Using the l2path package

Renxiong Liu

2021-08-15

The `l2path` package contains the implementations of path-following algorithms for both ridge regression and ℓ_2-regularized logistic regression. The goal is to approximate the solution path with high accuracy while saving the overall computation as much as possible. To this end, both the grid points and the optimization algorithm should be designed carefully to strike a balance between "optimization error" and "interpolation error". The details of our proposal can be found in Zhu and Liu (2021) *An algorithmic view of ℓ_2 regularization and some path-following algorithms*.

Using `l2main` functions

Generating approximated solution path by `l2path` packages requires to:

1. Specify the model class for implementation.
2. Specify the method for computing the whole path.

We elaborate these points by considering both ridge regression and ℓ_2-regularized logistic regression in our simulation studies.

Ridge regression

Data simulation

We consider the following linear model:

$$Y_i = X_i^\top \theta^\star + \epsilon\,,$$

where $X_i \sim N_p(0, I_{p\times p})$, $\epsilon \sim N(0,1)$, $\theta^\star = (1/\sqrt{p}, \ldots, 1/\sqrt{p})^\top$. We simulate $n = 100$ observations from above models with $p = 3$.

```r
n=100;
p=3;
X=matrix(rnorm(n*p), n, p);
Y=X%*%rep(1/sqrt(p), p)+rnorm(n);
```

Solution path generation

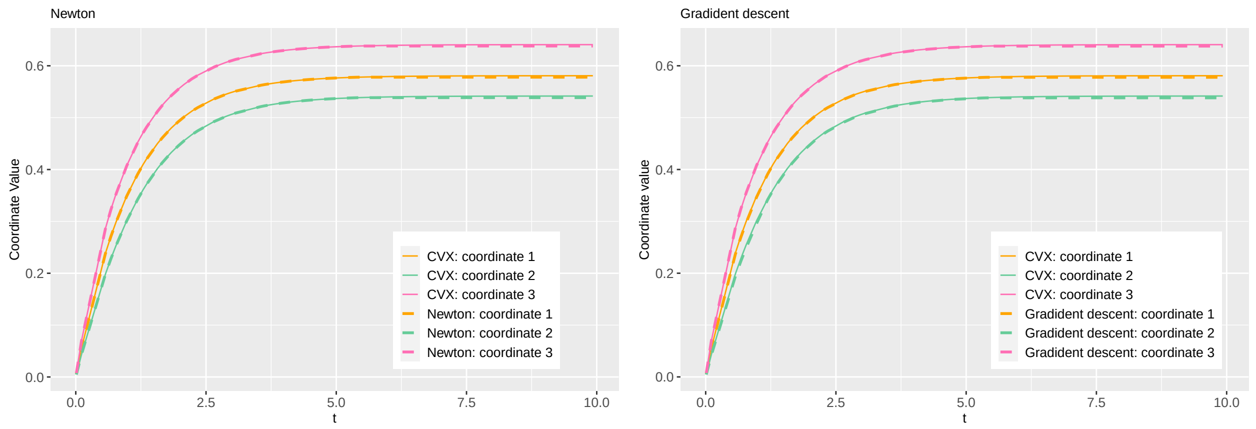
For above simulated dataset, we apply `l2main` function to compute approximated solution path $\tilde{\theta}(t)$ to the following problem:

$$\theta(t) = \arg\min_{\theta\in\mathbb{R}^p} \frac{(e^t-1)}{2n}\|Y - X\theta\|_2^2 + \frac{1}{2}\|\theta\|_2^2\,. \quad (1)$$

To this end, we need to set `class = "square"` for this ridge regression task and specify $t_{\max}$ to indicate that we want to approximate $\theta(t)$ over $[0, t_{\max}]$. Meanwhile, we can choose the methods to compute the solution, which include gradient descent method (`method = "GD"`) and Newton method (`method = "Newton_low"` when $n \geq p$ and `method = "Newton_high"` when $n < p$) for current version.

```
library(l2path);
t_max=10;
alpha_init=0.1;
path_Newton=l2main(X, Y, t_max, alpha_init, class = "square", method = "Newton_low");
path_GD=l2main(X, Y, t_max, alpha_init, class = "square", method = "GD");
```

To see how well our method can approximate the true solution path $\theta(t)$, we visualize our obtained solution $\tilde{\theta}(t)$ and $\theta(t)$ in the same plots. Note to obtain the following plots, you may need to install `ggplot2`, `latex2exp`, `reshape` and `plyr` packages.



We can clearly see that our computed $\tilde{\theta}(t)$ approximates true solution path $\theta(t)$ very well for both methods.

