

Homework 1

CSE 541: Interactive Learning, Autumn 2026

Instructor: Kevin Jamieson

Due Thursday, October 22, 2026, 11:59 PM Pacific (Gradescope)

Instructions. Submit a single typeset PDF to Gradescope (L^AT_EX strongly recommended; no photos or scans). Put the code for Problems 1 and 3 at the end of the respective problem, after the figures. You may discuss with classmates, but write up your own solutions and code and list your collaborators. Read the course AI policy before you start; in particular, on proof questions use AI only minimally, and whatever you use it for, say so in your submission. Total: 100 points.

Notation. $[d] = \{1, \dots, d\}$, Δ_d is the probability simplex in $\mathbb{R}^d$, $\langle a, b \rangle = a^\top b$, and $\log$ is the natural logarithm. Throughout, “exponential weights” (EW) with learning rate η over d actions means $w_1 \equiv 1$, $p_t(i) = w_t(i) / \sum_j w_t(j)$, $w_{t+1}(i) = w_t(i) e^{-\eta \ell_t(i)}$, as in lecture.

Problem 1: Tracking the best stock: fixed share as a fund of funds (25 points)

Setting. d stocks. On day t stock i has gross return $r_t(i) = \text{price}_{t+1}(i) / \text{price}_t(i) > 0$; the vector $r_t \in \mathbb{R}_{>0}^d$ is revealed only after you have chosen a portfolio $p_t \in \Delta_d$, and your wealth is multiplied by $\langle p_t, r_t \rangle$. As in lecture, use the log loss $\ell_t(i) = -\log r_t(i)$ and learning rate $\eta = 1$, so that $e^{-\eta \ell_t(i)} = r_t(i)$: every exponential-weights expression below is a *wealth*, and nothing needs to be bounded or clipped.

Two algorithms from lecture, restated. Exponential weights with $\eta = 1$ keeps unnormalized weights $w_1 = (1, \dots, 1)$, $w_{t+1}(i) = w_t(i) r_t(i)$, and plays $p_t = w_t / W_t$ with $W_t = \sum_j w_t(j)$. Since $w_t(i) = \prod_{s < t} r_s(i)$ is the value of a dollar put into stock i on day 1, this is **buy and hold**: \$1/ d in each stock, never rebalanced. Lecture showed it loses at most $\log d$ in log-wealth to the best single stock in hindsight.

A moving target. Now compete with a stock that is allowed to *change*: a sequence $u = (u_1, \dots, u_T) \in [d]^T$ with $S(u) = |\{t < T : u_{t+1} \neq u_t\}|$ switches, and the *dynamic regret* in log-wealth

$$\text{D-Reg}_T(u) = \sum_{t=1}^T \log r_t(u_t) - \sum_{t=1}^T \log \langle p_t, r_t \rangle = \log \frac{\text{wealth of the switching strategy } u}{\text{wealth of the algorithm}}.$$

Buy and hold is hopeless here: it moves its wealth into a stock exactly as fast as that stock grows, so after a regime change it takes as long to move out as it took to move in. (Two stocks, the first doubling daily for $T/2$ days and then halving daily while the second does the reverse: buy and hold ends with a dollar, the one-switch comparator with 2^T .) Something has to *forget*.

Fixed share (Herbster and Warmuth, *Tracking the best expert*, Machine Learning 1998) forgets by moving a small fraction of the wealth back to uniform every day:

Fixed share(α) with log loss and $\eta = 1$. Initialize unnormalized weights $w_1 = (1, \dots, 1)$. For $t = 1, 2, \dots$: play $p_t = w_t/W_t$, $W_t = \sum_j w_t(j)$; observe r_t ; set

$$\tilde{w}_{t+1}(i) = w_t(i) r_t(i), \quad \widetilde{W}_{t+1} = \sum_{j=1}^d \tilde{w}_{t+1}(j), \quad w_{t+1}(i) = (1 - \alpha) \tilde{w}_{t+1}(i) + \frac{\alpha}{d} \widetilde{W}_{t+1}.$$

In words: let every dollar ride for one day (the buy-and-hold update), then take an α fraction of *all* the wealth and spread it uniformly over the d stocks. The played portfolio obeys $p_{t+1} = (1 - \alpha)\tilde{p}_t + \alpha(1/d, \dots, 1/d)$ with $\tilde{p}_t(i) = p_t(i)r_t(i)/\langle p_t, r_t \rangle$: the buy-and-hold step followed by the α mixing step. Write

$$Z_t := \langle p_t, r_t \rangle = \widetilde{W}_{t+1}/W_t \quad \text{for the factor by which fixed share's wealth grows on day } t.$$

You may use the following without proof. (It is lecture's best-single-stock argument, which was exact for the log loss, plus one bookkeeping inequality per round: $\tilde{v}_t(u_t) \leq w$ -mass of u_{t+1} on day $t + 1$ divided by $1 - \alpha$ if $u_{t+1} = u_t$, and by α/d if not.)

