Concentrated Differentially Private Gradient Descent with Adaptive per-Iteration Privacy Budget

Jaewoo Lee & Daniel Kifer

KDD 2018

Presented by Christian McDaniel

Introduction

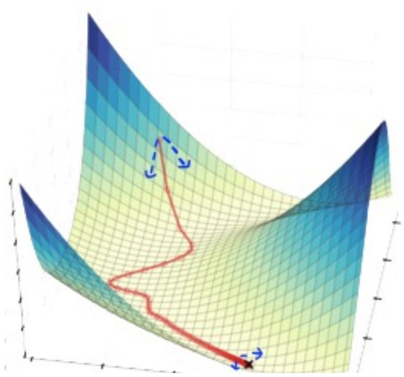
Introduction

Iterative optimization algorithm	Differentially Private iterative algorithms
+ min f by finding optimal paramter vector w* 1) Initialize w_0 2) Generate {w_t} s.t. w_t tends toward optimal w as t --> infinity e.g., Gradient Descent $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t(\nabla f(\mathbf{w}_t))$	Gradient Descent under Differential Privacy $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t(\nabla f(\mathbf{w}_t) + Y_t)$ Prior efforts: Naive conversion 1) Pick T a priori 2) Split the budget evenly $\epsilon_1 = \cdots = \epsilon_T = \epsilon/T.$ 3) Update parameters using a noisy graident

Introduction

Naive Privacy conversion has a few issues:

- Accuracy depends on T
  - Too Small v. Too Large
- The nature of the gradient changes as the optimum is approached
  - Early v. Near Optimum



Introduction

Naive Privacy conversion has a few issues:

- Accuracy depends on T
  - Too Small v. Too Large
- The nature of the gradient changes as the optimum is approached
  - Early v. Near

Authors Propose:

- (First known) Adaptive Privacy Budget for zero-mean Concentrated Differential Privacy (more later)
  - re: Larger v. Smaller norm gradients
  - Allows an adaptive T

Introduction

Naive Privacy conversion has a few issues:

- Accuracy depends on T
  - Too Small v Too Large
- The nature of the gradient changes as the optimum is approached
  - Early v. Near

Authors Propose:

- Adaptive Privacy Budget for zCDP
  - At each iteration:
    - Partial allocation to calculating the noisy gradient:

$$\tilde{S}_t = \nabla f(\mathbf{w}_t) + Y_t$$

Authors Propose:

- Adaptive Privacy Budget for zCDP
    - At each iteration:
        - Partial allocation to calculating the noisy gradient
        - Allocate the rest to determining the step size:
            - Minimize

$$f(\mathbf{w}_t - \alpha \tilde{S}_t)$$

            - Update accordingly:

Background

Background

Differential Privacy

Definition 3.1 ((ϵ, δ)-DP [7, 8]). A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for every event $S \subseteq \text{range}(\mathcal{M})$ and for all $D \sim D' \in \mathcal{D}^n$,

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in S] + \delta\,.$$

- When delta=0, said to be pure-DP

Definition 3.2 (L_1 and L_2 sensitivity). Let $q : \mathcal{D}^n \to \mathbb{R}^d$ be a query function. The L_1 (resp. L_2) sensitivity of q, denoted by $\Delta_1(q)$ (resp., $\Delta_2(q)$) is defined as

Background

Gaussian Mechanism

THEOREM 3.3 (GAUSSIAN MECHANISM [9]). *Let $\epsilon \in (0, 1)$ be arbitrary and q be a query function with L_2 sensitivity of $\Delta_2(q)$. The Gaussian Mechanism, which returns $q(D) + N(0, \sigma^2)$, with*

$$\sigma \geq \frac{\Delta_2(q)}{\epsilon}\sqrt{2\ln(1.25/\delta)} \tag{3}$$

is (ϵ, δ)-differentially private.

