, there is still a piece of meta information that renders the method of splitting the update vectors up into their dimensions useless: The arrival time of the packages. If they were to be sent right after each other, they would arrive at the server in a short timeframe. The server would receive groups of packages with larger breaks after each group and could assume that each such group contains all packages from exactly one user. Our solution to this problem is to set the length of one training iteration to *n* seconds. The clients will then not send their packages all at once but spread them randomly over the *n* seconds, thereby preventing any correlation attack based on the arrival time.

can only take the values −1, 0 and 1. The case of binary features can easily extended to integer features: Instead of sending one package containing $y_i x_{ij}$, the user sends $|x_{ij}|$ packages containing $y_i \operatorname{sgn}(x_{ij})$.

4.4 Attack 4: Package linkage via timing

Package content aside, there is still a piece of meta information that renders the method of splitting the update vectors up into their dimensions useless: The arrival time of the packages. If [illegible] to be sent right after each other, they would arrive at the [illegible] a short timeframe. The server would receive groups of packages with larger breaks after each group and could assume that each such group contains all packages from exactly one user. Our solution to this problem is to set the length of one training iteration to n seconds. The clients will then not send their packages all at once but spread them randomly over the n seconds, thereby preventing any correlation attack based on the arrival time.



$$\frac{\partial L(w, x_i, y_i)}{\partial w_j} = \delta(1 - y_i w^T x_i)\, y_i x_{ij}, \tag{7}$$

where $\delta(x)$ is 1 if $x > 0$ is true, and 0 otherwise. If we restrict ourselves to binary x_{ij}, we are in the lucky situation that the updates can only take the values −1, 0 and 1. The case of binary features can easily extended to integer features: Instead of sending one package containing $y_i x_{ij}$, the user sends $|x_{ij}|$ packages containing $y_i \operatorname{sgn}(x_{ij})$.

4.4 Attack 4: Package linkage via timing

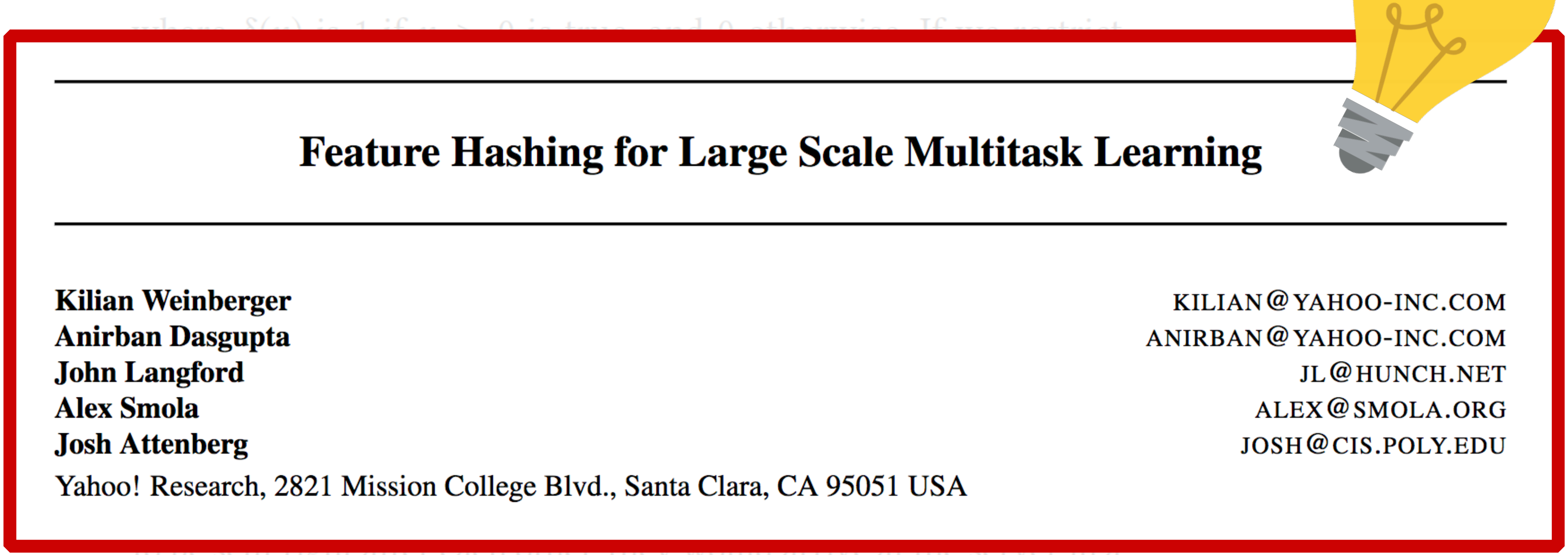
Package content aside, there is still a piece of meta information that renders the method of splitting the update vectors up into their dimensions useless: The arrival time of the packages. If they were to be sent right after each other, they would arrive at the server in a short timeframe. The server would receive groups of packages with larger breaks after each group and could assume that each such group contains all packages from exactly one user. Our solution to this problem is to set the length of one training iteration to n seconds. The clients will then not send their packages all at once but spread them randomly over the n seconds, thereby preventing any correlation attack based on the arrival time.

4.5 Attack 5: Identity inference via single features

The last point we need to address is a special case of Attack 2:

$$\frac{\partial L(w, x_i, y_i)}{\partial w_j} = \delta(1 - y_i w^T x_i)\, y_i x_{ij}, \qquad (7)$$

where $\delta(x)$ is 1 if $x \geq 0$ is true, and 0 otherwise. If we restrict



Feature Hashing for Large Scale Multitask Learning

Kilian Weinberger KILIAN@YAHOO-INC.COM
Anirban Dasgupta ANIRBAN@YAHOO-INC.COM
John Langford JL@HUNCH.NET
Alex Smola ALEX@SMOLA.ORG
Josh Attenberg JOSH@CIS.POLY.EDU
Yahoo! Research, 2821 Mission College Blvd., Santa Clara, CA 95051 USA

to be sent right after each other, they would arrive at the server in a short timeframe. The server would receive groups of packages with larger breaks after each group and could assume that each such group contains all packages from exactly one user. Our solution to this problem is to set the length of one training iteration to *n* seconds. The clients will then not send their packages all at once but spread them randomly over the *n* seconds, thereby preventing any correlation attack based on the arrival time.