Theorem 1 (Herbster and Warmuth, log-loss version). *Let $u \in [d]^T$ have $S = S(u)$ switches. Fixed share with log loss and $\eta = 1$ satisfies:*

- (i) *with a constant α , $\text{D-Reg}_T(u) \leq (S + 1) \log d + S \log \frac{1}{\alpha} + (T - 1 - S) \log \frac{1}{1 - \alpha} \stackrel{\alpha=S/(T-1)}{\leq} (S + 1) \log d + S \log \frac{e(T - 1)}{S}$;*
- (ii) *with the time-varying $\alpha_t = 1/t$ (use α_{t+1} when forming w_{t+1}), $\text{D-Reg}_T(u) \leq (S + 1) \log d + (S + 1) \log T$, with no knowledge of S .*

No $\sqrt{T}$ and no $\eta T/8$: for the log loss the per-round step is an identity, and what remains is the price of S switches, $O(S \log(dT))$; compare the $\sqrt{TS}$ one gets for bounded losses. Part (ii) comes from the same proof: a switch at round t costs $\log(d/\alpha_{t+1}) \leq \log(dT)$ and a non-switch costs $\log \frac{1}{1 - \alpha_{t+1}} = \log \frac{t+1}{t}$, which telescopes to $\log T$ over the whole horizon. Not having to know S costs only $\log T$ instead of $\log(T/S)$ per switch.

1.1 (12 points) Fixed share is a fund of funds. The algorithm above has a second, equivalent description that makes it transparent. For each day $s \in [T]$ let $\mathcal{B}_s$ be the *buy-and-hold fund opened on day s* : it starts with equal amounts in every stock and never rebalances, so on day $t \geq s$ its portfolio and its daily growth factor are

$$p_t^{(s)}(i) = \frac{\prod_{\tau=s}^{t-1} r_\tau(i)}{\sum_j \prod_{\tau=s}^{t-1} r_\tau(j)} \quad \text{and} \quad Z_t^{(s)} := \langle p_t^{(s)}, r_t \rangle$$

(so $p_s^{(s)}$ is uniform). Let $\nu_t(s)$ denote the *fraction of your total wealth sitting in fund $\mathcal{B}_s$ at the start of day t* , defined by $\nu_1(1) = 1$ and, for $t \geq 1$,

$$\nu_{t+1}(s) = (1 - \alpha) \nu_t(s) \frac{Z_t^{(s)}}{Z_t} \quad (s \leq t), \quad \nu_{t+1}(t + 1) = \alpha.$$

That is: during day t fund s grows by $Z_t^{(s)}$ while the whole pot grows by Z_t , so fund s 's share becomes $\nu_t(s)Z_t^{(s)}/Z_t$; then you skim α of everything to open a new fund $\mathcal{B}_{t+1}$. Every fund is a plain buy-and-hold portfolio; all the “learning” is in how the shares ν_t move. (If α varies with time, read α_{t+1} for α in the recursion.)

- (a) Prove by induction on t that the portfolio this fund of funds holds, $\sum_{s=1}^t \nu_t(s) p_t^{(s)}$, is exactly fixed share's p_t , and along the way that the pot's growth factor is $Z_t = \sum_{s \leq t} \nu_t(s) Z_t^{(s)}$ (so ν_{t+1} sums to one).
- (b) Unroll the recursion to show

$$\nu_t(s) \propto \pi_t(s) \prod_{\tau < s} Z_\tau \prod_{\tau=s}^{t-1} Z_\tau^{(s)}, \quad \pi_t(1) = (1 - \alpha)^{t-1}, \quad \pi_t(s) = \alpha(1 - \alpha)^{t-s} \quad (s \geq 2).$$

Read this as exponential weights with $\eta = 1$ over the *opening days* s , with prior π and with log loss: fund s is charged $-\log Z_\tau^{(s)}$ on each day $\tau \geq s$ it is open, and the pot's own $-\log Z_\tau$ on the days $\tau < s$ before it opened (an unopened fund is credited with whatever the pot did). Explain in a few sentences why that makes fixed share “exponential weights on top of exponential weights”, and point to the term in the theorem that is the price of the prior π .