Background

Concentrated DP

ρ-zCDP,

- for output o in range(M), the privacy loss random variable Z of M is

$$Z = \log \frac{\Pr[\mathcal{M}(D) = o]}{\Pr[\mathcal{M}(D') = o]}$$

- p-zCDP imposes a bound on the moment generating function of Z and requires it to be bound around 0
- Formally it satisfies:

$$e^{D_\alpha(\mathcal{M}(D)||\mathcal{M}(D'))} = \mathbb{E}\left[e^{(\alpha-1)Z}\right] \leq e^{(\alpha-1)\alpha\rho}, \ \forall\alpha \in (1, \infty)$$

Background

Concentrated DP Lemmas

- Composition

LEMMA 3.5 ([4]). *Suppose two mechanisms satisfy ρ_1-zCDP and ρ_2-zCDP, then their composition satisfies $(\rho_1 + \rho_2)$-zCDP.*

Background

Concentrated DP Lemmas

- zCDP Gaussian Mechanism

LEMMA 3.6 ([4]). *The Gaussian mechanism, which returns $q(D) + N(0, \sigma^2)$ satisfies $\Delta_2(q)^2/(2\sigma^2)$-zCDP.*

Background

Concentrated DP Lemmas

- zCDP - pure-DP conversion

LEMMA 3.7 ([4]). *If $\mathcal{M}$ satisfies ϵ-differential privacy, them $\mathcal{M}$ satisfies $(\frac{1}{2}\epsilon^2)$-zCDP.*

Background

Concentrated DP Lemmas

- zCDP - (e,d)-DP

Background

NoisyMax

Algorithm 1: NoisyMax($\Omega, \Delta_1(f), \epsilon$)

Input: Ω: a set of candidates, $\Delta_1(f)$: sensitivity of f, ϵ: privacy budget for pure differential privacy

$\widetilde{\Omega} = \{\tilde{v}_i = v + \mathrm{Lap}(\Delta_1(f)/\epsilon) : v \in \Omega, i \in [|\Omega|]\}$

return $\arg\max_{j \in [|\Omega|]} \tilde{v}_j$

NoisyMin = NoisyMax on - f(w)

NoisyMax

Implementation

Φ

- Initialize Phi with m equally-spaced pts b/t 0 and a_max
- Proposed to update a_max in Phi every r iters

$$\alpha_{\max} = (1+\eta)\max(\alpha_t, \alpha_{t-1}, \ldots, \alpha_{t-\tau+1}),$$

- Set m=20, r=10, n_=0.1,a_max=2

The Algorithm

DP-AGD

The Algorithm

Three Main Parts:

1. Calculate the Noisy Gradient
2. Determine the best step size
3. Adjust the privacy budget (if needed)

The Algorithm

- Consider ERM

$$\underset{\mathbf{w}\in C}{\text{minimize}}\ f(\mathbf{w};D) := \frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{w};d_i)$$

The Algorithm

Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$

- p_nmax: privacy budget in terms of p-zCDP
- p_ng: initial privacy budget for the gradient approximation
- E_tot, delta_tot: the budget parameters re (E,delta)-DP
- gamma: budget increase rate
- C_obj and C_grad: clipping thresholds for gradient calc and obj fnxn
- {d_1,...,d_n}: set of n observations where each d_i = (x_i, y_i)
- objective function: ERM

The Algorithm

Algorithm 2: DP-AGD

Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$

Initialize $\mathbf{w}_0$ and Φ

The Algorithm

Algorithm 2: DP-AGD

Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$

Initialize $\mathbf{w}_0$ and Φ

$t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to (`ϵ, δ`)-DP Algs`

$$\epsilon_{\text{tot}} \geq \rho + 2\sqrt{\rho \log(1/\delta_{\text{tot}})}\,.$$

(Lemma 3.8)

The Algorithm

Algorithm 2: DP-AGD

Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$

Initialize $\mathbf{w}_0$ and Φ

$t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to (ε, δ)-DP Algs`

while $\rho > 0$ **do**

- Checking use of privacy budget

Three Main Parts:

1. Calculate the Noisy Gradient
2. Determine the best step size
3. Adjust the privacy budget (if needed)

The Algorithm

1) Private Gradient Approximation

–	–
Algorithm 2: DP-AGD **Input:** privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$ Initialize $\mathbf{w}_0$ and Φ $t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs` **while** $\rho > 0$ **do**	+ Gradient Clipping technique (instead of assuming norm(x) <= 1) + variance of noise depends on

The Algorithm

1) Private Gradient Approximation

–	–
Algorithm 2: DP-AGD **Input:** privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$ Initialize $\mathbf{w}_0$ and Φ $t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs` **while** $\rho > 0$ **do** $i \leftarrow 0$ $\mathbf{g}_t \leftarrow \sum_{i=1}^{n} \left(\nabla \ell(\mathbf{w}_t; d_i) / \max(1, \frac{\lVert \nabla \ell(\mathbf{w}_t) \rVert_2}{C_{\text{grad}}}) \right)$ $\widetilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + N(0, (C_{\text{grad}}^2 / 2\rho_{\text{ng}})\mathbf{I})$	+ Add the noise

The Algorithm

1) Private Gradient Approximation

–	–

Algorithm 2: DP-AGD
Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$
Initialize $\mathbf{w}_0$ and Φ
$t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs`
while $\rho > 0$ **do**
    $i \leftarrow 0$
    $\mathbf{g}_t \leftarrow \sum_{i=1}^{n} \left(\nabla\ell(\mathbf{w}_t; d_i)/\max(1, \frac{\lVert\nabla\ell(\mathbf{w}_t)\rVert_2}{C_{\text{grad}}})\right)$
    $\widetilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + N(0, (C_{\text{grad}}^2/2\rho_{\text{ng}})\mathbf{I})$
    $\rho \leftarrow \rho - \rho_{\text{ng}}$
	+ Adjust the budget

The Algorithm

1) Private Gradient Approximation

The Algorithm

2) Step Size Selection

–	–

Algorithm 2: DP-AGD
Input: privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \dots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$
Initialize $\mathbf{w}_0$ and Φ
$t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs`
while $\rho > 0$ **do**
  $i \leftarrow 0$
  $\mathbf{g}_t \leftarrow \sum_{i=1}^{n} \left(\nabla \ell(\mathbf{w}_t; d_i) / \max(1, \frac{\lVert \nabla \ell(\mathbf{w}_t) \rVert_2}{C_{\text{grad}}})\right)$
  $\widetilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + N(0, (C_{\text{grad}}^2 / 2\rho_{\text{ng}})\mathbf{I})$
  $\rho \leftarrow \rho - \rho_{\text{ng}}$
  $\widetilde{\mathbf{g}}_t \leftarrow \widetilde{\mathbf{g}}_t / \lVert \widetilde{\mathbf{g}}_t \rVert_2$
  while $i = 0$ **do**
    $\Omega = \{f(\mathbf{w}_t - \alpha \widetilde{\mathbf{g}}_t) : \alpha \in \Phi\}$
    $\rho \leftarrow \rho - \rho_{\text{nmax}}$
    $i \leftarrow$ NoisyMax($-\Omega, C_{\text{obj}}, \sqrt{2\rho_{\text{nmax}}}$)
	+ Find optimal step size
	
	+ NoisyMax(set of candidates, sensitivity, privacy budget for pure DP - Lemma 3.7)
	
	+ Control the variance of the updates

The Algorithm

2) Step Size Selection

–	–
Algorithm 2: DP-AGD **Input:** privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \dots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$ 1 Initialize $\mathbf{w}_0$ and Φ 2 $t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs` 3 **while** $\rho > 0$ **do** 4 $i \leftarrow 0$ 5 $\mathbf{g}_t \leftarrow \sum_{i=1}^{n} \left(\nabla\ell(\mathbf{w}_t; d_i)/\max(1, \frac{\\|\nabla\ell(\mathbf{w}_t)\\|_2}{C_{\text{grad}}})\right)$ 6 $\widetilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + N(0, (C_{\text{grad}}^2/2\rho_{\text{ng}})\mathbf{I})$ 7 $\rho \leftarrow \rho - \rho_{\text{ng}}$ 8 $\widetilde{\mathbf{g}}_t \leftarrow \widetilde{\mathbf{g}}_t/\\|\widetilde{\mathbf{g}}_t\\|_2$ 9 **while** $i = 0$ **do**	Step 14: Check use of privacy budget before