4.5 Attack 5: Identity inference via single features

The last point we need to address is a special case of Attack 2:

Secret Vector Machine (SecVM)

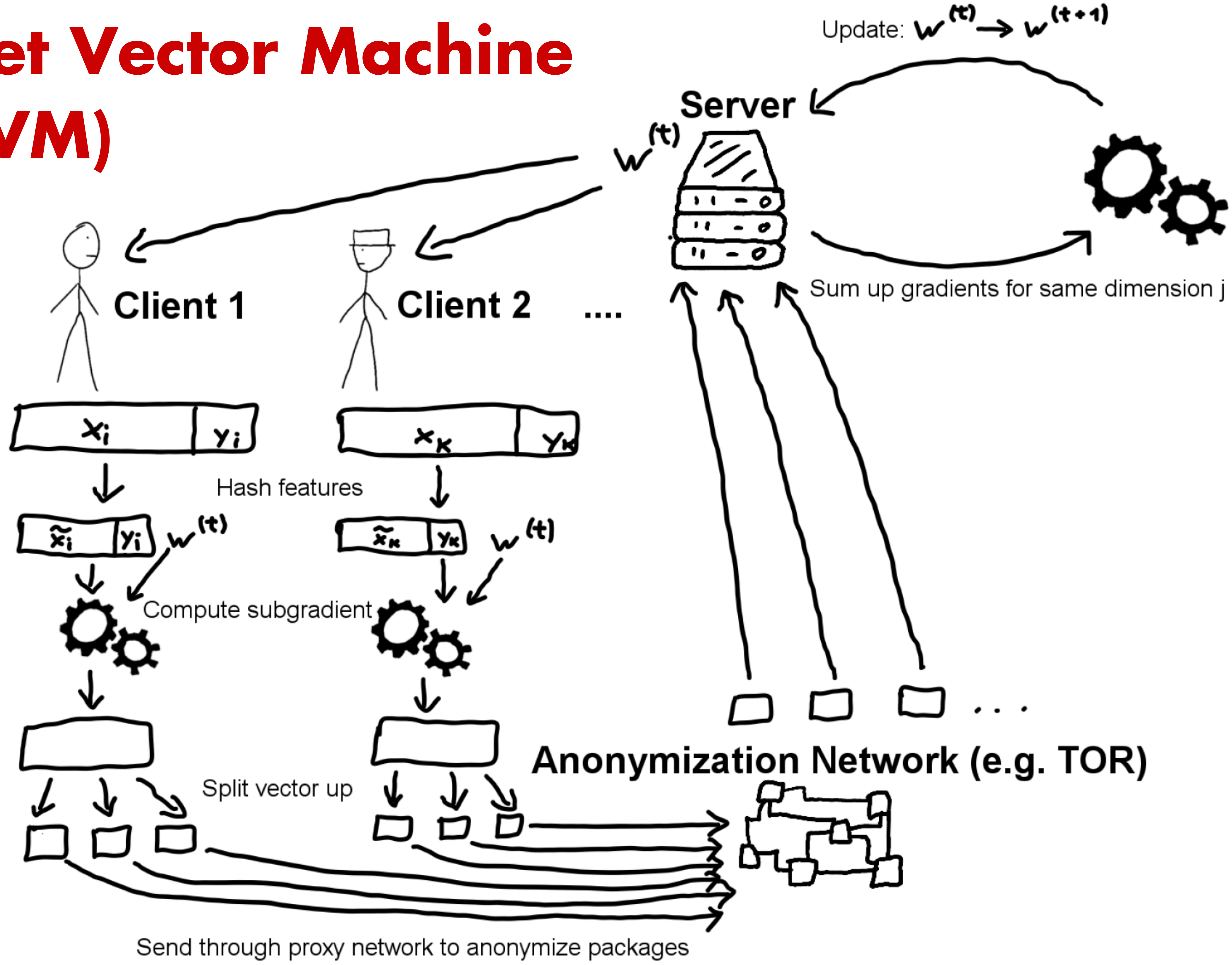

Update: $w^{(t)} \rightarrow w^{(t+1)}$
Server
$w^{(t)}$
Sum up gradients for same dimension j
Client 1
Client 2
....
x_i
y_i
x_K
y_K
Hash features
$\tilde{x}_i$
y_i
$w^{(t)}$
$\tilde{x}_K$
y_K
$w^{(t)}$
Compute subgradient
Split vector up
Anonymization Network (e.g. TOR)
Send through proxy network to anonymize packages

Figure 1: The SecVM model in a nutshell. As an initial step, all clients hash their feature vectors into a lower-dimensional space. Then the training procedure begins. In iteration t, the server sends out the current parameter vector $w = w^{(t)}$ to all clients. Each client i computes its local update $g_i = \delta(1 - y_i w^T x_i) y_i x_i$ and splits this vector into its individual entries g_{ij}. These entries, together with their indices j, are sent back to the server as individual packages at random points in time via a proxy network. The server sums up the updates g_{ij} corresponding to the same entry j of the parameter vector and updates the weight vector w. This procedure is repeated until the parameter vector has converged.

5 OFFLINE EVALUATION: GENDER INFERENCE FOR TWITTER USERS

We implemented and tested our approach in a real application with users connected via the internet, as described in Section 6. However, to assess its feasibility and to determine suitable hyperparameters, we first performed a test on an offline dataset. The authors of [6] generously provided us with their dataset of tweets collected from nearly 350,000 Twitter users, and demographic data inferred from their profiles. Around half of the users are labeled as male and half of them as female.

