A. Experiment details

In this section we provide the details of the experiments corresponding to the results presented in Section 5. In each case we describe the underlying problem, the initial optimization problem, and how this translates into proximal form for the Epsilon solver (that is, a sum of objective terms where each term has an efficient proximal operator or epigraph projection, plus a set of linear equality constraints), using the transformations from Section 3. All these examples are included in the Epsilon distribution, available at `http://epopt.io`.

A.1. Lasso

The Lasso problem is

$$\underset{\theta}{\text{minimize}} \;\; (1/2)\|X\theta - y\|_2^2 + \lambda\|\theta\|_1, \tag{40}$$

with input features $X \in \mathbb{R}^{m\times n}$, response variables $y \in \mathbb{R}^m$, and model parameters $\theta \in \mathbb{R}^n$. The regularization parameter $\lambda \geq 0$ controls the tradeoff between data fit and the ℓ_1 regularization term which encourages sparsity in the model parameters. The Lasso is especially useful in the high-dimensional case where $m < n$ as the sparsity induced by ℓ_1 regularization effectively controls the number of free parameters in the model, see Tibshirani (1996) for details. The Lasso problem as written is already in proximal form with

$$\begin{aligned} f_1(\theta) &= (1/2)\|X\theta - y\|_2^2, \\ f_2(\theta) &= \lambda\|\theta\|_1. \end{aligned} \tag{41}$$

In our experiments, we generate $X \in \mathbb{R}^{1500\times 5000}$ from a standard Normal distribution and set $y = X\theta_0 + \epsilon$ with θ_0 having 1% nonzero standard Normal entries and $\epsilon \sim \mathcal{N}(0, 0.05^2)$. We set the regularization parameter $\lambda = 0.5\|X^T y\|_\infty$.

A.2. Sparse inverse covariance

Sparse inverse covariance estimation models a multivariate Gaussian distribution over n variables by solving the optimization problem

$$\underset{\Theta}{\text{minimize}} \;\; -\log|\Theta| + \operatorname{tr} S\Theta + \lambda\|\Theta\|_1 \tag{42}$$

where $S \in \mathbb{S}^n$ is the sample covariance, $\Theta \in \mathbb{S}^n$ is the estimated inverse covariance and the ℓ_1 norm $\|\cdot\|_1$ is applied elementwise. As with the Lasso, the ℓ_1 penalty promotes sparse structure and is especially useful in the high-dimensional case with more variables than samples ($m < n$), see Friedman et al. (2008) for details. With respect to our framework, given a proximal operator for the $-\log|\cdot|$ term, sparse inverse covariance estimation can trivially be put in proximal form with

$$\begin{aligned} f_1(\Theta) &= -\log|\Theta| + \operatorname{tr} S\Theta, \\ f_2(\Theta) &= \lambda\|\Theta\|_1. \end{aligned} \tag{43}$$

where in the construction of f_1, we have exploited the fact that any proximal operator can easily be combined with a linear function (see e.g. Parikh & Boyd (2013) for details). In our experiments, we construct the sample covariance from 100 samples drawn from $\mathcal{N}(0, \Theta_0^{-1})$ where $\Theta_0 \in \mathbb{S}^{200}$ has uniform random entries and is 1% nonzero.

A.3. MNIST (2000 images)

The MNIST dataset (LeCun et al., 1998) consists of handwritten digits, constructed with the goal of building a classifier for automatically recognizing each digit ($\{0,\ldots,9\}$) in each image. As a linear classifier applied directly to the raw pixels performs poorly, we generate random Fourier features (Rahimi & Recht, 2007) and train a classifier using sparse softmax regression

$$\underset{\Theta}{\text{minimize}} \;\; \ell(X, y; \Theta) + \lambda\|\Theta\|_1 \tag{44}$$

where $X \in \mathbb{R}^{m\times n}$ are the image features, $y \in \{0,\ldots,9\}^m$ are the image labels, $\lambda \geq 0$ is the regularization parameter and the softmax loss, parameterized by $\Theta \in \mathbb{R}^{n\times 10}$, is given by

$$\ell(X, y; \Theta) = \sum_{i=1}^{m}\left(\log\sum_{k=1}^{10}\exp(x_i^T\theta_k) - x_i^T\theta_{y_i}\right). \tag{45}$$

Unlike the least squares loss employed in the Lasso, the softmax loss cannot easily be composed with an arbitrary linear function and this requires the introduction of an additional auxiliary variable, $Z \in \mathbb{R}^{m\times 10}$. With this additional variable, the proximal form for this problem is given by

$$\begin{aligned} f_1(Z) &= \sum_{i=1}^{m}\left(\log\sum_{k=1}^{10}\exp(Z_{ik})\right) \\ f_2(\Theta) &= \lambda\|\Theta\|_1 - \sum_{i=1}^{m} x_i^T\theta_{y_i} \end{aligned} \tag{46}$$

with equality constraints

$$Z = X\Theta \tag{47}$$

In our experiments, we train the classifier on 2000 images using 1000 random Fourier features and $\lambda = 0.1$.

