CS 49/149: 21st Century Algorithms (Fall 2018): Problems

I will be posting 2-3 problems every week (so I presume there will be around 20 problems in all). You have to submit answers to **10 problems** out of these. There will be 3 dates on Canvas where you will be submitting 3,4, and 3 answers respectively. You are of course free and indeed encouraged to try *all* these problems. Please submit those problems which you think you have solved it to your satisfaction, and give clear and concise answers.

Problem 1. ☕☕

In the experts problem, prove that no deterministic algorithm can do better than a 2-factor in the number of mistakes than the best expert. That is, for any deterministic algorithm $\mathcal{A}$, any parameter δ and any T, there is a setting of the experts problem such that the best expert makes $\leq \delta T$ mistakes, while $\mathcal{A}$ makes $\geq 2\delta T$ mistakes.

Problem 2. ☕☕

Suppose we changed the MWU algorithm as follows: after observing the losses $\ell_i(t)$ at time t, instead of updating the weights as $w_i(t+1) = w_i(t) \cdot (1 - \eta \ell_i(t))$ suppose you updated as

$$w_i(t+1) = w_i(t) \cdot e^{-\eta \ell_i(t)}$$

Analyze this algorithm and figure out the error term. Which one is better and under which circumstances?

Problem 3. ☕☕☕

In class we saw that the average regret of the MWU algorithm after T time steps was $O\left(\sqrt{\frac{\ln m}{T}}\right)$. Show that you can't get better. That is, show for any algorithm (even randomized), the average regret has to be $\Omega\left(\sqrt{\frac{\ln m}{T}}\right)$. To begin with, give an $\Omega(1/\sqrt{T})$ bound for 2 experts.

Hint: The following fact may be useful: if $X_1, \ldots, X_n$ are n independent, iid random variables taking values in $\{-1, +1\}$ with probability $1/2$ each, and if $S = |\sum_{i=1}^n X_i|$, then $\mathbf{Exp}[S] = \Theta(\sqrt{n})$.

Problem 4. ☕☕☕

In class when we wanted the average regret to be $O(\sqrt{\ln m/T})$, we set η to be something which depended on T. What is you didn't know T? That is, the online decision could stop at any *unknown* time T and you would still like the average regret to be $O(\sqrt{\ln m/T})$ (for all T). How would you modify the algorithm?

Hint: Keep a conservative "guess" of T and set η likewise; if the actual T is more than your guess, then bump η down.

Problem 5. ☕☕

In class, we saw how to approximate solve the LP

$$\min \sum_{j=1}^{n} c_j x_j \; : \; Ax \geq b; \quad 0 \leq x_j \leq 1$$

The width parameter was defined to be $\rho := \max_{i=1}^m |a_i^\top x - b_i|$ for any x returned by the oracle. We showed that for any $\varepsilon > 0$, the MWU algorithm can be used to return an $\overline{x}$ such that for every constraint $i \in [m]$, we get $a_i^\top \overline{x} \geq b_i - \varepsilon$. The number of iterations was $O\left(\frac{\rho^2 \ln m}{\varepsilon^2}\right)$.

In this exercise suppose $A, b \geq 0$, that is, all entries of the constraints are non-negative. Then, by dividing the ith row by b_i we can assume every $b_i = 1$. Prove that the MWU algorithm actually needs to only run $O\left(\frac{\rho \ln m}{\varepsilon^2}\right)$ many iterations.

Hint: Recall the loss function we defined in class. Instead of just using $|\ell_i(t)^2| \leq 1$, perhaps use that $|\ell_i(t)^2| \leq |\ell_i(t)|$.

Problem 6.

For each of these functions $f : \mathbb{R}^n \to \mathbb{R}$ below, calculate $\nabla f(x)$.

a. $f(x) = b^\top Ax$ for some $b \in \mathbb{R}^m$ and some $m \times n$ matrix A.

b. $f(x) = \frac{1}{2}x^\top Qx$ for some $n \times n$ matrix Q.

c. $f(x) = ||Ax - b||_2^2$ where A is an $m \times n$ matrix, and $b \in \mathbb{R}^m$.

Problem 7.