1.2 (13 points) Implementation. Starting from `hw1_starter.py`, implement (i) buy and hold, (ii) fixed share with log loss and $\eta = 1$, and (iii) the fund of funds of 1.1, i.e. $\sum_s \nu_t(s) p_t^{(s)}$ computed explicitly from the recursion for ν_t (keep every fund; $O(dt)$ per day is fine). Run them on

- the synthetic regime-switching market from the starter ($d = 8$, $T = 5000$, $S \in \{1, 3, 10\}$ regime changes; the daily drifts are exaggerated so that regimes are visible), against the comparator that holds the best stock within each regime; and
- the real data in `stock_closes.csv` ($d = 8$ stocks, 250 trading days of returns), against the comparator that holds the best stock within each block of 50 days ($S = 4$).

Use the parameter-free choice $\alpha_t = 1/t$; there is nothing to tune. For each setting plot the cumulative log-wealth of the three algorithms and of the comparator on one set of axes, with the log-wealth of each individual stock drawn faintly in the background (so one can see what the algorithm is doing relative to the raw assets), report $D\text{-Reg}_T(u)$ in a table together with the theorem's bound (ii), and confirm numerically that (ii) and (iii) hold the same portfolio every day up to floating point. Then, in at most half a page: what do you see, and how much does knowing S buy? Rerun fixed share with the tuned constant $\alpha = S/(T - 1)$ and with that value off by a factor of 10 in each direction, and compare.

Context. Fixed share goes back to Herbster and Warmuth (1998), who posed the “tracking the best expert” problem and introduced the mixing update; the theorem above is their bound specialized to the log loss. The same update is alive today: Algorithm 1 of Zhang, Wang, Zhao and Jamieson, *Universal Dynamic Portfolios* (2026, <https://arxiv.org/abs/2609.34643>) is *literally* the fixed-share recursion you just implemented, with the same parameter-free $\alpha_t = 1/t$ (which is why it needs no knowledge of how often the comparator moves) and with the d single stocks replaced by all constantly rebalanced portfolios $u \in \Delta_d$: the weights w_t become a density on the simplex, initialized to the uniform Dirichlet, updated by $w_{t+1}(u) \propto (1 - \alpha_t) w_t(u) \langle u, r_t \rangle + \alpha_t \text{Dir}(\mathbf{1})(u)$, and the algorithm plays the mean of w_t (Cover's universal portfolio with forgetting). What changes is the comparator: competing with arbitrary sequences in the simplex rather than with single stocks is what produces their path-length rates and the lower bound in that paper.

Problem 2: Exponential weights and gradient descent are the same algorithm (25 points)

Setting. Online linear optimization over a convex set $\mathcal{K} \subseteq \mathbb{R}^d$: on round t the learner plays $x_t \in \mathcal{K}$, then sees $g_t \in \mathbb{R}^d$ and pays $\langle g_t, x_t \rangle$. (Convex losses reduce to this by $\ell_t(x_t) - \ell_t(u) \leq \langle \nabla \ell_t(x_t), x_t - u \rangle$.)

Follow the regularized leader (FTRL) with regularizer $R : \mathcal{K} \rightarrow \mathbb{R}$ and learning rate $\eta > 0$ plays

$$x_t = \arg \min_{x \in \mathcal{K}} \Phi_t(x), \quad \Phi_t(x) = \sum_{s=1}^{t-1} \langle g_s, x \rangle + \frac{1}{\eta} R(x),$$

so $x_1 = \arg \min_{\mathcal{K}} R$. Write $\text{Regret}_T(u) = \sum_{t=1}^T \langle g_t, x_t - u \rangle$.

2.1 (8 points) Be the leader. Think of R/η as a fictitious loss suffered at round 0, so that FTRL is plain “follow the leader” on the sequence $R/\eta, \langle g_1, \cdot \rangle, \langle g_2, \cdot \rangle, \dots$, and $x_1 = \arg \min_{\mathcal{K}} R$ is the leader after round 0. Prove by induction on T that for every $u \in \mathcal{K}$,

$$\sum_{t=1}^T \langle g_t, x_{t+1} \rangle + \frac{1}{\eta} R(x_1) \leq \sum_{t=1}^T \langle g_t, u \rangle + \frac{1}{\eta} R(u),$$

and deduce the master bound

$$\text{Regret}_T(u) \leq \frac{R(u) - R(x_1)}{\eta} + \sum_{t=1}^T \langle g_t, x_t - x_{t+1} \rangle.$$

(The first term is the range of the regularizer, $R(u) - \min_{\mathcal{K}} R \geq 0$; the second is the *stability* of the iterates. This is the “be-the-leader” lemma of Kalai and Vempala (2005); the hypothetical player who uses tomorrow’s iterate today beats every fixed u .)