The Algorithm

3) Adoptive Noise Reduction - Gradient Averaging

–	–
15 **else** 16 $\rho_{\text{old}} \leftarrow \rho_{\text{ng}}$ 17 $\rho_{\text{ng}} \leftarrow (1+\gamma)\rho_{\text{ng}}$ 18 $\widetilde{\mathbf{g}}_t \leftarrow \text{GradAvg}(\rho_{\text{old}}, \rho_{\text{ng}}, \mathbf{g}_t, \tilde{\mathbf{g}}_t, C_{\text{grad}})$ 19 $\rho \leftarrow \rho - (\rho_{\text{ng}} - \rho_{\text{old}})$ 25 **Function** GradAvg($\rho_{\text{old}}, \rho_H, \mathbf{g}, \tilde{\mathbf{g}}, C_{\text{grad}}$): 26 $\tilde{\mathbf{g}}_2 \leftarrow \mathbf{g} + N(\mathbf{0}, (\frac{C_{\text{grad}}^2}{2(\rho_H - \rho_{\text{old}})})\mathbf{I})$ 27 $\tilde{S} \leftarrow \frac{\rho_{\text{old}}\tilde{g} + (\rho_H - \rho_{\text{old}})\tilde{g}_2}{\rho_H}$ 28 **return** $\tilde{S}$ 29 **end**	+ Increase p_t to p_t+1 + Only use p_t+1 - p_t + Average the gradient from steps t and t+1 to only use a total of p_t+1 across both

The Algorithm

–	–
Algorithm 2: DP-AGD **Input:** privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$ 1 Initialize $\mathbf{w}_0$ and Φ 2 $t \leftarrow 0, \rho \leftarrow$ solve (5) for ρ `// To compare to` (ϵ, δ)`-DP Algs` 3 **while** $\rho > 0$ **do** 4 $i \leftarrow 0$ 5 $\mathbf{g}_t \leftarrow \sum_{i=1}^{n} \left(\nabla \ell(\mathbf{w}_t; d_i) / \max(1, \frac{\lVert \nabla \ell(\mathbf{w}_t) \rVert_2}{C_{\text{grad}}}) \right)$ 6 $\widetilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + N(0, (C_{\text{grad}}^2 / 2\rho_{\text{ng}})\mathbf{I})$ 7 $\rho \leftarrow \rho - \rho_{\text{ng}}$ 8 $\widetilde{\mathbf{g}}_t \leftarrow \widetilde{\mathbf{g}}_t / \lVert \widetilde{\mathbf{g}}_t \rVert_2$ 9 **while** $i = 0$ **do** 10 $\Omega = \{ f(\mathbf{w}_t - \alpha \widetilde{\mathbf{g}}_t) : \alpha \in \Phi \}$ 11 $\rho \leftarrow \rho - \rho_{\text{nmax}}$ 12 $i \leftarrow$ NoisyMax($-\Omega, C_{\text{obj}}, \sqrt{2\rho_{\text{nmax}}}$) 13 **if** $i > 0$ **then** 14 **if** $\rho > 0$ **then** $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha_i \widetilde{\mathbf{g}}_t$ 15 **else** 16 $\rho_{\text{old}} \leftarrow \rho_{\text{ng}}$ 17 $\rho_{\text{ng}} \leftarrow (1 + \gamma)\rho_{\text{ng}}$ 18 $\widetilde{\mathbf{g}}_t \leftarrow$ GradAvg($\rho_{\text{old}}, \rho_{\text{ng}}, \mathbf{g}_t, \tilde{\mathbf{g}}_t, C_{\text{grad}}$) 19 $\rho \leftarrow \rho - (\rho_{\text{ng}} - \rho_{\text{old}})$ 20 **end** 21 **end** 22 $t \leftarrow t + 1$	+ loop, dynamically computing budget p (lines 19, 11, 7) + thanks to composition propterties

