ML Confidential
Machine Learning on Encrypted Data

Michael Naehrig

Microsoft Research
mnaehrig@microsoft.com

joint work with
Thore Graepel (MSR Cambridge)
and Kristin Lauter

Crypto Group Lunch, 27 July 2012

A simple machine learning task

Supervised learning

- Goal: derive a function from labelled training data
- Function can "reasonably" label test data according to the experience learned from the training data

A simple example: binary classification

- Given a set T of m samples of labelled training data $(\mathbf{x}, y_{\mathbf{x}}) \in \mathbb{R}^n \times Y$, where $Y = \{-1, 1\}$
- derive a function $f : \mathbb{R}^n \to Y$ that labels a test vector $\mathbf{x}$ by a "reasonable" $y_{\mathbf{x}} = f(\mathbf{x})$

Linear Means classifier

- Divide training data T into classes T_{+1} and T_{-1} according to their label

$$T_{\pm 1} = \{\mathbf{x} \in T \mid y_{\mathbf{x}} = \pm 1\}, \quad m_{\pm 1} = |T_{\pm 1}|$$

- Compute class-conditional mean vectors

$$\mathbf{m}_{+1} = \frac{1}{m_{+1}} \sum_{\mathbf{x} \in T_{+1}} \mathbf{x} = \frac{\mathbf{s}_{+1}}{m_{+1}}, \quad \mathbf{m}_{-1} = \frac{1}{m_{-1}} \sum_{\mathbf{x} \in T_{-1}} \mathbf{x} = \frac{\mathbf{s}_{-1}}{m_{-1}}$$

- Compute difference vector $\mathbf{w}^* = \mathbf{m}_{+1} - \mathbf{m}_{-1}$
- and mid-point between means $\mathbf{x}_0 = (\mathbf{m}_{+1} + \mathbf{m}_{-1})/2$
- define a hyperplane between the means, “separating” the two classes

Linear Means classifier

The score function (given a test vector $\mathbf{x} \in \mathbb{R}^n$) is:

$$f^*(\mathbf{x}; \mathbf{w}^*, c^*) = \mathbf{w}^{*T}\mathbf{x} - c^*$$

- where $c^* = \mathbf{w}^{*T}\mathbf{x}_0$
- classification $y_{\mathbf{x}} = \mathrm{sign}(f^*(\mathbf{x}; \mathbf{w}^*, c^*))$
- f^* is linear in $\mathbf{x}$, quadratic in training data
(considering the numbers m_{+1} and m_{-1} to be constants)

Polynomial learning algorithm

Definition: A learning algorithm

$$A : (\mathbb{R}^n \times \mathcal{Y})^m \times \mathbb{R}^n \to \mathcal{Y}$$

is called *polynomial of degree* d if it is a polynomial of degree d in its arguments (including training data).

- Linear Means classifier is polynomial of degree 2
(if we consider m_{+1} and m_{-1} to be constants)
(if we forget about the sign)

In a division-free world

Imagine you haven't yet learned how to divide real numbers…
You only know how to add, subtract and multiply.

- Come up with algorithms that avoid division
- multiply through with all denominators
- keep denominators separate

same idea: projective coordinates for elliptic curve arithmetic

- cost: more multiplications

Division-Free Linear Means classifier

- Replace means $\mathbf{m}_{\pm 1}$ by

$$m_{-1} \cdot \sum_{\mathbf{x} \in T_{+1}} \mathbf{x} = m_{-1}\mathbf{s}_{+1}, \quad m_{+1} \cdot \sum_{\mathbf{x} \in T_{-1}} \mathbf{x} = m_{+1}\mathbf{s}_{-1}$$

- and compute $\tilde{\mathbf{w}}^* := m_{-1}\mathbf{s}_{+1} - m_{+1} \cdot \mathbf{s}_{-1} = m_{+1}m_{-1}(\mathbf{m}_{+1} - \mathbf{m}_{-1}) = m_{+1} \cdot m_{-1}\mathbf{w}^*$
- replace c^* by $\tilde{c}^* = 2m_{+1}^2 m_{-1}^2 c^*$
  using $\tilde{\mathbf{x}}_0 := m_{-1}\mathbf{s}_{+1} + m_{+1}\mathbf{s}_{-1} = 2m_{+1}m_{-1}\mathbf{x}_0$
- get new score function

$$\tilde{f}^*(\mathbf{x}; \tilde{\mathbf{w}}^*, \tilde{c}^*) := 2m_{+1}m_{-1}\tilde{\mathbf{w}}^{*T}\mathbf{x} - \tilde{c}^* = 2m_{+1}^2 m_{-1}^2 f^*(\mathbf{x}; \mathbf{w}^*, c^*)$$

- result has the same sign as original score
- work with suitable multiples of the original values

In an integer world

Imagine you don't know real numbers, only integers...