A.4. Robust SVM

In our experiments we use an ℓ_∞ variant of the support vector machine, similar to the formulations in (Lanckriet et al., 2003; Shivaswamy et al., 2006), but with an ℓ_∞ uncertainty

Function			Proximal / epigraph operator	
Type	**Atom**	**Definition**	**Method**	**Complexity**
Elementwise $x, y \in \mathbb{R}$	Absolute	$f(x) = \lvert x\rvert \ (\equiv \lVert x\rVert_1)$	soft thresholding sum-of-max	$O(n)$ $O(n)$
	Square	$f(x) = x^2 \ (\equiv \lVert x\rVert_2^2)$	linear equation cubic equation	$O(n)$ $O(n)$
	Hinge	$f(x) = \max\{x, 0\}$	soft thresholding sum-of-max	$O(n)$ $O(n)$
	Deadzone	$f(x) = \max\{\lvert x\rvert - \epsilon, 0\},\ \epsilon \geq 0$	soft thresholding sum-of-max	$O(n)$ $O(n)$
	Quantile	$f(x) = \max\{\alpha x, (\alpha - 1)x\},\ 0 \leq \alpha \leq 1$	soft thresholding sum-of-max	$O(n)$ $O(n)$
	Logistic	$f(x) = \log(1 + \exp(x))$	Newton Primal-dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)
	Inverse positive	$f(x) = 1/x,\ x \geq 0$	cubic equation Implicit dual Newton	$O(n)$ $O(n) \cdot$ (# Newton)
	Negative log	$f(x) = -\log(x),\ x \geq 0$	quadratic equation Implicit dual Newton	$O(n)$ $O(n) \cdot$ (# Newton)
	Exponential	$f(x) = \exp(x)$	Newton Primal-dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)
	Negative entropy	$f(x) = x \cdot \log(x),\ x \geq 0$	Projected Newton Implicit dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)^2
	KL Divergence	$f(x, y) = x \cdot \log(x/y),\ x, y \geq 0$	Projected Newton Implicit dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)^2
	Quadratic over linear	$f(x, y) = x^2/y,\ y \geq 0$	Projected Newton Implicit dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)^2
Vector $x \in \mathbb{R}^n$	ℓ_2-norm	$f(x) = \lVert x\rVert_2$	group soft thresholding analytic projection	$O(n)$ $O(n)$
	Maximum	$f(x) = \max_i x_i \ \ (\lVert x\rVert_\infty \equiv \max_i \lvert x_i\rvert)$	sum-of-max sum-of-max	$O(n)$ $O(n)$
	Sum-k-largest	$f(x) = \sum_{i=1}^k x_{[i]} \ \ (x_{[i]} \geq x_{[i+1]})$	sum-of-clip bisection	$O(n)$ $O(n) \cdot$ (# Bisection)
	Log-sum-exp	$f(x) = \log\left(\sum_{i=1}^n \exp(x_i)\right)$	Newton Primal-dual Newton	$O(n) \cdot$ (# Newton) $O(n) \cdot$ (# Newton)
Matrix $X \in \mathbb{R}^{n \times n}$	Negative log det Nuclear norm Spectral norm	$f(X) = -\log\det(X), X \in \mathbb{S}^n$ $f(X) = \lVert\sigma(X)\rVert_1, X \in \mathbb{R}^{m \times n}$ $f(X) = \lVert\sigma(X)\rVert_\infty, X \in \mathbb{R}^{m \times n}$	$-\log$ on $\lambda(X)$ $\lVert\cdot\rVert_1$ on $\sigma(X)$ $\lVert\cdot\rVert_\infty$ on $\sigma(X)$	$O(n^3)$ $O(n^3)$ $O(n^3)$

Table 2. Complete list of proximal and epigraph projection operators implemented for this work. Most proximal operators (except sum-k-largest) have appeared in some form in previous literature, but epigraph projections are typically novel to this work.