As the classification task for the SVM, we decided to predict the gender of a user from the words he used in his tweets. The feature space is therefore very high-dimensional – after preprocessing, we obtained 95,880,008 distinct words. On the other hand, the feature vectors are very sparse: The average number of words per user

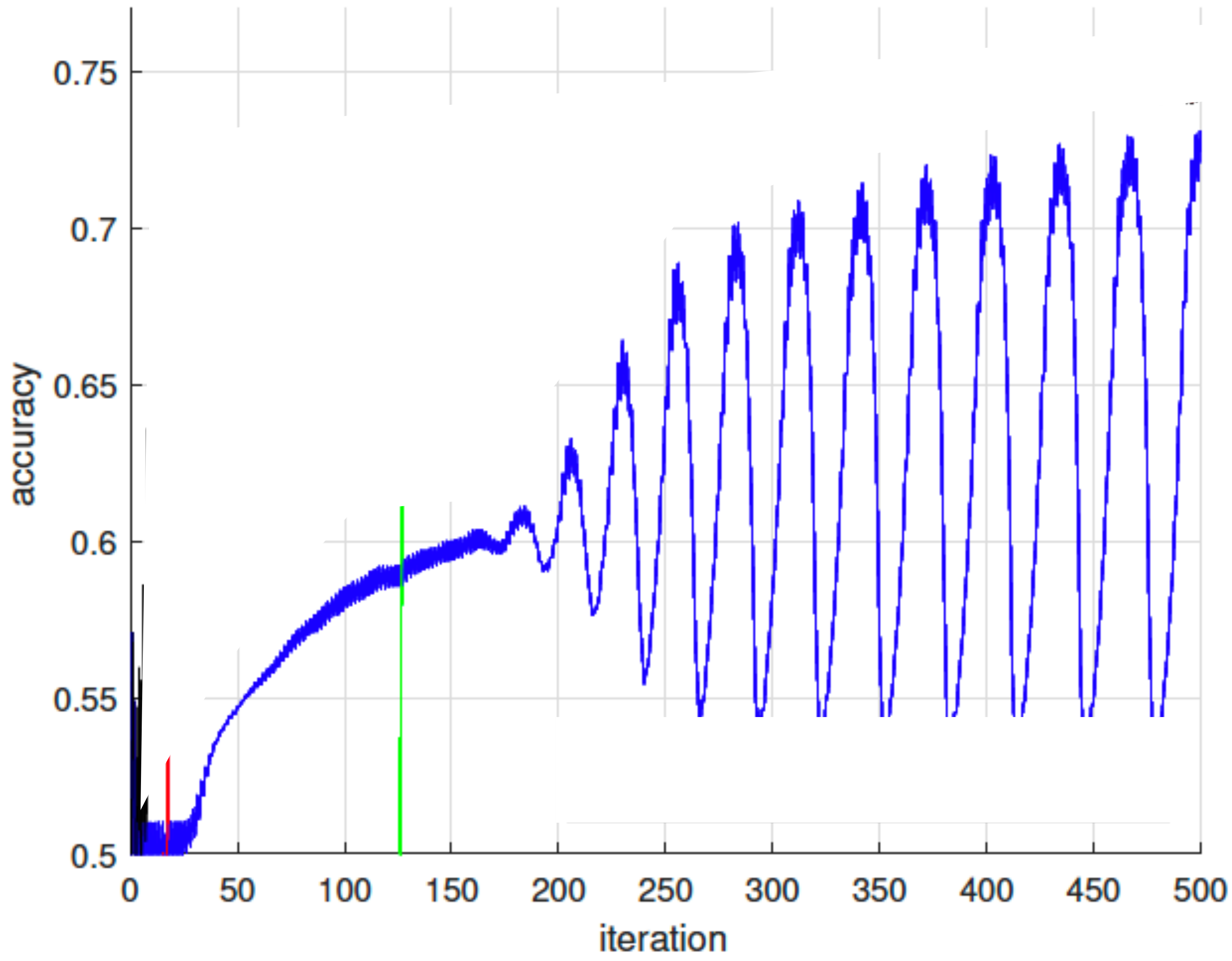


Figure 2: Learning curve for offline evaluation on Twitter gender prediction task (Sec. 5): test-set accuracy as function of subgradient-descent iteration number. No feature hashing used. We tried various numbers of averaged weight vectors, each shown as one curve.