- Represent all real data by integers
- normalize: shift mean to 0 and divide by standard deviation
- fix required precision
- move decimal point to the right, accordingly
- round to the nearest integer

$$[18.94, 21.31, 123.6, 1130, 0.09009, 0.1029, 0.108, 0.07951, 0.1582, 0.05461$$
$$\downarrow$$
$$[126, 43, 117, 133, -91, -39, -9, 41, -113, -123]$$

ML Confidential

The world of Somewhat Homomorphic Encryption (SHE)

- Can only use integer messages
  (polynomials with integer coefficients)
- can not divide
- can not compare
- multiplication is extremely expensive

But can do

- division-free
- integer
- low-degree polynomial

learning algorithms under SHE

(No FHE, because bootstrapping, modulus switching, key switching are too painful and maybe not really necessary)

Somewhat homomorphic encryption

(Fan, Vercauteren, 2012)

- Consider ring $R = \mathbb{Z}[x]/(f(x))$, $f(x) = x^d + 1$, $d = 2^k$
- Work in $R_q = R/qR$, q a power of 2
- Message space: $R_t = \mathbb{Z}_t[x]/(f(x))$, t a power of 2
- $\Delta = q/t$
- discrete Gaussian $\chi = D_{\mathbb{Z}^d,\sigma}$

$\mathbf{SH}.\mathbf{Keygen}$

- Sample small $s \leftarrow \chi$, secret key $\mathrm{sk} = s$.

Sample RLWE instance:

- Sample $a_1 \leftarrow R_q$ unif. rand., small error $e \leftarrow \chi$.

Public key

- $\mathrm{pk} = (a_0 = -(a_1 s + e), a_1)$.

Somewhat homomorphic encryption

(Fan, Vercauteren, 2012)

$\mathbf{SH.Enc}$

Given $\mathrm{pk} = (a_0, a_1)$ and a message $m \in R_q$,

- sample $u \leftarrow \chi$, and $f, g \leftarrow \chi$,

Set ciphertext

- $\mathrm{ct} = (c_0, c_1) := (a_0 u + g + \Delta m, a_1 u + f)$.

Somewhat homomorphic encryption

(Fan, Vercauteren, 2012)

$\mathbf{SH.Dec}$

Given $\mathrm{sk} = s$ and a ciphertext $\mathsf{ct} = (c_0, c_1)$,

- compute $\widetilde{m} = c_0 + c_1 s \in R_q$
- lift to integer coefficients, compute $\widetilde{m} \cdot t/q$
- round to nearest integer and reduce mod t

Correctness:

$$
\begin{aligned}
\widetilde{m} = c_0 + c_1 s &= (a_0 u + g + \Delta m) + (a_1 u + f)s \\
&= -(a_1 s + e)u + g + \Delta m + a_1 u s + f s \\
&= \Delta m + (g + f s - e u).
\end{aligned}
$$

Then $\widetilde{m} \cdot t/q = m + (g + fs - eu)t/q$, rounding gives back m.

Homomorphic operations

$\mathbf{SH.Add}$

Given $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{ct}' = (c_0', c_1')$, set the new ciphertext

- $\mathsf{ct}_{\mathsf{add}} = (c_0 + c_0', c_1 + c_1')$
  $= (a_0(u + u') + (g + g') + \Delta(m + m'), a_1(u + u') + (f + f'))$.

$\mathbf{SH.Mult}$

Given $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{ct}' = (c_0', c_1')$,

- compute
  $(c_0 + c_1 X)(c_0' + c_1' X) = c_0 c_0' + (c_0 c_1' + c_0' c_1)X + c_1 c_1' X^2$
  $= e_0 + e_1 X + e_2 X^2$
- $\mathsf{ct}_{\mathsf{mlt}} = (\lfloor t e_0 / q \rceil, \lfloor t e_1 / q \rceil, \lfloor t e_2 / q \rceil)$

Encoding integers

$$\begin{aligned}\mathrm{encode}: \mathbb{Z} \to R_t, \qquad z &= \mathrm{sign}(z)(z_s, z_{s-1}, \ldots, z_1, z_0)_2 \\ &\mapsto m_z = \mathrm{sign}(z)(z_0 + z_1 x + \ldots + z_s x^s) \bmod t.\end{aligned}$$