The Algorithm

-	-
Algorithm 2: DP-AGD **Input:** privacy budget $\rho_{\text{nmax}}, \rho_{\text{ng}}, \epsilon_{\text{tot}}, \delta_{\text{tot}}$, budget increase rate γ, clipping thresholds $C_{\text{obj}}, C_{\text{grad}}$, data $\{d_1, \ldots, d_n\}$, objective function $f(\mathbf{w}) = \sum_{i=1}^{n} \ell(\mathbf{w}; d_i)$	

```python
def agd_rho(X, y, rho, eps_total, delta, grad_func, loss_func, test_func,
            obj_clip, grad_clip, reg_coeff=0.0, batch_size=-1, exp_dec=1.0,
            gamma=0.1, splits=60, verbose=False):
    N, dim = X.shape

    # parameters
    eps_nmax = (eps_total * 0.5) / splits
    sigma = compute_sigma(eps_nmax, delta, grad_clip)

    # intial privacy budget
    rho_nmax = 0.5 * (eps_nmax**2)
    rho_ng = (grad_clip**2) / (2.0 * sigma**2)
    # rho_ng = rho_nmax

    # initialize the parameter vector
    w = np.zeros(dim)
    t = 0
    chosen_step_sizes = []
    max_step_size = 2.0
    n_candidate = 20
```

```
def agd_rho():
    ...
while rho > 0:
        if verbose:
            loss = loss_func(w, X, y) / N
            acc = test_func(w, X, y)
            print "[{}] loss: {:.5f}  acc: {:5.2f}".format(t, loss, acc*100)
        if batch_size > 0:
            # build a mini-batch
            rand_idx = np.random.choice(N, size=batch_size, replace=False)
            mini_X = X[rand_idx, :]
            mini_y = y[rand_idx]
        else:
            mini_X = X
            mini_y = y

        # non-private (clipped) gradient
        grad = grad_func(w, mini_X, mini_y, grad_clip)

        sigma = grad_clip / math.sqrt(2.0 * rho_ng)
        noisy_grad = grad + sigma * np.random.randn(dim)
        noisy_unnorm = np.copy(noisy_grad)
        noisy_grad /= np.linalg.norm(noisy_grad)

        rho -= rho ng
```

```
def agd_rho():
    ...
    while rho > 0:
        ...
        while idx == 0:
                    # test if this is a descent direction
                    step_sizes = np.linspace(0, max_step_size, n_candidate+1)
                    candidate = [w - step * noisy_grad for step in step_sizes]
                    scores = [loss_func(theta, mini_X, mini_y, clip=obj_clip)
                              for theta in candidate]
                    scores[0] *= exp_dec

                    # deduct the privacy budget used for noisy max
                    lmbda = obj_clip/math.sqrt(2.0 * rho_nmax)
                    idx, _ = noisy_max(scores, lmbda, bmin=True)
                    rho -= rho_nmax
```

```python
def agd_rho():
    ...
    while rho > 0:
        ...
        while idx == 0:
            ...
            # used up the budget
            if rho < 0:
                break

            if idx > 0:
                # don't do the update when the remain budget is insufficient
                if rho >= 0:
                    w[:] = candidate[idx-1]
                rho -= rho_ng
            else:
                rho_old = rho_ng
                rho_ng *= (1.0 + gamma)
                # max_step_size *= 0.9
                if verbose:
                    print "\tbad gradient: resample (rho_ng={})".format(rho_ng)

                noisy_grad = grad_avg(rho_old, rho_ng, grad, noisy_unnorm,
                                      grad_clip)
                noisy_grad /= np.linalg.norm(noisy_grad)
                rho -= (rho_ng - rho_old)

                if reg_coeff > 0:
                    noisy_grad += reg_coeff * w
```

```python
def agd_rho():
    ...
    while rho > 0:
        ...
        while idx == 0:
            ...
            chosen_step_sizes.append(step_sizes[idx])
            t += 1

            if (t % 10) == 0:
                max_step_size = min(1.1*max(chosen_step_sizes), 2.0)
                del chosen_step_sizes[:]
```

```python
def grad_avg(rho_old, rho_H, true_grad, noisy_grad, grad_clip):
    dim = true_grad.shape[0]
    sigma = grad_clip / math.sqrt(2.0 * (rho_H - rho_old))

    # new estimate
    g_2 = true_grad + sigma * np.random.randn(dim)

    beta = rho_old / rho_H
    # weighted average
    s_tilde = beta * noisy_grad + (1.0 - beta) * g_2

    return s_tilde
```