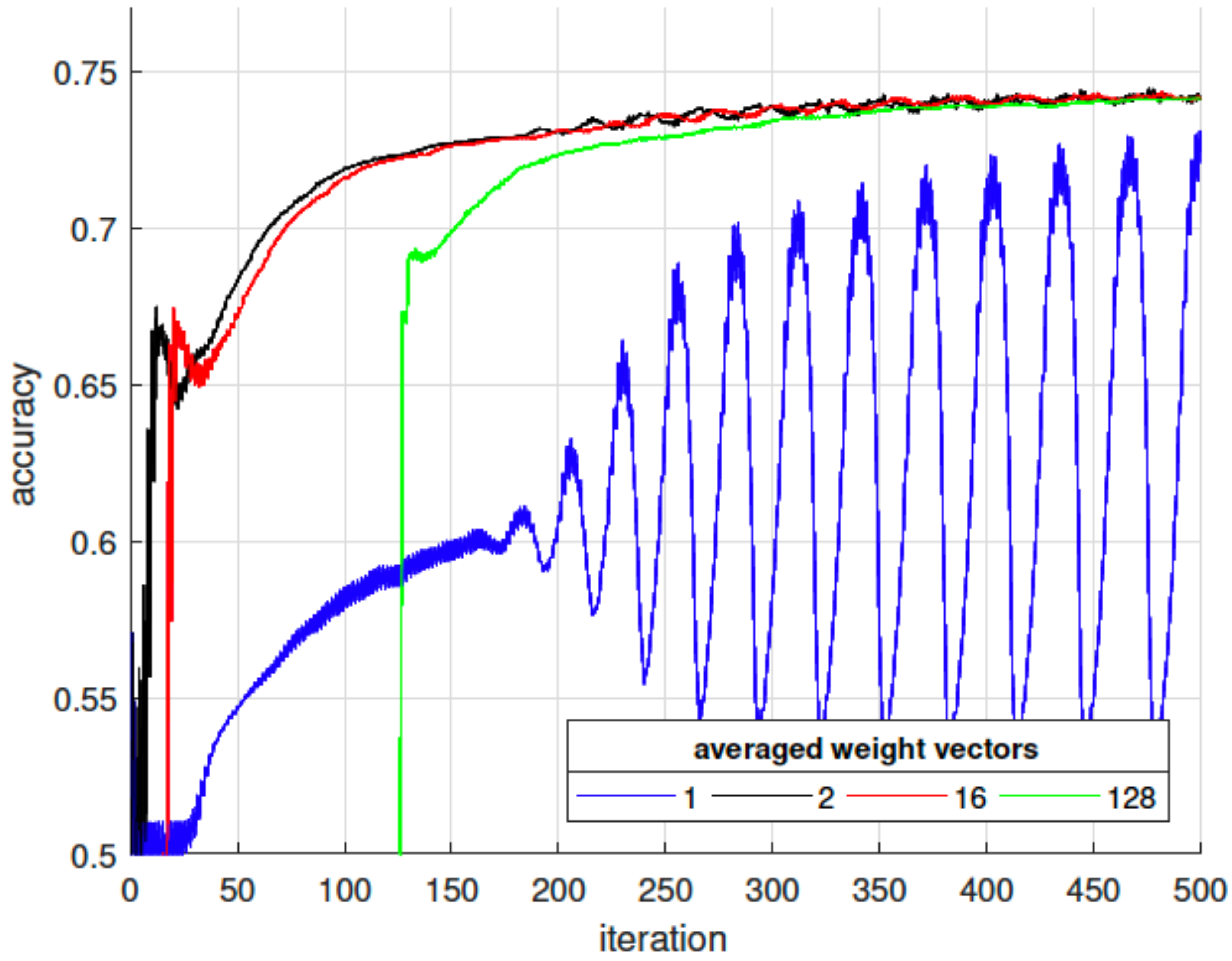


Figure 2: Learning curve for offline evaluation on Twitter gender prediction task (Sec. 5): test-set accuracy as function of subgradient-descent iteration number. No feature hashing used. We tried various numbers of averaged weight vectors, each shown as one curve.