2.2 (8 points) Negative entropy gives exponential weights. Take $\mathcal{K} = \Delta_d$ and $R(p) = \sum_i p_i \log p_i$.

- Solve the FTRL step (Lagrange multiplier for the constraint $\sum_i p_i = 1$) and show $x_t(i) \propto \exp(-\eta \sum_{s < t} g_s(i))$, i.e. FTRL *is* exponential weights.
- Assume $g_t \in [0, 1]^d$. Show $\langle g_t, x_t - x_{t+1} \rangle \leq \eta \sum_i x_t(i) g_t(i)^2 \leq \eta$. *Hint:* $x_{t+1}(i) = x_t(i) e^{-\eta g_t(i)} / Z$ with $Z = \sum_j x_t(j) e^{-\eta g_t(j)}$; explain why $Z \leq 1$ here, then use $e^{-a} \geq 1 - a$.
- Conclude $\text{Regret}_T(u) \leq \frac{\log d}{\eta} + \eta T$. Lecture proved $\frac{\log d}{\eta} + \frac{\eta T}{8}$ for the same algorithm. In one or two sentences, where did the factor of 8 go?

2.3 (7 points) A quadratic regularizer gives gradient descent. Take $R(x) = \frac{1}{2} \|x\|_2^2$ and assume $0 \in \mathcal{K}$, so $x_1 = 0$. Let $\Pi_{\mathcal{K}}$ denote Euclidean projection onto $\mathcal{K}$.

- Show $x_{t+1} = \Pi_{\mathcal{K}}(-\eta \sum_{s \leq t} g_s)$. For $\mathcal{K} = \mathbb{R}^d$ conclude $x_{t+1} = x_t - \eta g_t$, exactly online gradient descent. (For general $\mathcal{K}$ this is “lazy” projection: lecture’s OGD projects the running iterate rather than the running gradient sum; both have the same guarantee.)
- Show $\|x_t - x_{t+1}\|_2 \leq \eta \|g_t\|_2$ and hence $\langle g_t, x_t - x_{t+1} \rangle \leq \eta \|g_t\|_2^2$.
- Conclude $\text{Regret}_T(u) \leq \frac{\|u\|_2^2}{2\eta} + \eta \sum_t \|g_t\|_2^2$, and with $\|u\|_2 \leq D$, $\|g_t\|_2 \leq G$ and a good η , $\text{Regret}_T \leq \sqrt{2} DG \sqrt{T}$.

2.4 (2 points) The pattern. In both cases the regret came out as

$$\text{Regret}_T(u) \leq \frac{R(u) - R(x_1)}{\eta} + \eta \sum_{t=1}^T \|g_t\|_2^2,$$

with $\|\cdot\| = \|\cdot\|_\infty$ for negative entropy on the simplex and $\|\cdot\| = \|\cdot\|_2$ for the quadratic. This is a general fact (which you may take as given): whenever the regularizer R is *strongly convex* in a norm, FTRL's iterates move by at most η times the gradient measured in the corresponding dual norm, and the stability term is at most $\eta \|g_t\|^2$ in that dual norm. Negative entropy is strongly convex in $\|\cdot\|_1$ (whose dual is $\|\cdot\|_\infty$) and $\frac{1}{2} \|\cdot\|_2^2$ is strongly convex in $\|\cdot\|_2$ (self-dual); everything else about exponential weights and gradient descent is the same algorithm. For 2 points: using this fact and 2.1, write down in one line the regret bound you get from a regularizer whose range on $\mathcal{K}$ is at most D_R when all gradients satisfy $\|g_t\| \leq G$ in the dual norm, and the η that optimizes it.

Problem 3: Does online learning actually do anything? ERM versus online gradient descent (25 points)

Lectures 3–4 proved that on IID data, ERM gets the $1/n$ rate for squared loss, and that online gradient descent, after online-to-batch conversion, gets the same rate up to a log (Problem 4). So what, if anything, does the online method buy us? This problem is empirical; the second half of `hw1_starter.py` contains the data generators and plotting scaffolding.