ball instead of an ℓ_2 uncertainty ball. In this setting, given input data (x_i, y_i) we wish to train an SVM by minimizing the standard regularized hinge loss,

$$\underset{\theta}{\text{minimize}} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{i=1}^m \max\{0, 1 - y_i \cdot \bar{x}_i^T\theta\} \tag{48}$$

but where $\bar{x}_i$ lies in some uncertainty set centered at x_i, $\bar{x}_i = x_i + Pu$ where $\|u\|_\infty \leq 1$. This can be expressed as the optimization problem

$$\underset{\theta}{\text{minimize}} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{i=1}^m \sup_{\|u_i\|_\infty \leq 1} \max\{0, 1 - y_i \cdot \theta^T(\bar{x}_i + Pu_i)\} \tag{49}$$

which, using the relation that $\sup_{\|u\|_\infty \leq 1} c^T u = \|c\|_1$ where, is equivalent to the optimization problem

$$\underset{\theta}{\text{minimize}} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{i=1}^m \max\{0, 1 - y_i \cdot \theta^T x_i + \|P^T\theta\|_1\}. \tag{50}$$

As discussed in Section 3, this is transformed to proximal form

$$\begin{aligned} f_1(\theta) &= \frac{\lambda}{2}\|\theta\|_2^2 \\ f_2(z_3) &= \sum \max\{z_3, 0\} \\ f_3(z_1, z_2) &= I\{\|z_1\|_1 \leq z_2\} \end{aligned} \tag{51}$$

and equality constraints

$$\begin{aligned} z_1 &= P^T x \\ z_3 &= 1 - \operatorname{diag}(y) X\theta + 1 z_2. \end{aligned} \tag{52}$$

In our experiments, we generated $X \in \mathbb{R}^{2500\times 750}$ random uniform [0,1], $\theta \in \mathbb{R}^{750}$ also random uniform, and set $y = \operatorname{sign}(x_i^T\theta + \mathcal{N}(0, 0.1))$. To create well-separated points, we further added $x_i \leftarrow x_i + 0.7y_i \cdot \theta$, and chose $P = \operatorname{diag}(\mathrm{N}(0, 750))$.

Support vector data description Given a set of unlabeled points, $x_1, \ldots, x_m \in \mathbb{R}^n$, support vector data description (Tax & Duin, 2004; Chang et al., 2007) describes those points with an n-dimensional Euclidean ball by solving

$$\underset{\rho,a}{\text{minimize}} \quad \sum_{i=1}^{m} [\|x_i - a\|_2^2 - \rho)]_+ + \lambda[\rho]_+ \tag{53}$$

with optimization variables $\rho \in \mathbb{R}$ and $a \in \mathbb{R}^n$. The first term penalizes points outside a ball centered at a with radius $\sqrt{\rho}$ while the second term regularizes the radius with $\lambda \geq 0$ controlling the tradeoff. This problem is transformed to proximal form with operators

$$\begin{aligned} f_1(t, \rho) &= \sum_{i=1}^{m} [t_i]_+ + \lambda[\rho]_+ \\ f_3(a, s) &= \sum_{i=1}^{m} I(\|x_i - a\|_2^2 \leq s_i) \end{aligned} \tag{54}$$

and equality constraint

$$t = s - \rho. \tag{55}$$

In our experiments, we generate 5000 random points uniformly over the 200-dimensional unit hypersphere and then choose 100 outliers at random and add noise, $\epsilon \sim \mathcal{N}(0, I)$; we fit the model with $\lambda = 1$.

Robust Regression Consider a noisy matrix bounded by a unit ball over some known perturbation directions A_i,

$$\mathcal{A} = \{\bar{A} + c_1 A_1 + \ldots + c_p A_p \mid \|p\|_2 \leq 1\}. \tag{56}$$

The maximum error incurred from performing linear regression over the uncertain set can be written as

$$\begin{aligned} &\sup_{\|c\|_2\leq 1} \|(\bar{A} + c_1 A_1 + \ldots + c_p A_p)x - b\|_\infty \\ &= \max_k \left| \sup_{\|c\|_2\leq 1} c_k(A_k x) + (\bar{a}_k^T x - b_k) \right| \\ &= \max_k \left| \|A_k x\|_2 + |\bar{a}_k^T x - b_k| \right|. \end{aligned} \tag{57}$$

For robust regression (Boyd & Vandenberghe, 2004, pg. 323), we wish to find a solution x that minimize this worst possible error,

$$\underset{x}{\text{minimize}} \ \max_k \left| \|A_k x\|_2 + |\bar{a}_k^T x - b_k| \right|. \tag{58}$$

With additional variable $t, u, v, p, q \in \mathbb{R}^k$, this problem is transformed to proximal form with operators

$$\begin{aligned} f_1(t) &= \max_{i=1,\ldots,k} (t_i), \\ f_2(p, u) &= \sum_{i=1}^{k} \mathcal{I}(\|p_i\|_2 \leq u_i), \\ f_3(q, v) &= \sum_{i=1}^{k} \mathcal{I}(|q_i| \leq v_i), \end{aligned} \tag{59}$$

with equality constraints

$$\begin{aligned} t &= u + v, \\ p_i &= A_i x, \\ q_i &= \bar{a}_i^T x - b_i, \ \forall i = 1, \ldots, k. \end{aligned} \tag{60}$$