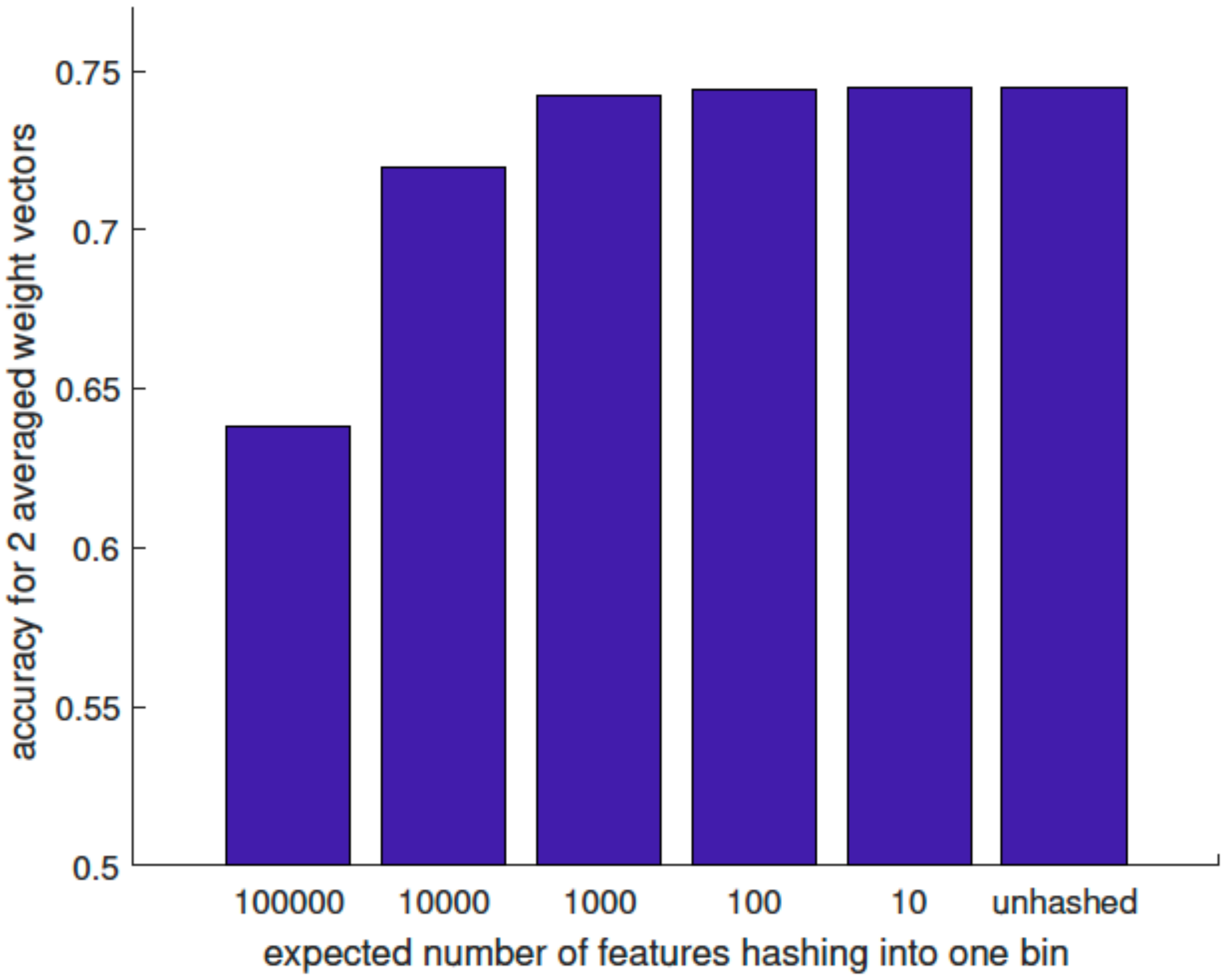
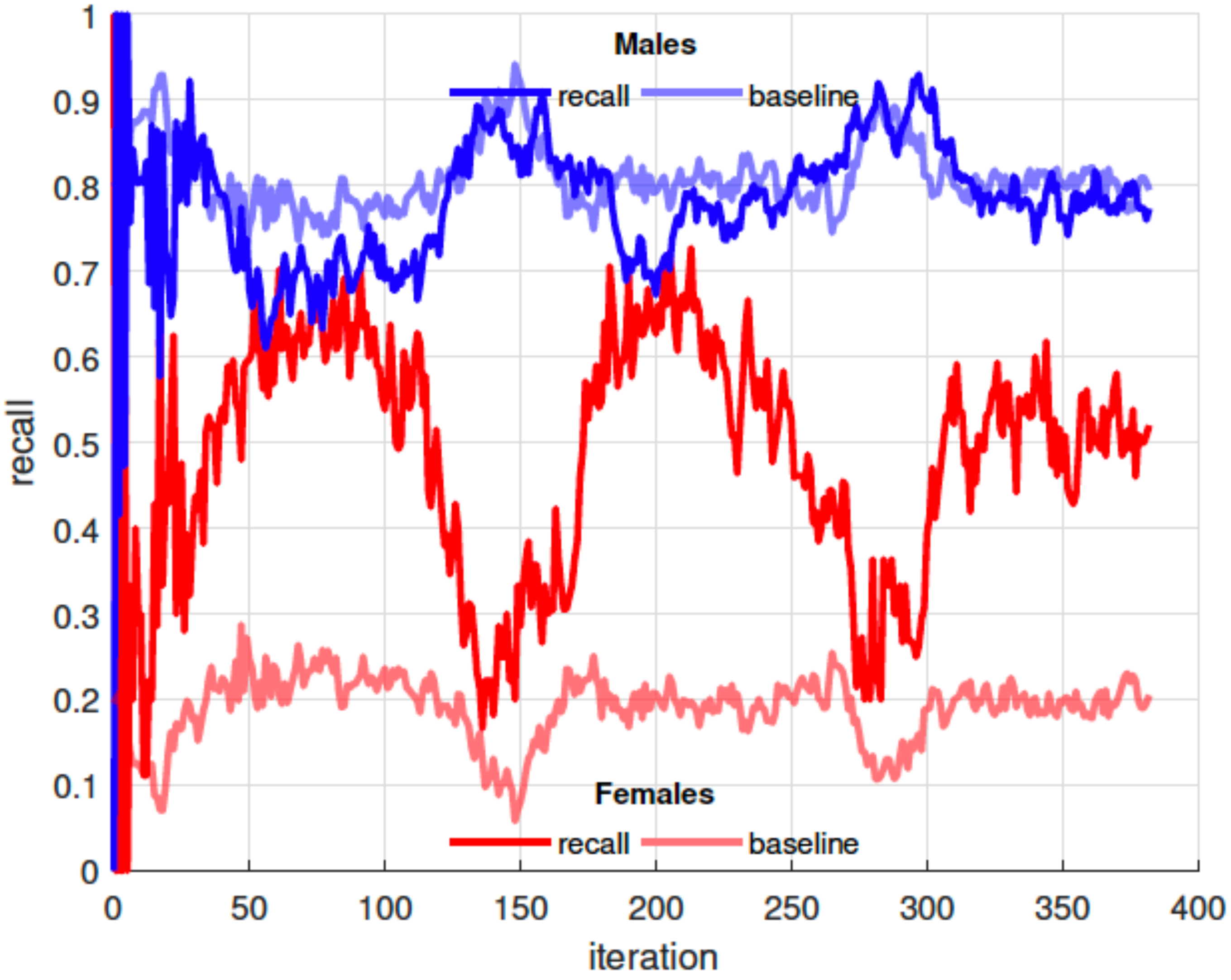