Experimental Results

Experimental Results

Experimental Results

Baselines:

`ObjPert`: Objective Perturbation, add linear perturbation term to the objective function and solve using non-private solver

`OutPert`: Output Perturbation, perturb the output after batch GD for predetermined steps

`PrivGene`: DP genetic algorithm

`SGD-Adv`: DP algorithm using advanced composition wwith privacy amplification output (upper bound)

`SGD-MA`: Uses an improved composition method called moments accountant

`NonPrivate`: Non-private L-BFGS algorithm

`Majority`: Chooses majority class

- All known DP algorithms use a predetermined privacy budget sequence

Experimental Results

Experiment Design

- Report the classification accuracy and final objective value
- Perform 20 repeats of 5-fold CV

Preprocessing:

- one-hot encoding of all binary variables
- rescale numberical variables on [0,1]
- for ObjPert, normalie to unit norm (per requirements)

Experimental Results

Effects of Parameters

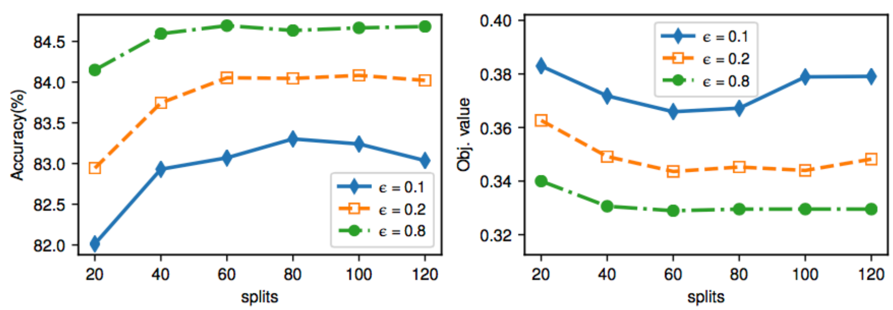
- Ran DP-AGD on Adult dataset
- Overall: robust to parameters - moderate settings have small effect

Experimental Results

Effects of Parameters (DP-AGD on Adult)

- Splits (T) : Estimate initial privacy budget via number of splits

$$\epsilon_{\text{nmax}} = \epsilon_{\text{ng}} = \frac{\epsilon_{\text{tot}}}{2 \cdot \text{splits}}$$



(a) Effect of splits (left: accuracy, right: obj. value)