Setting (streaming linear regression). $x_t \in \mathbb{R}^d$ is drawn uniformly from the unit sphere (so $\Sigma = \mathbb{E}[xx^\top] = I/d$), and $y_t = \langle w_t^*, x_t \rangle + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, $\|w_t^*\|_2 = 1$. The round- t loss is $\ell_t(w) = \frac{1}{2}(\langle w, x_t \rangle - y_t)^2$. For a fixed target w^* the population excess risk has the closed form $L(w) - L(w^*) = \frac{1}{2d} \|w - w^*\|_2^2$, and L is λ -strongly convex with $\lambda = 1/d$, so Problem 4's step size is $\eta_t = d/t$. The starter provides three streams: **stationary** ($w_t^* \equiv w^*$), **drift** (w_t^* rotates slowly in a fixed 2-plane, completing a few revolutions over the horizon), and **switch** ($w_t^* = w^*$ for $t \leq T/2$ and $-w^*$ afterwards). Use $d = 10$, $T = 5000$, $\sigma = 0.1$, and average every curve over 10 seeds.

Algorithms to implement.

- (i) **ERM / follow the leader:** $w_t = \arg \min_w \sum_{s \leq t} \ell_s(w) + \frac{\rho}{2} \|w\|^2$ with a tiny $\rho = 10^{-3}$ for invertibility, re-solved each round (direct least squares; $O(d^3)$ per round is fine here, or use recursive least squares).
- (ii) **OGD, decaying step:** $w_{t+1} = \Pi_{\|w\| \leq 2}(w_t - \eta_t g_t)$ with $g_t = (\langle w_t, x_t \rangle - y_t)x_t$ and $\eta_t = d/t$; also keep the running average $\bar{w}_t = \frac{1}{t} \sum_{s \leq t} w_s$.
- (iii) **OGD, constant step:** same with $\eta_t \equiv \eta \in \{0.05, 0.2, 0.5\}$.

3.1 (8 points) Stationary data: everyone gets $1/n$. On the stationary stream, plot the excess risk $\frac{1}{2d} \|w - w^*\|^2$ of (i) w_t , (ii) w_t and $\bar{w}_t$, and (iii) with $\eta = 0.2$, against t on log-log axes. Fit a slope to each curve over $t \in [500, 5000]$ and report it. Which methods achieve the $1/n$ rate of Problem 4.3 and Lecture 3? Where does constant-step OGD plateau, and how does the plateau depend on η (try the three values)? Give a one-line heuristic explanation.

3.2 (10 points) Moving targets: now the online method does something. On the drift and switch streams, plot the cumulative dynamic regret $\sum_{s \leq t} (\ell_s(w_s) - \ell_s(w_s^*))$ against t for (i), (ii) (last iterate) and (iii) with all three η 's, and separately plot the tracking error $\|w_t - w_t^*\|$ for ERM and for the best constant step. Which methods have *linear* dynamic regret, and why? Explain in terms of how each method weights the past: show that ERM and $\eta_t \propto 1/t$ both estimate (roughly) the time-average of $w_1^*, \dots, w_{t-1}^*$, whereas constant-step OGD is an exponentially weighted average with memory of about d/η rounds. For the rotating target, what does ERM's estimate converge to, and what per-round excess risk does that imply? For the switch, how long does ERM take to recover, and how long does constant-step OGD take?

3.3 (5 points) Fixing ERM, and what it costs. Implement sliding-window ERM (least squares on the most recent W rounds) for $W \in \{50, 200, 1000\}$ and compare it with constant-step OGD on the drift stream. Which W corresponds to which η , roughly? Then compare per-round cost: count the floating-point operations of (a) re-solving least squares on a window, (b) recursive least squares with forgetting, and (c) OGD, as functions of d and W . For $d = 10^4$, which of these are feasible at 10^4 rounds per second?

3.4 (2 points) In at most a paragraph: based on 3.1–3.3 and the lectures, when does online learning actually do something that ERM does not, and when is ERM simply better?

Problem 4: Curvature, $\log T$ regret, and the $1/n$ rate (25 points)

Setting. Online convex optimization over a closed convex $\mathcal{K} \subseteq \mathbb{R}^d$ with online gradient descent, $x_{t+1} = \Pi_{\mathcal{K}}(x_t - \eta_t g_t)$, $g_t = \nabla \ell_t(x_t)$, $\|g_t\|_2 \leq G$. A function f is λ -strongly convex on $\mathcal{K}$ if $f(u) \geq f(x) + \langle \nabla f(x), u - x \rangle + \frac{\lambda}{2} \|u - x\|_2^2$ for all $x, u \in \mathcal{K}$.

