

# LASSO/Poisson DML implementation

Thomas R. Covert, Yixin Sun

September 15, 2020

## Contents

<b>1</b>	<b>Setup from Chernozhukov et al - Finite Nuisance Parameter Approach</b>	<b>1</b>
1.1	The Linear Setting . . . . .	2
1.1.1	Implementation - LASSO . . . . .	3
1.2	The Poisson Setting . . . . .	4
1.2.1	Implementation - Poisson LASSO . . . . .	5
1.3	Lasso Specification . . . . .	6
<b>2</b>	<b>Concentrating Out Approach</b>	<b>7</b>
2.1	The Linear Setting . . . . .	7
2.1.1	Implementation - LASSO . . . . .	8
2.1.2	Implementation - Regression Forest . . . . .	9
2.2	The Poisson Setting . . . . .	10
2.2.1	Implementation - Poisson . . . . .	10
2.2.2	Implementation - Regression Forest . . . . .	11

## 1 Setup from Chernozhukov et al - Finite Nuisance Parameter Approach

Let  $\theta$  be the thing we care about and  $\beta$  be the nuisance parameters (location, time etc). The data is  $W = (Y, D, X)$  where  $Y$  is an outcome,  $D$  is the vector of stuff we care about and  $X$  is the stuff we don't

care about. The true values of  $\theta$  and  $\beta$ , denoted as  $\theta_0$  and  $\beta_0$ , fit the data best, in the sense that

$$(\theta_0, \beta_0) = \operatorname{argmax}_{\theta, \beta} E_{W} [l(W, \theta, \beta)]$$

where  $l(W, \theta, \beta)$  is some criterion (squared deviation, log likelihood etc).

The *Neyman Orthogonal Score*  $\psi$  is defined by:

$$\psi(W, \theta, \beta, \mu) = \frac{\partial}{\partial \theta} l(W, \theta, \beta) - \mu \frac{\partial}{\partial \beta} l(W, \theta, \beta)$$

The vector  $\mu$  above is defined by the hessian of this criterion function. Let  $J$  be:

$$J = \begin{pmatrix} J_{\theta, \theta} & J_{\theta, \beta} \\ J_{\beta, \theta} & J_{\beta, \beta} \end{pmatrix} = \frac{\partial}{\partial \theta \partial \beta} E_W \left[ \frac{\partial}{\partial \theta \partial \beta} l(W, \theta, \beta) \right]$$

Then we define  $\mu$  as  $\mu = J_{\theta, \beta} J_{\beta, \beta}^{-1}$ .

## 1.1 The Linear Setting

The linear regression, the function  $l$  is

$$l(W; \theta, \beta) = -\frac{(Y - D\theta - X'\beta)^2}{2}$$

and the necessary gradients needed for  $\psi$  are:

$$\partial \ell_{\theta}(W; \theta, \beta) = (Y - D\theta - X'\beta) D$$

$$\partial \ell_{\beta}(W; \theta, \beta) = (Y - D\theta - X'\beta) X$$

The entries in the Hessian matrix that we need to compute  $\mu$  are:

$$J_{\theta\beta} = -E [DX']$$

$$J_{\beta\beta} = -E [XX']$$

yielding this expression for  $\mu$ :

$$\mu = E [DX'] (E [XX'])^{-1}$$

The Neyman orthogonal score is then given by:

$$\psi(W; \theta, \eta) = (Y - D\theta - X'\beta) (D - \mu X)$$

### 1.1.1 Implementation - LASSO

These steps give a single point estimate,  $\hat{\theta}$  and an associated covariance matrix for a given split structure. See below for how we combine point estimates and covariance matrices across many split structures into a single point estimate/covariance matrix that should be less sensitive to the monte carlo nature of splitting.

1. Make  $k$  splits of the data into training and estimation sets. Default is  $k = 5$  folds in our implementation
2. For each  $k = 1, \dots, K$ , implement the following steps:
  - (a) In a **training** set  $k$ , use a linear LASSO of  $Y$  on  $D$  and  $X$  to select the covariates (making sure  $D$  is always included),  $\hat{X}_k$ . Then fit a linear regression of  $Y$  on  $\hat{X}_k$  and  $D$ , and let  $\hat{\beta}_k$  be the estimated coefficients on  $\hat{X}_k$ .
  - (b) In the **estimation** set  $k$ , compute  $s_k = \hat{X}_k \hat{\beta}_k$ .
  - (c) In the **training** set  $k$ , compute a linear LASSO of  $D_j$  on  $X$  to select covariates  $\tilde{X}_{k,j}$ , for each variable of interest. Fit a linear regression of  $D_j$  on  $\tilde{X}_{k,j}$ , and denote this  $\tilde{\mu}_{k,j}$ . Collect the  $\tilde{\mu}_{k,j}$  into a vector to form  $\tilde{\mu}_k$ .
  - (d) In the **estimation** set, construct