Prove the following about convex functions.

a. The function $x \mapsto \ln x$ is concave (that is, $-\ln x$ is convex). Use the definition of convexity to deduce the AM-GM inequality, that is, for any $x_1, \ldots, x_m > 0$,

$$\left(\prod_{i=1}^{m} x_i\right)^{1/m} \leq \frac{\sum_{i=1}^{m} x_i}{m}$$

b. Given a collection (a_t, b_t) for $t = 1, 2, \ldots, T$, where each $a_t \in \mathbb{R}^n$ and $b_t \in \mathbb{R}$, prove that

$$f(x) := \max_{t=1}^{T}(a_t^\top x + b_t)$$

is convex. Use this to deduce that the following function, $x \mapsto x^{[\ell]}$ where the RHS is the sum of the ℓ largest entries of x is a convex function.

c. Prove that if $h : \mathbb{R} \to \mathbb{R}$ is convex and increasing, and $g : \mathbb{R}^n \to \mathbb{R}$ is a convex function, then $f(x) := h(g(x))$ is a convex function.

d. (Extra Credit)

Prove that

$$f(x) := \log\left(\sum_{i=1}^{n} e^{x_i}\right)$$

is convex.

Hint: Use the Hessian

Problem 8.

a. Consider unconstrained convex optimization $\min_{x \in \mathbb{R}^n} f(x)$ where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex, differentiable function. Let x_* be global minimizer of f. Prove that $\nabla f(x_*)$ is the 0-vector. In particular, given any x with $\nabla f(x) \neq 0$, demonstrate a vector y with $f(y) < f(x)$.

b. Now consider constrained convex optimization $\min_{x \in S} f(x)$ where S is a convex set. Agian, let $x_* \in S$ be a global minimizer. Is $\nabla f(x_*) = 0$ a necessary condition? What is a necessary condition you can assert for $\nabla f(x_*)$ and points in S. Prove your statement.

Problem 9.

In class, we showed that for any $\varepsilon > 0$, if we set $\eta = \varepsilon^2/\rho$ and we run for $T = D^2\rho^2/\varepsilon^2$ rounds, then we can find a point x_t with $f(x_t) \le f(x_*) + \varepsilon$. Here $D := ||x_1 - x_*||_2$ and $||\nabla f(x)||_2 \le \rho$ for all x.

In this exercise, you take a "time-dependent-parameter-independent" step size. That is, η_t is not a fixed η but is defined to be $\eta_t := \frac{1}{\sqrt{t}}$. Formally, we start at point x_1 and then at time t, we take the step

$$x_{t+1} = x_t - \frac{1}{\sqrt{t}} \nabla f(x_t)$$

Analyze the above algorithm. Your goal is to find out given ε how long do we need to run to get some $f(x_t) \le f(x_*) + \varepsilon$. The running time will depend on D, ρ; your goal is to find out if it is better or worse than the fixed parameter-dependent step size.

Problem 10.

Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, prove that $||\nabla f(x)||_2 \le \rho$ for all x *if and only if* $|f(x) - f(y)| \le \rho \cdot ||x - y||_2$ for all x, y.

Problem 11.

Consider $f : \mathbb{R}^n \to \mathbb{R}$ which is L-smooth and convex. In class, we proved that vanilla gradient descent with $\eta = 1/L$ if run for $T := LD^2/\varepsilon$ iterations, where $D := ||x_1 - x_*||_2$, one gets a point $f(x) \le f(x_*) + \varepsilon$.

In this exercise, you are supposed to extend it to the case of constrained convex optimization with projected gradient descent. In particular, the algorithm is

$$z_{t+1} = x_t - \eta \nabla f(x_t); \qquad x_{t+1} = \Pi_S(z_{t+1}) = \arg\min_{v \in S} ||z_{t+1} - v||_2$$

Prove that the *same* convergence bound holds. Indeed, there is only one extra line in the whole analysis.
Hint: Recall the fact about projection: if $u \in S$, $v \notin S$, and $p := \Pi_S(v)$ is the projection of v onto S, then $(u - p)^\top (v - p) \le 0$. That is, the line joining v and p, and the line joining u and p make an obtuse angle.

Problem 12.

In this exercise, you will analyze gradient descent assuming *only* strong convexity of f. Recall that $f : \mathbb{R}^n \to \mathbb{R}$ is ℓ-strongly convex iff for all x, y we have

$$f(y) \ge f(x) + (y - x)^\top \nabla f(x) + \frac{\ell}{2} \cdot ||y - x||_2^2$$

Assume that $||\nabla f(x)||_2 \le \rho$ for all x (note that f may not be smooth).