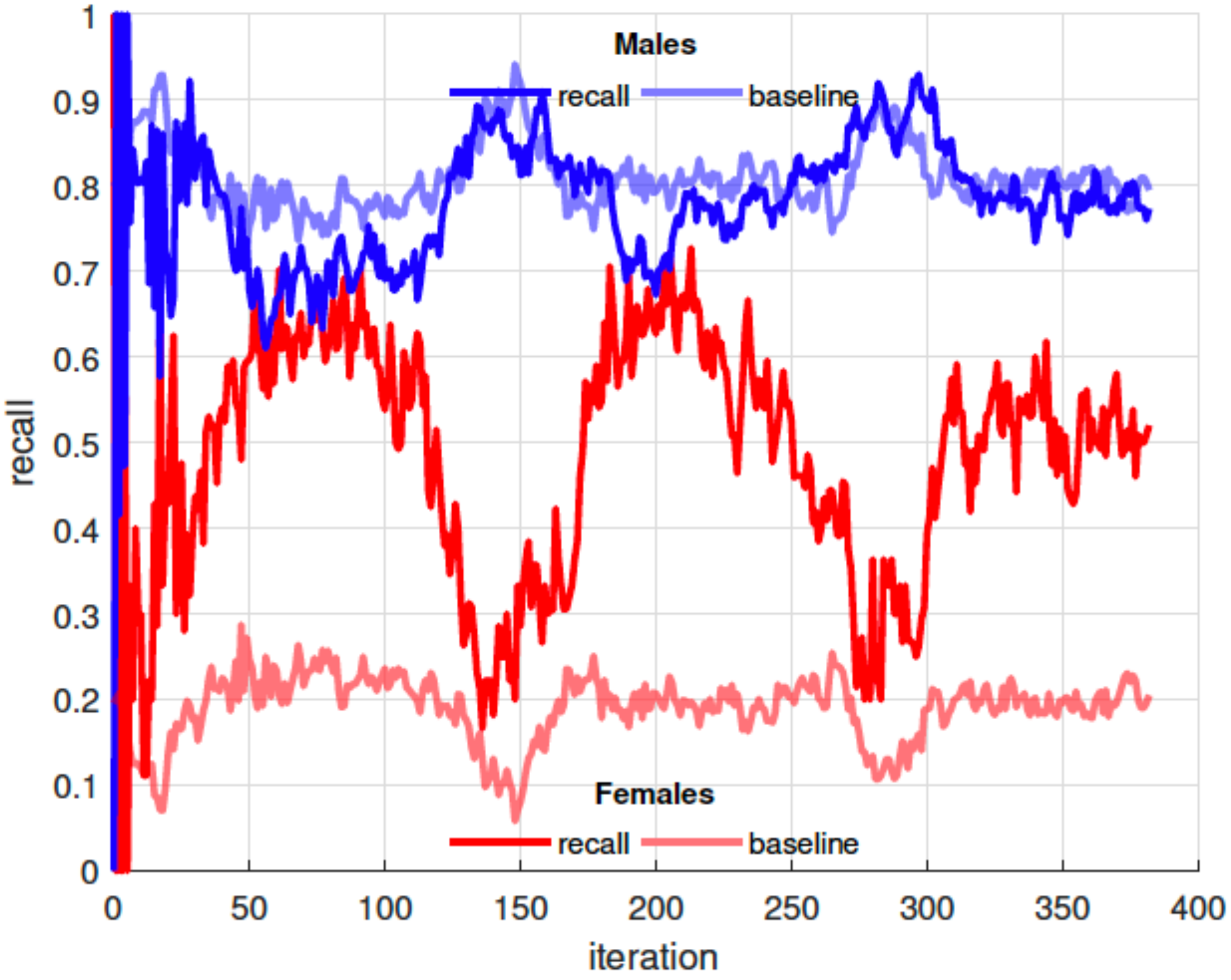