$$m_k = \tilde{X}_k \tilde{\mu}_k$$

3. Using the DML2 algorithm, for each  $k$ , construct the average of the moment:

$$\frac{1}{n} \sum_{i=1}^n (D_i - m_i)' (Y_i - s_i - D_i \theta')$$

Then average over each of these folds to get the final objective function, which we use to compute  $\hat{\theta}$  by minimizing squared deviations from zero (if  $D$  is univariate, we can just do root-finding). Note, this is also different from what STATA does. It seems like they compute this moment in one step using all the data, and ignore the hold out structure.

4. To get a covariance matrix for this estimate of  $\theta$ , we first compute  $J_0$  defined by:

$$\begin{aligned} J_0 &= \frac{\partial}{\partial \theta} E_W \left[ \psi(W, \hat{\theta}, \hat{\beta}) \right] \\ &= -E_W \left[ D'(D - m) \right] \end{aligned}$$

Next we compute  $\Psi$  :

$$\begin{aligned} \Psi &= E_W \left[ \psi(W, \theta, \tilde{\theta}, \hat{\beta}) \psi(W, \theta, \tilde{\theta}, \hat{\beta})' \right] \\ \psi(W, \hat{\theta}, \tilde{\theta}, \hat{\beta}) &= (Y_i - s_i - D_i \hat{\theta}') (D_i - m_i) \end{aligned}$$

And we can compute:

$$\hat{\text{Var}}(\hat{\theta}) = \frac{1}{n} J_0^{-1} \Psi J_0^{-1}$$

## 1.2 The Poisson Setting

In Poisson regression, the function  $l$  is

$$l(W; \theta, \beta) = Y(D\theta + X\beta) - \exp(D\theta + X\beta)$$

and its associated gradients needed for the definition of  $\psi$  are

$$\begin{aligned} \frac{\partial}{\partial \theta} l(W, \theta, \beta) &= (Y - \exp(D\theta + X\beta))D \\ \frac{\partial}{\partial \beta} l(W, \theta, \beta) &= (Y - \exp(D\theta + X\beta))X \end{aligned}$$

The entries in the Hessian matrix that we need to compute  $\mu$  are:

$$\begin{aligned} J_{\theta, \theta} &= -E \left[ D' D \exp(D\theta + X\beta) \right] \\ J_{\theta, \beta} &= -E \left[ D' X \exp(D\theta + X\beta) \right] \\ J_{\beta, \beta} &= -E \left[ X' X \exp(D\theta + X\beta) \right] \end{aligned}$$

yielding this expression for  $\mu$ :

$$\mu = E \left[ D' X \exp(D\theta + X\beta) \right] \left( E \left[ X' X \exp(D\theta + X\beta) \right] \right)^{-1}$$

This construction is revealing, since it looks like weighted least squares, with  $D$  as the outcome,  $X$  as the covariates, and weights equal to  $\exp(D\theta + X\beta)$ .

The Neyman Orthogonal moment for Poisson regression is then:

$$\psi = (Y - \exp(D\theta + X\beta))(D - X\mu)$$

### 1.2.1 Implementation - Poisson LASSO

1. Make  $k$  splits of the data into training and estimation sets. Default is  $k = 5$  in our implementation.
2. For each  $k = 1, \dots, K$ , implement the following steps:
  - (a) In a **training** set  $k$ , use Poisson LASSO of  $Y$  on  $D$  and  $X$  to select the covariates,  $\hat{X}_k$  (making sure  $D$  is always included). Then fit a Poisson regression of  $Y$  on  $\hat{X}_k$  and  $D$ , and let  $\hat{\beta}_k$  be the estimated coefficients on  $\hat{X}_k$ .
  - (b) In the corresponding **estimation** set  $k$ , compute  $s_k = \hat{X}_k \hat{\beta}_k$ .
  - (c) Back in the **training** set  $k$ , compute weights  $w_k = \exp(D\hat{\theta}_k + X\hat{\beta}_k)$  using the results from step 2. Compute a linear LASSO of  $D_j$  on  $X$  using those weights to select covariates  $\tilde{X}_{k,j}$  for each variable of interest. Fit a linear regression of  $D_j$  on  $\tilde{X}_{k,j}$ , and denote this  $\tilde{\mu}_{k,j}$ . Collect the  $\tilde{\mu}_{k,j}$  into a vector to form  $\tilde{\mu}_k$ .
  - (d) In **estimation** set, construct