Consider gradient descent with *time-dependent step-size* $\eta_t := \frac{1}{\ell t}$. Prove that if we run this for T iterations, and if $\mathsf{err}(t) := f(x_t) - f(x_*)$, then

$$\frac{1}{T} \cdot \sum_{t=1}^{T} \mathsf{err}(t) \le \frac{\rho^2 \ln T}{\ell T}$$

In particular, for any $\varepsilon > 0$ this shows that running for $T = \frac{2\rho^2}{\ell} \cdot \left(\frac{1}{\varepsilon} \ln \frac{4\rho^2}{\ell\varepsilon}\right)$ rounds would give error $\le \varepsilon$.
Hint: Use the fact that for strongly convex functions we have a better upper bound on $\mathsf{err}(t)$. In particular,

$$\mathsf{err}(t) \le (x_t - x_*)^\top \nabla f(x_t) - \frac{\ell}{2} D_t^2$$

Now use the fact that the η's are decaying to show that when you sum up the $\mathsf{err}(t)$'s cancellations occur to give you what you need. It may be useful to recall the fact that $1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdot + 1/k \;\; \ln k$.

(Extra Credit): In fact, you can also get a $O(1/\varepsilon)$-convergence rate (assuming ℓ, ρ are constant) with some slight modification. If you have had fun solving the problem above, then maybe you can try this too.

Problem 13. ☕(This was done in class – hence the single coffeecup. Please **do not** consult your handwritten notes while writing this solution up.)

Let $\Phi : \mathbb{R}^n \to \mathbb{R}$ be any strictly convex function, and S be any convex set S. Let z be an arbitrary point outside S, and let x be an arbitrary point inside S. Define

$$p := \arg\min_{v \in S} D_\Phi(v, z)$$

Prove $D_\Phi(x, z) \geq D_\Phi(x, p)$.

Problem 14. ☕☕(This is an instructional and informational exercise – highly recommended!)

Recall the four steps of Mirror Descent from a given point x_t:

a. $y_t := \nabla\Phi(x_t)$

b. $y_{t+1} := y_t - \eta\nabla f(x_t)$

c. $z_{t+1} := \nabla\Phi^{-1}(y_{t+1})$

d. $x_{t+1} := \arg\min_{p \in S} D_\Phi(p, z_{t+1})$.

Prove that the the y, z's can be "eliminated" to give the following "one-liner" connecting x_t and x_{t+1}.

$$x_{t+1} = \arg\min_{p \in S} \left(\eta p^\top \nabla f(x_t) + D_\Phi(p, x_t)\right)$$

Hint: There are no inequalities in the above exercise ... keep writing the definitions and eliminating first z_{t+1}, then y_{t+1} and then y_t. Recall the slick trick when computing $\arg\min$ – if we add a constant to the function we are taking $\arg\min$ of, the $\arg\min$ doesn't change.

Problem 15. ☕☕(This may need some familiarity with linear algebra)

Consider the problem of minimizing $f(x)$ over the ellipsoid $x^\top Qx \leq 1$. Here Q is a positive definite matrix, that is, Q is symmetric ($Q^\top = Q$), and all eigenvalues of Q are positive. In particular, this implies Q^{-1} exists and its eigenvalues are precisely the reciprocals of Q.

Consider the mirror map $\Phi(x) := \frac{1}{2}x^\top Qx$.

a. What is $\nabla\Phi(x)$? Write your answer as succinctly as possible.

b. What is $D_\Phi(y, x)$? Write your answer as succinctly as possible.

c. Write down the four steps of mirror descent for this particular mirror map.

d. Given a point $z \notin S$, that is, $z^\top Qz > 1$, what is $\arg\min_{p \in S} D_\Phi(p, z)$? Prove that your answer is correct.

Problem 16. ☕☕(This was alluded to, very briefly, in class)

Consider minimizing $f(x)$ unconstrained as follows: at each step we sample $i \in \{1, 2, \ldots, n\}$ uniformly at random and only modify the ith coordinate of x_t to get x_{t+1}.

a. What should this modification be such that $\mathbf{Exp}[x_{t+1} - x_t \mid x_t] = -\eta\nabla f(x_t)$?

b. What can you say about the number of steps required to get ϵ-close to the optimun $f(x_*)$? More precisely, in how many steps, would you get a point $\tilde{x}$ s.t. $f(\tilde{x}) \leq f(x_*) + \epsilon$. Assume $||\nabla f(z)||_2 \leq \rho$ for all $z \in \mathbb{R}^n$.