Figure 3: Offline evaluation on Twitter gender prediction task (Sec. 5): accuracy achieved for different numbers of features after feature hashing (using as weight vector for prediction the average of last two weight vectors seen during training; *cf.* Fig. 2).

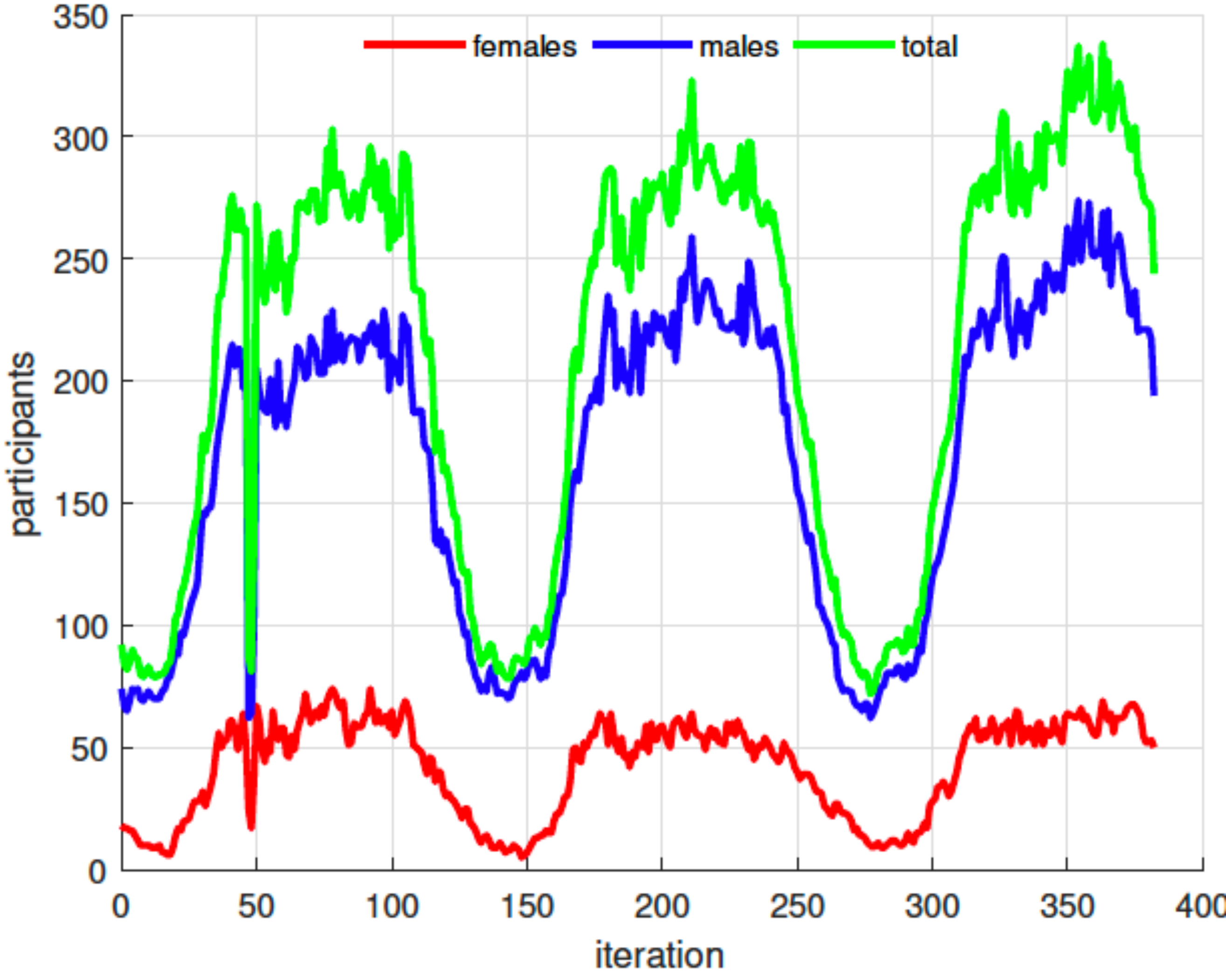
Learning curves

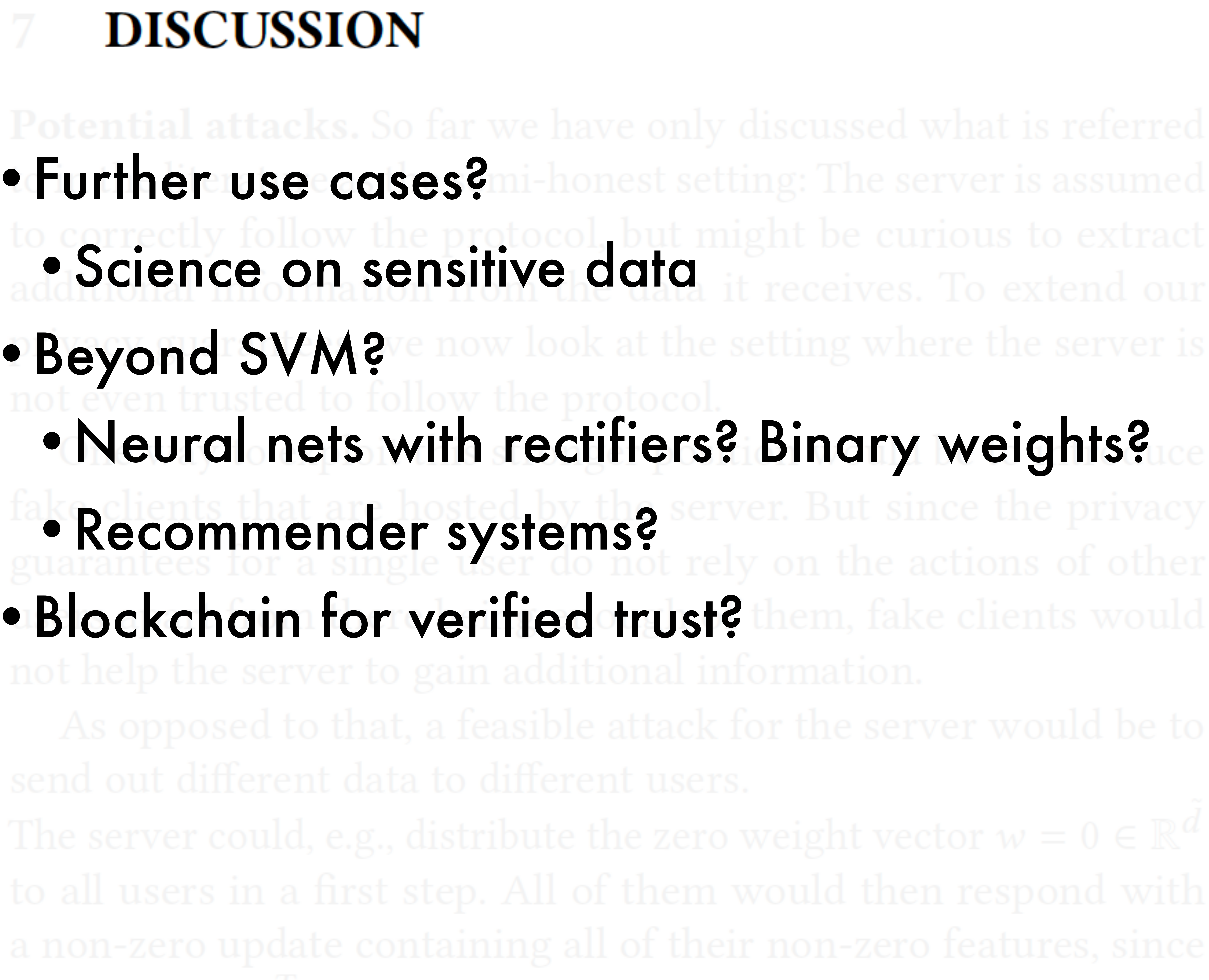


Learning curves



Number of test users





7 DISCUSSION

Potential attacks. So far we have only discussed what is referred [illegible]mi-honest setting: The server is assumed to correctly follow the protocol, but might be curious to extract additional information from the data it receives. To extend our [illegible]ve now look at the setting where the server is not even trusted to follow the protocol.

[illegible]ce fake clients that are hosted by the server. But since the privacy guarantees for a single user do not rely on the actions of other [illegible] them, fake clients would not help the server to gain additional information.

As opposed to that, a feasible attack for the server would be to send out different data to different users.
The server could, e.g., distribute the zero weight vector $w = 0 \in \mathbb{R}^{\tilde{d}}$ to all users in a first step. All of them would then respond with a non-zero update containing all of their non-zero features, since

- Further use cases?
  - Science on sensitive data
- Beyond SVM?
  - Neural nets with rectifiers? Binary weights?
  - Recommender systems?
- Blockchain for verified trust?

Merci!

robert.west@epfl.ch

Privacy-Preserving Classification with Secret Vector Machines

Valentin Hartmann*
TUM
valentin.hartmann@tum.de

Konark Modi
Cliqz
konarkm@cliqz.com

Josep M. Pujol
Cliqz
josep@cliqz.com

Robert West
EPFL
robert.west@epfl.ch

ABSTRACT

Today, large amounts valuable data are distributed among millions of user-held devices, such as personal computers, phones, or Internet-of-things devices. Many companies collect such data with the goal of using it for training machine learning models allowing them to improve their services. However, user-held data is often sensitive, and collecting it is problematic in terms of privacy.