- Less effect when budget is large
- Too small v Too large

Experimental Results

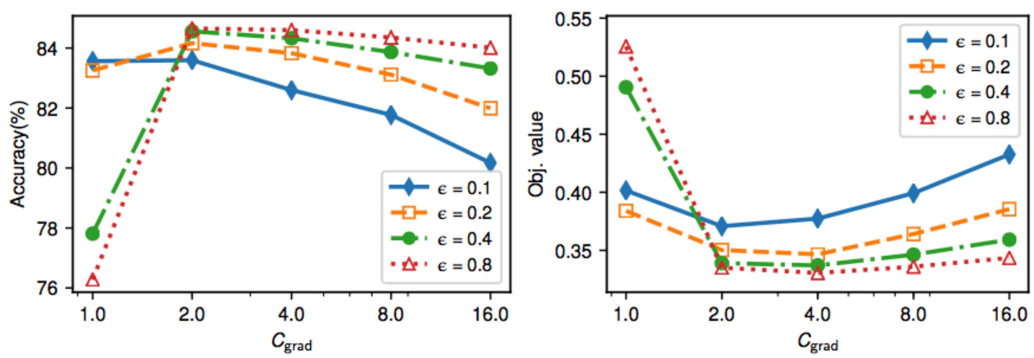
Effects of Parameters (DP-AGD on Adult)

- Gamma

Experimental Results

Effects of Parameters (DP-AGD on Adult)

- C_grad



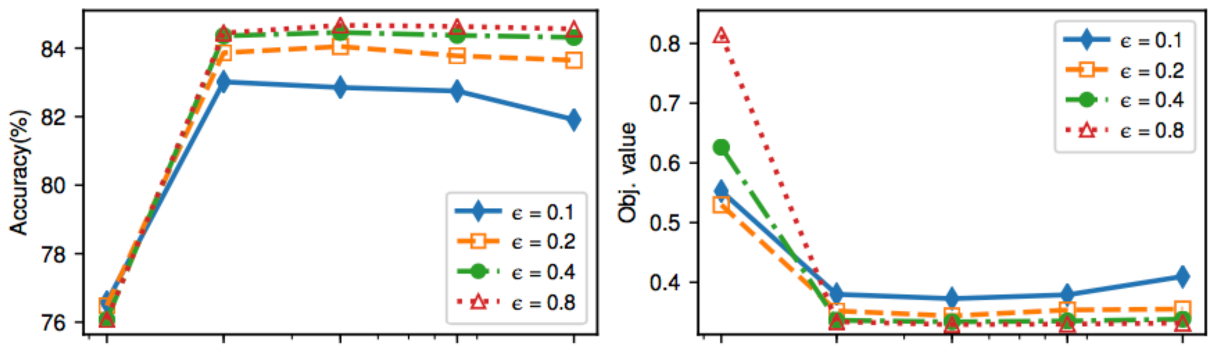
(c) Effect of C_{grad} (left: accuracy, right: obj. value)

- C_x are the clipping parameters : sets a bound on the sensitivity to satisfy zCDP
- C_grad refers to the gradient calculation
- Too small: low sensitivity, but can cause too much information loss
- Too large: high sensitivity, results in too much noise

Experimental Results

Effects of Parameters (DP-AGD on Adult)

- C_obj



Experimental Results

Logistic Regression

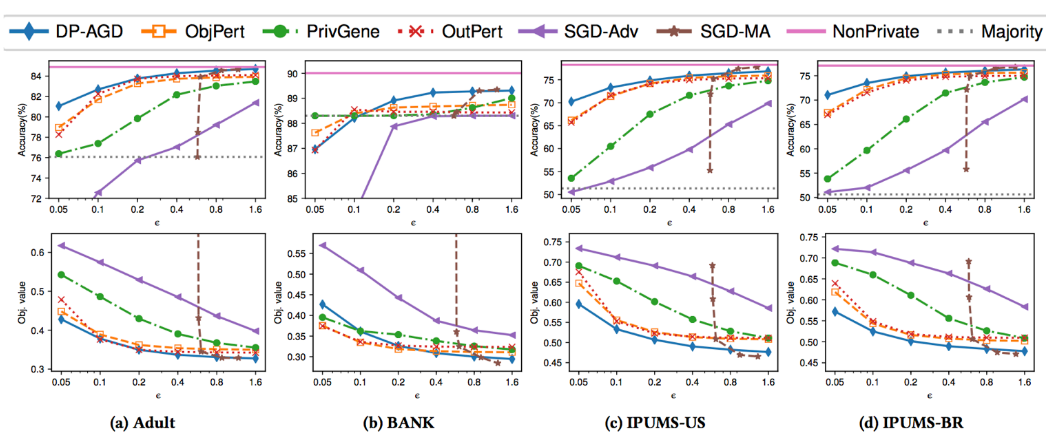


Figure 2: Logistic regression by varying ϵ (Top: classification accuracies, Bottom: objective values)