Problem 17. ☕☕

You are given a very large directed graph $G = (V, E)$ (think the Web Graph) and you want to estimate the number of edges in the graph. Assume you know the number of vertices. Furthermore, assume you cansample a node uniformly at random, and also, assume you can query the out-degree of any node. Each of these steps takes one unit of time. Describe a randomized algorithm which returns a $(1 \pm \epsilon)$ *multiplicative* approximation to the number of edges in G. You are allowed to fail with probability δ. What is the running time? Under what conditions of the degree distribution is it better than the naive deterministic algorithm that goes vertex-by-vertex and answers exactly? Under what conditions of the degree distribution is the running time independent of the number of vertices?

Problem 18. ☕☕☕(This problem has 3 coffeecups only because it is long. But it is very instructive!)

In this exercise we investigate *double hashing*, that I alluded to in class, to store a dictionary $D \subseteq U$ of size s. Assume that *any* hash can be evaluated in $O(1)$ time. Consider selecting a random hash function $h : U \to [n]$ from a pairwise independent family H. For any $i \in [n]$, let n_i denote the number of entries of D that map to i.

a. For a fixed h, how much time does it take to evaluate $\sum_{i=1}^{n} n_i^2$?

b. Calculate an upper bound on $\mathbf{Exp}[\sum_{i=1}^{n} n_i^2]$. Recall, the randomness is over the selection of h.

c. If we keep sampling $h \in H$ till we get one such that $\sum_{i=1}^{n} n_i^2 \leq 5s$, in expectation how many times do we need to keep sampling?

d. After you get such an h satisfying part (c)'s condition, consider choosing n different hash functions $h_i : U \to [n_i^2]$ for each $i \in [n]$. (Of course, if $n_i = 0$, you ignore that i). These hash functions need to be *perfect*, that is, for any i, and for any $x, y \in D, x \neq y$, we must have $h_i(x) \neq h_i(y)$. How much time, in terms of s, does this require? (Recall perfect hashing was done in class)

Using all the machinery you have developed above, describe a hashing scheme which given input a dictionary D of size s, takes $O(s)$ processing time in expectation, but after that can answer *any query* of the form "Is $q \in S$?" for a $q \in U$ in *deterministic* $O(1)$ time.

Problem 19. ☕☕☕[Bloom Filters]

In this exercise, we investigate Bloom Filters. These are extremely space efficient objects which return only *approximate* answers to membership queries. More precisely, there can be false positives; the filter (algorithm) can say a certain $q \in D$ even though it is not. But there would be no false negatives, that is, if $x \in D$, then the filter would definitely say $x \in D$. Bloom filters are used *everywhere*, so please do this exercise!

The storage is a *bit-array* $A[1 : n]$ where the size $n = cs$ for some constant c we will fix later. The pre-processing step involves *independently* sampling k different hash functions $h_1, \ldots, h_k : U \to [n]$ from a pairwise independent Hash family H. The parameter k will be fixed later too. Subsequently, we do the following

For each element $x \in D$ and for each $i \in \{1, \ldots, k\}$, we set $A[h_i(x)] = 1$.

At time of a query $q \in U$, we check if **all** of the bits $A[h_i(q)]$, $i \in [k]$ are set to 1 or not. If so, we say, Yes, $q \in D$, otherwise we say, No, $q \notin D$.

a. What is the time (in terms of k) taken for each query?

b. Why are there no false negatives? That is, if $x \in D$, why would the query return Yes?

c. This is the heart of the problem: what is the chance of a false positive? That is, for any $q \notin D$, what is the probability (over the random choices of the k hash functions) that our algorithm returns Yes? Your answer should be upper-bounded by a function of k and c alone.