$$m_k = \tilde{X}_k \tilde{\mu}_k$$

3. Using the DML2 algorithm, for each  $k$ , construct the average of the moment:

$$\frac{1}{n} \sum_{i=1}^n (D_i - m_i)' (Y_i - \exp(s_i - D_i \theta'))$$

Then average over each of these folds to get the final objective function, which we use to compute  $\hat{\theta}$  by minimizing squared deviations from zero (if  $D$  is univariate, we can just do root-finding).

4. To get a covariance matrix for this estimate of  $\theta$ , we first compute  $J_0$ , defined by:

$$\begin{aligned} J_0 &= \frac{\partial}{\partial \theta} E_W \left[ \psi(Y, D, X, \hat{\theta}, \tilde{\theta}, \tilde{\beta}) \right] \\ &= -E_W \left[ D' \exp(D\hat{\theta} + s)(D - X\hat{\mu}) \right] \end{aligned}$$

Next we compute  $\Psi$ :

$$\begin{aligned} \Psi &= E_W \left[ \psi(W, \hat{\theta}, \tilde{\theta}, \tilde{\beta}) \psi(W, \hat{\theta}, \tilde{\theta}, \tilde{\beta})' \right] \\ &= E_W \left[ (Y - \exp(D\hat{\theta} + s))^2 (D - X\hat{\mu})(D - X\hat{\mu})' \right] \end{aligned}$$

In both cases, we compute each of these as the average over points in the estimation set  $k$ , and then average over each of the estimation sets within a split structure.<sup>1</sup> We then get:

$$\hat{Var}(\hat{\theta}) = \frac{1}{n} J_0^{-1} \Psi J_0^{-1}$$

### 1.3 Lasso Specification

When using the `dml` function, the default machine learning method is a conventional lasso, implemented using `cv.glmnet` from the `glmnet` package. The user also has the option to use `ml = "hal"`, which uses functions from `hal9001`, an R package that implements the scalable highly adaptive lasso (HAL). HAL is a nonparametric regression estimator that applies L1-regularized lasso regression to a design matrix composed of indicator functions corresponding to the support of the functional over a set of covariates and interactions. Intuitively, we are creating a matrix composed of basis functions based on the covariates. Using the empirical CDFs of the covariates, we create a matrix consisting of indicator basis functions (generating dummy variables from the covariate). This results in a large, sparse matrix with binary entries. Many basis functions are created, and we can use the usual lasso methods to select the useful basis functions. For both methods, the user can specify the polynomial degree of the covariates that the lasso is implemented on (Coyle et al., 2020).

---

<sup>1</sup>For an example of this averaging, see the formula for  $\hat{J}_0$  on page C27 of Chernozhukov et al. (2018)

## 2 Concentrating Out Approach

Chernozhukov et. al write that the approach for constructing Neyman orthogonal scores is closely related to the “concentrating-out approach”. For all  $\theta \in \Theta$ , let  $\beta_\theta$  be the solution of the following optimization problem:

$$\operatorname{argmax}_{\beta \in \mathcal{B}} E[\ell(W; \theta, \beta)]$$

where  $\beta_\theta$  satisfies

$$\partial_\beta E[\ell(W; \theta, \beta_\theta)] = 0$$

Differentiating this with respect to  $\theta$  and interchanging the order of differentiation gives us

$$\begin{aligned} 0 &= \partial_\theta \partial_\beta E[\ell(W; \theta, \beta_\theta)] = \partial_\beta \partial_\theta E[\ell(W; \theta, \beta_\theta)] \\ &= \partial_\beta E[\partial_\theta \ell(W; \theta, \beta_\theta) + [\partial_\theta \beta_\theta]' \partial_\beta \ell(W; \theta, \beta_\theta)] \\ &= \partial_\beta E[\psi(W; \theta, \beta, \partial_\theta \beta_\theta)]|_{\beta=\beta_\theta} \end{aligned}$$

So our score function here is:

$$\psi(W; \theta, \beta, \partial_\theta \beta_\theta) := \partial_\theta \ell(W; \theta, \beta) + [\partial_\theta \beta_\theta]' \partial_\beta \ell(W; \theta, \beta)$$