We addresses this issue by proposing a novel way of training supervised classifiers in a distributed setting akin to the recently proposed federated learning paradigm [22], but under the stricter privacy requirement that the server is assumed to be untrusted. [illegible] particular, our framework, called secret vector machines (SecV[illegible] provides an algorithm for training support vector machines (SV[illegible] in a setting in which data-holding clients communicate with [illegible] untrusted server by exchanging messages designed to not re[illegible] personally identifying information.

We evaluate our model in two ways. First, in an offline ev[illegible] tion, we train SecVM to predict user gender from tweets, sho[illegible] that we can preserve user privacy without sacrificing classifi[illegible] performance. Second, we implement SecVM's distributed framework for the Cliqz web browser and deploy it for predicting user gender in a large-scale online evaluation with thousands of clients, outperforming baselines by a large margin and thus showcasing that SecVM is practicable in production environments.

Overall, this work demonstrates the feasibility of machine learning on data from thousands of users without collecting any personal data. We believe this is an innovative approach that will help reconcile machine learning with data privacy.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Valentin Hartmann, Konark Modi, Josep M. Pujol, and Robert West. 2018. Privacy-Preserving Classification with Secret Vector Machines. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'18).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

With the growing number of smartphones, intelligent cars and smart home devices, the amount of highly valuable data that is spread am[illegible]ny devices increases at a rapid pace. Those devices are [illegible] in possession of end users and so is the data produce[illegible]ored on them. Of course companies are inter[illegible] of this da[illegible]etect usa[illegible] [illegible] [illegible] to au[illegible] the default be[illegible] for hidin[illegible] [illegible]ly co[illegible] or to show rel[illegible]t websi[illegible] [illegible]sed on [illegible]ferred age groups.

The classical way of building the necessary prediction model would be to collect the users' data on a central server and then run a machine learning algorithm on it. But this comes with severe disadv[illegible]tages. First, the user has to put the neces[illegible] in [illegible] Eve[illegible] between the two parties is very imbalanced, [illegible] the European Union's General Data Protection Regulation (GDPR) [12] and e-privacy frameworks try to rebalance the relationship to be more fair. But still, even in the case of perfect regulation, the collection of user data incurs a privacy risk. There are many ways in which privacy could be compromised: Hacks leading to a data breach [2], disgruntled or unethical employees that use the data for their own benefit [10], companies going bankrupt and selling the data as assets [29], and of course government-issued subpoenas and backdoors [18, 26]. All in all, it is safe to assume that gathering users' data puts their privacy at risk, regardless of the comprehensiveness of the data management policies in place.

It is thus desirable to be able to build prediction models without learning any personally identifying information about the individual users whose data is used in the training process. For instance, the party who is training the model should not be able to infer labels or feature values of individual users. This requirement immediately precludes us from using the standard machine learning setup, where feature vectors for all users are stored in a feature matrix, labels in a label vector, and an optimization algorithm is then used to find the model parameters that minimize a given loss function.

The issue with the vanilla machine learning setup is that the party training the model sees all data—features and labels—at the same time, which typically makes it easy to infer user identities, even if the data is pseudo-anonymize, i.e., if actual user ids have been replaced with random identifiers. One way to achieve this is by tying together multiple features associated with the same user, as was the case with the now-infamous AOL data breach, where users thought to be anonymized were identified by analyzing all their search queries together [7]. Moreover, user privacy can also be compromised by correlating the pseudo-anonymized data with external datasets, which was done with the Netflix movie rating dataset [28].

One way to address these problems would be to adopt homomorphic encryption [14], where users encrypt their data before sharing it with other parties, and where the encryption scheme is devised such that meaningful operations (such as training machine learning models) can be performed directly on the encrypted data without the need for decrypting it first. Unfortunately, homomorphic enryption is not a sufficiently mature technology yet to be [illegible] complex tasks such as training a machine learning model. [illegible]her recently proposed way forward is given by the para[illegible] federated learning [22]. Here, model fitting does not happen [illegible]on the machines of a single party; rather, it works on dis[illegible] data without the need for central aggregation. In essence, [illegible]d learning models perform gradient descent in a distributed way, where, instead of sharing their raw data with the server who [illegible]ing the model, clients only share the gradient updates nec[illegible] improve the loss function locally for their personal data. [illegible]e this is a promising approach, it was not originally designed with the goal of preserving privacy. Later extensions have addressed this issue, by injecting random noise into the data on the client-side before sending it to the server [4] or by using cryptography [8].

Present work: Secret vector machines (SecVM). This work pro[illegible]oses a novel and different approach to training a supervised ma[illegible]hine learning classifier in a privacy-preserving manner. Crucially, in our setup, the server is assumed to be *untrusted*, i.e., potentially malicious. Our key observation is that support vector machines (SVM), a popular machine learning model, are particularly well-suited for privacy-preserving classification, due to the hinge loss function used in its objectives: when features and labels are binary (as is often the case), the SVM gradient can only take on the three discrete values −1, 0, and 1, which means particularly little information about the client is revealed.

Starting from this observation, we identify additional ingredients necessary to maintain user privacy and design a distributed protocol for training SVMs. We term our algorithm *secret vector machine (SecVM)*. As in the federated learning setup [22], the SecVM model is trained on the server side in collaboration with the data-owning clients. What sets SecVM apart from prior proposals for privacy-preserving federated learning is that it assumes an untrusted server and works without adding random noise. Instead, it leverages the above observation regarding the SVM loss and makes the recovery of the original data from the updates shared by clients impossible by means of feature hashing, splitting updates into small chunks, and

*Work performed partly while visiting EPFL.

KDD'18, August 2018, London, UK
© 2018 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4