   Hint: Note that the algorithm will say $q \in D$ if and only if *for all* $i \in [k]$, we have $A[h_i(q)] = 1$. Fix such an i and let $p_i = h_i(q)$. Now, $A[h_i(q)] = 1$ occurs if and only if there is some $j \in [k]$ (may be i, mayn't be i) and some $x \in D$ such that $h_j(x)$ also evaluates to p_i. What are the chances of that? Take a deep breath and apply the union bound correctly.

d. For what choices of k and c can you make this error go below 1%

Problem 20. [Sim-Hash]

Suppose our universe U is the collection of unit vectors, that is, $U := \{v : ||v||_2 = 1\}$. Consider the *angular distance* function defined over this universe:

$$d(u,v) = \frac{1 - \langle u, v\rangle}{2}$$

In this exercise, you will prove a certain family of Hash functions is LSH for this angular distance metric.

The family is $H := \{h_r : ||r||_2 = 1\}$ is an infinite family of functions where

$$h_r(u) := \mathbf{sgn}(\langle r, u\rangle)$$

where $\mathbf{sgn}x = -1$ if $x < 0$ and $+1$ if $x \geq 0$. For what parameters P_1, P_2 is this family (R, cR, P_1, P_2)-LSH?

Problem 21. [Count-Min Sketch]

In class we looked at the following randomized estimator but then moved on to the Count-Sketch which had better error properties. In this exercise, we formalize the easier algorithm.

Recall, the algorithm picks a hash function $h : [m] \to [k]$ uniformly at random from a universal hash family. It maintains k counters. When a arrives in the stream, the algorithm increments $C[h(a)]$. At the end of the stream, it estimates the frequency of any element a as $\hat{f}_a := C[h(a)]$. Let's introduce another notation : $||\vec{f}_{-}a||_1 := \sum_{b \neq a} f_b$.

a. What is $\mathbf{Exp}[\hat{f}_a]$? Write your answer in terms of f_a, $||\vec{f}_{-}a||_1$, and k.

b. What k would you set, so that $\Pr[|\hat{f}_a - f_a| > \varepsilon ||\vec{f}_{-}a||_1] \leq 1/2$?

c. Imagine running t copies of the above experiment in parallel. That is, sample independently t hash functions $h_1, \ldots, h_t$ uniformly at random from the UHF. Maintain an $t \times k$ matrix of counters, and when you encounter a, increment the counters $C[i][h_i(a)]$ for all $1 \leq i \leq t$. Finally, return $\hat{f}_a := \min_{1 \leq i \leq t} C[h_t(a)]$.

   For what value of t, do you have $\Pr[|\hat{f}_a - f_a| > \varepsilon ||\vec{f}_{-}a||_1] \leq \delta$?

Finally, why would you run the above instead of the deterministic Misra-Gries? Consider a stream where each entry of the stream is of the form $(a, +)$ or $(a, -)$ where the former adds a while the latter wipes out a. Argue that the above algorithm works even for this streaming model with deletions. Misra-Gries has no such analog...

Problem 22. ☕☕

Prove the inequality we didn't prove in class. Namely, for any m non-negative numbers $f_1, \ldots, f_m$, prove that (note for us $\sum_{i=1}^m f_i$ was equal to n)

$$\left(\sum_{i=1}^{m} f_i\right) \cdot \left(\sum_{i=1}^{m} f_i^3\right) \quad \leq \quad \sqrt{m} \cdot \left(\left(\sum_{i=1}^{m} f_i^2\right)\right)^2$$

Hint: There are many ways to do this, and some things might be useful to recall. One, is Cauchy-Schwarz which states $\sum_{i=1}^m (a_i b_i) \leq \sqrt{\sum_{i=1}^m a_i^2}\sqrt{\sum_{i=1}^m b_i^2}$ for any a_i, b_i's. Two, since the function $h(t) = t^k$ is convex for any $k \geq 1$, we get $\left(\frac{1}{m}\sum_{i=1}^m a_i\right)^k \leq \frac{1}{m}\sum_{i=1}^m a_i^k$. Lastly, for any $\theta \geq 1$ and any non-negative z_i's we have $\sum_{i=1}^m z_i^\theta \leq \left(\sum_{i=1}^m z_i\right)^\theta$. This follows by noticing that if $M := \max_i z_i$, then $\sum_{i=1}^m z_i^\theta \leq \sum_{i=1}^m M^{\theta-1} z_i \leq M^{\theta-1}\left(\sum_{i=1}^m z_i\right) \leq \left(\sum_{i=1}^m z_i\right)^\theta$, since $M \leq \sum_{i=1}^m z_i$ and since $\theta - 1 \geq 1$.