### 2.1 The Linear Setting

Consider again the function:

$$\ell(W; \theta, \beta) = -\frac{(Y - D\theta - \beta(X))^2}{2}$$

Taking FOC with respect to  $\beta$ , we get:

$$\begin{aligned} 0 &= E[Y - D\theta - \beta(X)|X] \\ 0 &= E[Y - D\theta|X] - \beta(X) \\ \Rightarrow \beta(X) &= E[Y - D\theta|X] \end{aligned}$$

Letting  $g(x) = E[Y|X]$  and  $m(x) = E[D|X]$ , we get the following Neyman orthogonal score

$$\begin{aligned}\psi(W; \theta, \beta_\theta) &= -\frac{1}{2} \frac{d \{Y - D\theta - E[Y - D\theta|X]\}^2}{d\theta} \\ &= (D - E[D|X]) \times (Y - E[Y|X] - (D - E[D|X])\theta) \\ &= (D - m(X)) \times (Y - g(x) - (D - m(x))\theta)\end{aligned}$$

Note that we now use ML methods to calculate  $E[D|X]$  and  $E[Y|X]$ , which we plug into the score function we derived above. This is different from the finite-nuisance parameter approach, which uses ML to calculate the finite-nuisance parameter component,  $s = X\beta$ . The implication of this is that the ML steps for the concentrating-out approach are fairly uniform across ML methods and model types; as you can see below, what changes is just the score function that we are optimizing over.

### 2.1.1 Implementation - LASSO

1. Make  $k$  splits of the data into training and estimation sets. Default is  $k = 5$  folds in our implementation.
2. For each  $k = 1, \dots, K$ , implement the following steps:
  - (a) In a **training** set  $k$ , use a linear LASSO to generate a model of  $Y$  on and  $X$  to select the covariates,  $\hat{X}_k$ . Then fit a linear regression of  $Y$  on  $\hat{X}_k$ , and let  $\hat{\beta}_k$  be the estimated coefficients on  $\hat{X}_k$ .
  - (b) In the **estimation** set  $k$ , compute  $g_k = E[Y|X]$  using the model generated in the previous step.
  - (c) In the **training** set  $k$ , compute a linear LASSO of  $D_j$  on  $X$  to select covariates  $\tilde{X}_{k,j}$  for each variable of interest. Fit a linear regression of  $D_j$  on  $\tilde{X}_{k,j}$ , and denote the estimated coefficients as  $\tilde{\mu}_{k,j}$ . Collect the  $\tilde{\mu}_{k,j}$  into a vector to form  $\tilde{\mu}_k$ .
  - (d) In the **estimation** set, compute  $m_k = E[D|X]$ , using the model generated from the previous step.
3. Using the DML2 algorithm, for each  $k$ , construct the average of the moment:

$$\frac{1}{n} \sum_{i=1}^n (D_i - m_i)' (Y_i - g_i - (D_i - m_i)\theta')$$



Then average over each of these folds to get the final objective function, which we use to compute  $\hat{\theta}$  by minimizing squared deviations from zero.

4. To get a covariance matrix for this estimate of  $\theta$ , we first compute  $J_0$  defined by:

$$\begin{aligned} J_0 &= \frac{\partial}{\partial \theta} E_W [\psi(W, \hat{\theta}, \hat{\beta})] \\ &= -E_W [(D - m)'(D - m)] \end{aligned}$$

Next we compute  $\Psi$  :

$$\begin{aligned} \Psi &= E_W [\psi(W, \theta, \tilde{\theta}, \hat{\beta}) \psi(W, \theta, \tilde{\theta}, \hat{\beta})'] \\ \psi(W, \hat{\theta}, \tilde{\theta}, \hat{\beta}) &= (Y_i - g_i - (D_i - m_i)\hat{\theta}')(D_i - m_i) \end{aligned}$$

And we can compute:

$$\hat{\text{Var}}(\hat{\theta}) = \frac{1}{n} J_0^{-1} \Psi J_0^{-1}$$

### 2.1.2 Implementation - Regression Forest

1. Make  $k$  splits of the data into training and estimation sets. Default is  $k = 5$  folds in our implementation
2. For each  $k = 1, \dots, K$ , implement the following steps:
  - (a) In **training** set  $k$ , use a regression forest to generate a model of  $Y$  on and  $X$ . Use this trained regression forest and apply to data in the **estimation** set to calculate  $g_k = E[Y|X]$ .
  - (b) In **training** set  $k$ , use a regression forest to generate a model of  $D_j$  on  $X$ . Use this trained regression forest and apply to data in the **estimation** set to calculate  $m_{k,j} = E[D_j|X]$ . Collect to form  $m_k = (m'_{k,1}, m'_{k,2}, \dots, m'_{k,J})$
3. The rest of the implementation should follow steps 3 and 4 of the lasso instructions in the previous section.