- Homomorphic properties w.r.t. R_t, i.e. mod t and $x^d + 1$
- avoid reduction mod t and mod $x^d + 1$ to ensure meaningful computations
- need t and d large enough (or integers small enough)
- $\mathrm{decode}: R_t \to \mathbb{Z},\ m(x) \mapsto m(2)$
- redundant representation
- $m_{11}(x) = 1 + x + x^3$, $m_{13}(x) = 1 + x^2 + x^3$, $(m_{11} + m_{13})(x) = 2 + x + x^2 + 2x^3$, $(m_{11} + m_{13})(2) = 24$

DFI-LM experiments

(P_1) $q = 2^{128}$, $t = 2^{15}$, $\sigma = 16$, $d = 4096$

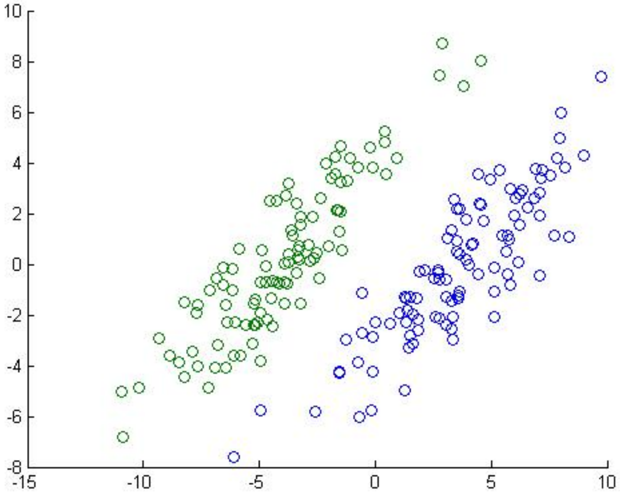
SH.Keygen	SH.Enc	SH.Dec(2)	SH.Dec(3)	SH.Add	SH.Mult
156	379	29	52	1	106

Timing in ms in Magma on a single core of an Intel Core i5 CPU650 @ 3.2 GHz. 128-bit security with distinguishing advantage 2^{-64}.

data	# features	algorithm	train	classify
surrogate	2	linear means	230	235
Iris	4	linear means	510	496

not measuring encryption, communication, decryption. “train”: time for training phase, i.e. to compute classifier from encrypted training data. “classify”: time for classifying a test vector.

Surrogate data set



Fisher's Linear Discriminant classifier

- Same score function as LM, but hyperplane takes into account class-conditional covariance
- change $\mathbf{w}^*$ to $\mathbf{w}^* = C^{-1}(\mathbf{m}_{+1} - \mathbf{m}_{-1})$, $C = C_{+1} + C_{-1}$

$$\mathbf{C}_{\pm 1} := \frac{1}{m_{\pm 1}} \sum_{x \in T_{\pm 1}} (\mathbf{x} - \mathbf{m}_{\pm 1})(\mathbf{x}_i - \mathbf{m}_{\pm 1})^T$$

- approximate $\mathbf{w}^*$ by gradient descent when minimizing $E(\mathbf{w}) := \frac{1}{2}||\mathbf{C}\mathbf{w} - (\mathbf{m}_{+1} - \mathbf{m}_{-1})||^2$
- gradient of E is $\nabla_{\mathbf{w}} E(\mathbf{w}) = \mathbf{C}\mathbf{w} - (\mathbf{m}_{+1} - \mathbf{m}_{-1})$
- iterate $\mathbf{w}_{j+1} = \mathbf{R}\mathbf{w}_j + \mathbf{a}$, $\mathbf{w}_0 = \mathbf{m}_{+1} - \mathbf{m}_{-1}$, where $\mathbf{R} := \mathbf{I} - \eta\mathbf{C}$, $\mathbf{a} := \eta(\mathbf{m}_{+1} - \mathbf{m}_{-1})$
- can get DFI version working with multiples, numbers grow quickly $\rightarrow t$ large $\rightarrow q$ large

DFI-FLD experiments

(P_2) $q = 2^{252}$, $t = 2^{35}$, $\sigma = 8$, $d = 8192$ (128-bit security)

SH.Keygen	SH.Enc	SH.Dec(2)	SH.Dec(3)	SH.Add	SH.Mult
382	853	98	193	4	370

(P_3) $q = 2^{340}$, $t = 2^{40}$, $\sigma = 8$, $d = 8192$ (80-bit security)

SH.Keygen	SH.Enc	SH.Dec(2)	SH.Dec(3)	SH.Add	SH.Mult
403	879	118	231	4	446

Surrogate data set, 2 features

algorithm	parameters	train	classify
1-step linear discriminant	(P_2)	58710	1490
2-step linear discriminant	(P_3)	74770	2680

All timings in ms.