–	–
+ DP-AGD outperforms / performs competitively + especially when budget is very small	+ SGD-Adv and SGD-MA depend on T (predetermined) + SGD-MA outperforms all others when budget > 0.8 (more splits due to tight bound on privacy loss) --> impractical under strict privacy control

Experimental Results

SVM

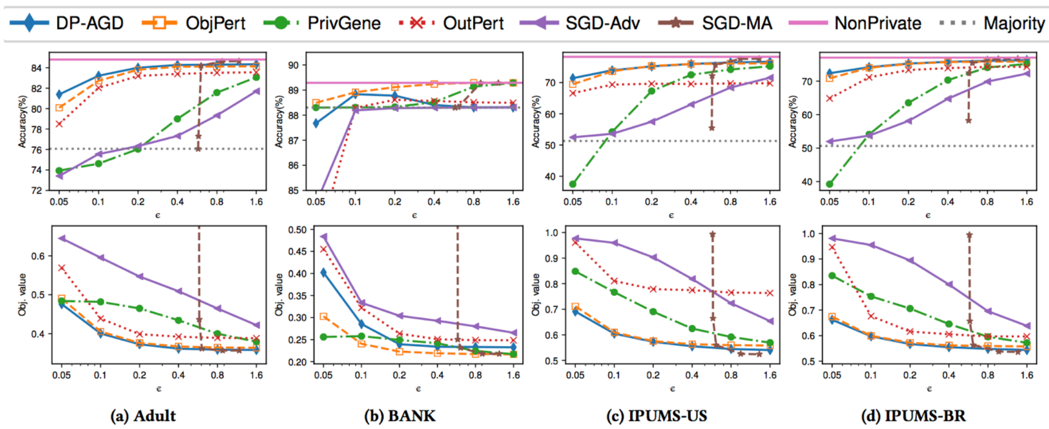
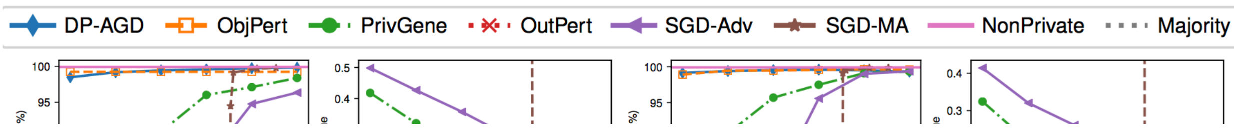


Figure 3: SVM by varying ϵ (Top: classification accuracies, Bottom: objective values)

- DP-AGD scores competitive accuracies
  - shows unstable behavior on BANK (acc. decreases despite increased use of budget)
  - Still, objective value consistently decreases as the budget is increased

Experimental Results

KDDCup99



Conclusions:

- First known iterative optimization algorithm for differential privacy
- satisfies zCDP, which is weaker than (e)-DP, but stronger than (e,d)-DP
- Per-iteration privacy budget is adaptively determined during runtime based on utility of privacy-preserving statistics
- Uses gradient averaging to utilize non-descending gradients and preserve the budget
- Outperforms or performs competitively with baselines, with an advantage under strict privacy settings
- Code available: https://github.com/ppmlguy/DP-AGD (https://github.com/ppmlguy/DP-AGD)
- Citation:

```
@inproceedings{Lee2018ConcentratedDP,
title={Concentrated Differentially Private Gradient Descent with Adapti
ve per-Iteration Privacy Budget},
author={Jaewoo Lee and Daniel Kifer},
booktitle={KDD},
year={2018}
}
```

  - Lee, J., & Kifer, D. (2018). Concentrated Differentially Private Gradient Descent with Adaptive per-Iteration Privacy Budget. KDD.

Questions?