## 2.2 The Poisson Setting

We have only solved the orthogonal score for the case when  $D$  is a univariate binary variable. Consider again for the regression case, the function  $l$  is

$$l(W; \theta, \beta) = Y(D\theta + X\beta(X)) - \exp(D\theta + \beta(X))$$

Taking FOC with respect to  $\beta$ , we get:

$$\begin{aligned}\beta_\theta(X) &= E[\exp(D\theta + \beta(X))|X] \\ &= \exp(\beta(X))E[\exp(D\theta)|X] \\ &= \exp(\beta(x)) (Pr(D = 1|X) \exp(\theta) + (1 - Pr(D = 1|X))) \\ &= \exp(\beta(x)) (E[D|X] \exp(D\theta) + (1 - E[D|X]))\end{aligned}$$

As before, let  $g(X) = E[Y|X]$  and  $m(X) = E[D|X]$  and rearrange to get:

$$\exp(\beta_\theta(X)) = \frac{g(X)}{\exp(\theta)m(X) + 1 - m(X)}$$

From this, we have the following:

$$\begin{aligned}\beta_\theta(X) &= \log \left( \frac{g(X)}{\exp(\theta)m(X) + 1 - m(X)} \right) \\ \partial_\theta \beta_\theta &= - \frac{m(X) \exp(\theta)}{m(X) \exp(\theta) + (1 - m(x))}\end{aligned}$$

Putting this altogether and plugging into the score function, we get:

$$\psi(W; \theta, g(X), m(X)) = \left( Y - \frac{\exp(D\theta)g(X)}{\exp(\theta)m(X) + 1 - m(X)} \right) \left( D - \frac{\exp(\theta)m(X)}{\exp(\theta)m(X) + 1 - m(X)} \right)$$

### 2.2.1 Implementation - Poisson

Steps 1 and 2 are the same as the linear model in the previous section.

3. Using the DML2 algorithm, for each  $k$ , construct the average of the moment:

$$\frac{1}{n} \sum_{i=1}^n \left( Y_i - \frac{\exp(D_i\theta)g_i}{\exp(\theta)m_i + 1 - m_i} \right) \left( D_i - \frac{\exp(\theta)m_i}{\exp(\theta)m_i + 1 - m_i} \right)$$

4. To get a covariance matrix for this estimate of  $\theta$ , we first compute  $J_0$  defined by

$$J_0 = \frac{\partial}{\partial \theta} E_W \left[ \psi(W, \hat{\theta}, \hat{\beta}) \right]$$

To break down this calculation, let

$$P_i = \exp(\theta)m_i + 1 - m_i$$

$$A_i = \frac{\exp(D_i\theta)g_i}{P}$$

$$B_i = \frac{\exp(\theta)m_i}{P}$$

So we have

$$J_0 = E_W \left[ A \times \left( \frac{\exp(\theta)m}{P} + \frac{\exp(\theta)^2 g^2}{P^2} \right) + B \times \left( \frac{D \exp(D\theta)g}{P} + \frac{\exp(D\theta)g}{P^2} \right) \right]$$

As always, we have

$$\Psi = E_W \left[ \psi(W, \theta, \tilde{\theta}, \hat{\beta}) \psi(W, \theta, \tilde{\theta}, \hat{\beta})' \right]$$

And we can compute

$$\hat{\text{Var}}(\hat{\theta}) = \frac{1}{n} J_0^{-1} \Psi J_0^{-1}$$

### 2.2.2 Implementation - Regression Forest

Steps 1 and 2 follow the same steps as the instructions in section 2.1.2, while steps 3 and 4 follow the Poisson LASSO in the previous section.

## References

**Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins**, “Double/debiased machine learning for treatment and structural parameters,” *The Econometrics Journal*, February 2018, 21 (1), C1–C68.

**Coyle, Jeremy R, Nima S Hejazi, and Mark J van der Laan,** “hal9001: The Scalable Highly Adaptive Lasso,” June 2020.