ℓ_2-regularized logistic regression.

Data simulation

We sample the components of the response vector $Y \in \mathbb{R}^n$ from a Bernoulli distribution, where $\mathbb{P}(Y_i = +1) = 1/2$ and $\mathbb{P}(Y_i = -1) = 1/2$ for $i = 1, 2, \ldots, n$. Conditioned on Y_i, we generate X_i's independently from $N_p(Y_i\mu, \sigma^2 I_{p\times p})$, where $\mu \in \mathbb{R}^p$ and $\sigma^2 > 0$. Note that μ and σ^2 controls the Bayes risk, which is $\Phi(-\|\mu\|_2/\sigma)$ under the 0/1 loss, where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal random variable. Here we choose $\mu = (1/\sqrt{p}, \ldots, 1/\sqrt{p})$ and $\sigma^2 = 1$ so that the Bayes risk is $\Phi(-1) \approx 0.15$. We simulate $n = 100$ observations from above models with $p = 3$.

```
n=100;
p=3;
X=matrix(rep(0,n*p),n,p);
mu=c(-1/(sqrt(p)),1/(sqrt(p)));
Y=rbinom(n,1,0.5);
mu_vec=mu[Y+1];
for(i in 1:n){
  X[i,]=c(rnorm(p,mu_vec[i],sd=1));
}
Y=2*Y-1;
```

Solution path generation

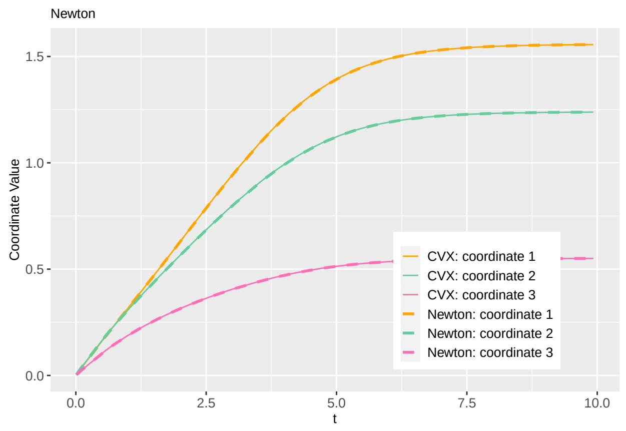
We apply `l2main` function to compute approximated solution path $\tilde{\theta}(t)$ to the following problem:

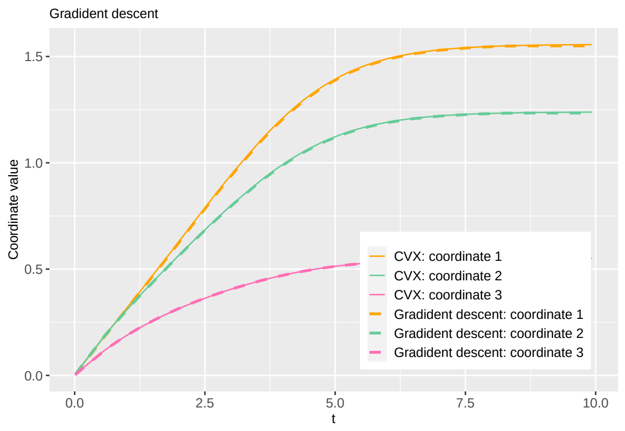
$$\theta(t) = \arg\min_{\theta\in\mathbb{R}^p} \frac{(e^t-1)}{n}\sum_{i=1}^{n}\log(1+e^{-Y_iX_i^\top\theta}) + \frac{1}{2}\|\theta\|_2^2\,. \tag{2}$$

To this end, we need to set `class = "logistic"` and specify $t_{\max}$ to indicate the range we are considering is $[0, t_{\max}]$. Similar to the regression case, we can choose the methods to compute the solution, which include gradient descent method (`method = "GD"`) and Newton method (`method = "Newton_low"` when $n \geq p$ and `method = "Newton_high"` when $n < p$) for current version.

```
library(l2path);
t_max=10;
alpha_init=0.1;
path_Newton=l2main(X, Y, t_max, alpha_init, class = "logistic", method = "Newton_low");
path_GD=l2main(X, Y, t_max, alpha_init, class = "logistic", method = "GD");
```

We visualize our obtained solution $\tilde{\theta}(t)$ and $\theta(t)$ as follows. In addition to the packages we mentioned in regression case, you may also need to install `CVXR` to compute the true solution path.





Again, we can see that our generated solution path approximate the true solution (computed by `CVXR`) very well.