(Extra Credit): Generalize the above inequality by proving that for any $k \geq 1$, we have

$$\left(\sum_{i=1}^{m} f_i\right) \cdot \left(\sum_{i=1}^{m} f_i^{2k-1}\right) \quad \leq \quad m^{\left(1-\frac{1}{k}\right)} \cdot \left(\left(\sum_{i=1}^{m} f_i^k\right)\right)^2$$

Problem 23. ☕☕☕☕

Imagine edges of a graph G streaming at you, but the same edge can appear many times. Give a small space algorithm (your answer should take space $o(n)$ (something like $\sqrt{n}$ or $\log n$) where n is the number of vertices, to estimate the $\sum_{v \in V} \deg^2(v)$ where $\deg(v)$ is the degree in the graph with multiple edges removed.

Problem 24. ☕☕

This problem is to make sure you have understood things concretely. You are given a stream of n elements. You are also input an integer $k \geq 1$ and $0 \leq \varepsilon < 1/2$ before the stream starts; think of $k = 5$ and $\varepsilon = 0.01$. Your job is to make one pass on the stream and return a set $S \subseteq [m]$ (recall the entries of the stream are from $[m]$) which satisfies the following two properties:

a. Every $j \in [m]$ which appears more than $> n/k$ times must be in S.

b. Every $j \in S$ must appear $\geq n \cdot (\frac{1}{k} - \varepsilon)$ times.

I just want the *pseudocode* for this problem, and no proof is needed (provide one if you feel like it). But you have to clearly state all details (you can't say "Oh, I will just use the LSH used in class"). You have to be clear on your memory requirements (assume you know m and n for now). You are of course allowed to fail with probability 2^{-100}.

Finally, write a paragraph on what you will do if you had no idea what n and m were. How and where would your pseudocode change?

Problem 25. ☕☕[Dimension Reduction]

Suppose you have n vectors $v_1, \ldots, v_n$ and each of them lie in $\mathbb{R}^d$. Suppose A is a $k \times d$ *random* matrix such that each $A[i,j]$ *independently* is $+1$ with probability $1/2$ and -1 with probability $1/2$. Let $w_i = \frac{1}{\sqrt{k}} \cdot A v_i$ be a k-dimensional random vector. We are interested in how the *lengths* of the w_i's correspond to the v_i's.

a. What is $\mathbf{Exp}[||w_i||_2^2]$?

b. For a fixed i, and a given ε, δ, what is the probability $\Pr[\frac{||w_i||_2^2}{||v_i||_2^2} \notin (1 \pm \varepsilon)]$? Your answer should be in terms of k.

CS 49/149: 21st Century Algorithms (Fall 2018): Exercises

Below are some drill exercises which are **not for submission** but are meant for reinforcement of material we did in class.

Exercise 1. Consider the online decision making problem done in class where $g_i(t)$ is the *gain/profit* obtained if we play action i at time t. Design and analyze the MWU algorithm for maximizing total expected gain as compared to the best fixed action in hindsight. This will mimic the analysis done in class; please try doing this without looking at notes, etc.

Exercise 2. Show an example of a setting where the MWU algorithm actually does better (gets less loss or more gain) than the fixed action in hindsight.

Exercise 3. Implement the approximate LP solver in your favorite language. Generate a "random" linear program by taking entries of A, b, c at random. Does your implementation work "fast"? Do you see a dependence on the width?

Exercise 4. Run the approximate LP solver done in class on the vertex cover LP :

$$\min \sum_{v \in V} c(v) x_v : \;\; \forall (u,v) \in E, \; x_u + x_v \geq 1, \quad 0 \leq x_u \leq 1$$

What is the oracle? What is the width? What is the final algorithm actually doing?

Exercise 5. We are given a population of m individuals where each person is either a man or a woman. You want to estimate the *number* of women in the population, upto a *multiplicative* factor. More precisely, suppose w was the number of women. You are given parameters ϵ, δ, and your goal is to return a number W

$$\Pr[W \notin (1 \pm \epsilon) w] \leq \delta$$

That is, with probability $\geq (1 - \delta)$, the fraction W/w is in the range $[1 - \epsilon, 1 + \epsilon]$.

The only access you have is to sample an individual u.a.r from the population and know their gender. How many samples will you need?