In the experiment, we generate the $\bar{A}$, A_i, and b from uniform distribution, then normalize $\bar{A}$ and A_i to a unit ball. We choose $p = 5000$ and $A, A_i \in \mathbb{R}^{10\times 200}$, for $i = 1, \ldots, p$.

sum-k-largest softmax The softmax loss is a multiclass loss function defined as

$$\text{softmax}(x, y, \Theta) = \frac{\exp(x^T \Theta_{y_i})}{\sum_k \exp(x^T \Theta_k)}, \tag{61}$$

where $\Theta \in \mathbb{R}^{n\times c}$ is the weight for each class. The softmax loss is commonly used in, for example, the regularized logistic regression, which can be formulated by the softmax loss plus a regularization term. Here, we consider to minimize the worst k loss incurred from the regression. I.e., we only minimize

$$\underset{\Theta}{\text{minimize}} \quad \sum_{i=1}^{k} z_{[i]} + \lambda\|\Theta\|_2^2, \tag{62}$$

where $z_{[i]}$ is the i-largest element of vector z, and $z_i = -\log \text{softmax}(x_i, y_i, \Theta)$ is the multiclass softmax loss. With additional variable $u_i \in \mathbb{R}^c, \ \forall i = 1, \ldots, m$, this problem is transformed to proximal form with operators

$$\begin{aligned} f_1(\Theta) &= -\sum_{i=1}^{m} x_i^T \Theta_{y_i} + \lambda\|\Theta\|_2^2, \\ f_2(z) &= \text{sum-}k\text{-largest}(z), \\ f_3(\Theta, z) &= \mathcal{I}(\text{log-sum-exp}(u_i) \leq z_i), \end{aligned} \tag{63}$$

Solver	Problem	Time	
		Epsilon	Solver
liblinear	hinge_l1	3.71s	0.49s
	hinge_l1_sparse	14.26s	4.26s
	hinge_l2	3.58s	0.16s
	hinge_l2_sparse	1.82s	0.83s
glmnet	lasso	3.69s	0.84s
	lasso_sparse	13.58s	0.67s
	logreg_l1	3.70s	2.31s
	logreg_l1_sparse	6.69s	1.96s
	mv_lasso	7.14s	7.40s
Gurobi	lp	0.33s	6.02s
	qp	1.39s	4.12s
QUIC	covsel	0.93s	6.24s

Table 3. Comparison of running times between Epsilon and specialized solvers.

with equality constraints

$$u_i = x_i^T \Theta, \; \forall i = 1, \ldots, m. \tag{64}$$

In the experiment we choose $X \in \mathbb{R}^{400 \times 10}$, $k = 5$, and the number of classes to be 120. We generate the data from normalized uniform distribution, and assign classes uniformly.

B. Comparison with specialized solvers

In this section, we compare Epsilon to an assortment of specialized solvers which are available for common problems. Before doing so, we emphasize that the general convex programming approach offers many advantages to specialized algorithms in terms of reuse and extensibility. In addition, most convex problems do not have dedicated, mature software packages readily available in common mathematical programming environments (e.g. Matlab, R, Python). Furthermore, even when specialized solvers are available, translating problems to the interface provided by a particular package requires effort to understand and conform to the idiosyncrasies of each implementation. In contrast, general convex programming offers a uniform syntax and interface allowing problems to be easily formulated, extended and solved.

In terms of running times, Table 3 compares Epsilon to four dedicated software packages implementing specialized algorithms: liblinear (Fan et al., 2008), glmnet (Friedman et al., 2010), Gurobi (Optimization et al., 2012) and QUIC (Hsieh et al., 2013). The default stopping criteria is used for each solver, corresponding to moderate accuracy for Epsilon and high accuracy for the specialized solvers. For the most part, Epsilon is competitive, although on a few problems liblinear and glmnet are significantly faster. This is due to the ability of these specialized algorithms to exploit sparsity in the solution which arises due to ℓ_1 regularization (Lasso problems) or a small number of support vectors in the dual SVM formulation (hinge problems). At present, Epsilon does not take advantage of such structure and thus may have a disadvantage on sparse problems.

On the other hand, Table 3 shows that Epsilon is significantly faster than Gurobi and QUIC in solving linear/quadratic programs and sparse inverse covariance estimation, respectively. However, it is important to highlight that the specialized algorithms solve these problems to high accuracy (e.g. tolerances of 10^{-8} or smaller) while Epsilon targets only moderate accuracy (e.g. 10^{-3}). For moderate accuracy, the operator splitting approach can be highly competitive, allowing Epsilon to be significantly faster on some problems.