4.1 (9 points) OGD with strongly convex losses. Suppose every ℓ_t is λ -strongly convex and use the *decreasing* step sizes $\eta_t = \frac{1}{\lambda t}$. Show that for every $u \in \mathcal{K}$,

$$\sum_{t=1}^T (\ell_t(x_t) - \ell_t(u)) \leq \frac{G^2}{2\lambda} (1 + \log T).$$

Hint: start from $\ell_t(x_t) - \ell_t(u) \leq \langle g_t, x_t - u \rangle - \frac{\lambda}{2} \|x_t - u\|^2$ and the OGD inequality $\langle g_t, x_t - u \rangle \leq \frac{\|x_t - u\|^2 - \|x_{t+1} - u\|^2}{2\eta_t} + \frac{\eta_t}{2} \|g_t\|^2$ from lecture; with $1/\eta_0 := 0$, look at the coefficient of $\|x_t - u\|^2$ after summing. Then say in one sentence what happened to lecture's trade-off between $D^2/(2\eta)$ and $\eta G^2 T/2$.

4.2 (5 points) Online to batch. Now let (x_t, y_t) be i.i.d., $\ell_t(w) = \ell(w; x_t, y_t)$, and $L(w) = \mathbb{E}[\ell(w; X, Y)]$, convex in w . Let $w_1, \dots, w_T$ be the iterates of any online algorithm for which w_t depends only on $(x_s, y_s)_{s < t}$, and let $\bar{w}_T = \frac{1}{T} \sum_t w_t$. Show that for every fixed u ,

$$\mathbb{E}[L(\bar{w}_T)] - L(u) \leq \frac{\mathbb{E}[\text{Regret}_T(u)]}{T}, \quad \text{Regret}_T(u) = \sum_{t=1}^T (\ell_t(w_t) - \ell_t(u)).$$

Combined with 4.1, an i.i.d. problem with λ -strongly convex losses has excess risk $O\left(\frac{G^2 \log n}{\lambda n}\right)$ after n samples: a $1/n$ rate.

4.3 (9 points) Squared loss. Let $\ell(w; x, y) = \frac{1}{2}(\langle w, x \rangle - y)^2$ with $\|x\|_2 \leq 1$, $y \in [0, 1]$, and $\mathcal{K} = \{w : \|w\|_2 \leq B\}$. Write $\Sigma = \mathbb{E}[XX^\top]$.

- Show that for a single sample $\ell(\cdot; x, y)$ is *not* λ -strongly convex for any $\lambda > 0$ (when $d \geq 2$), but that the population risk L is λ -strongly convex on $\mathbb{R}^d$ if and only if $\Sigma \succeq \lambda I$.
- Show that the proof of 4.1 goes through *in expectation* using only strong convexity of L : with stochastic gradients $g_t = \nabla \ell_t(w_t) = (\langle w_t, x_t \rangle - y_t)x_t$ and $\eta_t = 1/(\lambda t)$,

$$\mathbb{E}[L(\bar{w}_T)] - \min_{u \in \mathcal{K}} L(u) \leq \frac{(B+1)^2 (1 + \log T)}{2\lambda T}.$$

Hint: $\mathbb{E}[\langle g_t, w_t - u \rangle \mid w_t] = \langle \nabla L(w_t), w_t - u \rangle$, and the OGD inequality holds pathwise.

- (c) Lecture 3 proved a $\log / n$ excess-risk rate for ERM over any *convex* class $\mathcal{F}$ with no condition like $\Sigma \succeq \lambda I$; the class $\mathcal{F} = \{x \mapsto \langle w, x \rangle : w \in \mathcal{K}\}$ here is convex. In a short paragraph: what does each argument assume, what does each deliver (algorithm? constants? dimension dependence?), and why is realizability ($f^* \in \mathcal{F}$) irrelevant to both?

4.4 (2 points) The squared loss on a bounded domain is *exp-concave* even when Σ is degenerate, and the Online Newton Step algorithm achieves $O(d \log T)$ regret without any λ . In two sentences, connect this to (i) the “no slack” phenomenon on the mixability slide and (ii) the gap between OGD and Cover’s algorithm on the portfolio slide. (No proof.)