Incorporating known risk factors into models

Jiaxuan Wang

<2017-02-23 Thursday>

1 objective

Incorporating known risk factors with unknown risk factors in predicting outcome. In the case of choosing between correlated variables, the model should favor known risk factors.

2 measuring success

Fixing the level of performance, the task of learning is to allocate weights to features so that desired structures are kept. A familiar example is lasso regression where the diamond shaped contour forces weights to be sparse so that they are interpretable in a sense. Another example is ridge regression where it corresponds to having a gaussian prior on weights. Learning is really a combination of data and domain expertise. Each model implicitly carries assumptions into the learning task and there's no universal consistent algorithm that works best under all distributions (no free lunch thm). However, there is one thing that for sure we want no matter what assumptions we make, that is the credibility of the model. In addition to having good performance, we want the model to be consistent with the current literature. More concretely, we want the model to place high weights on relevant and known features while keeping the unknown relevant features sparse. This whole process should be done in a data driven way so that the known risk factors are merely suggestions for the model to consider instead of forced constraints. In this spirit, we list the following properties of a credible model:

a) credibility should not come at the cost of performance b) irrelevant features whether known or unknown should have low weights c) within a group of dependent features, known risk factors should be given more weights and the their weights should be dense d) within a group of dependent features of all unknown risk factors, the weights should be sparse

criteria a) are acheived by grid searching over validation set so that models in consideration has similar level of performance. b) is acheive by constraining on the size of parameters which all regularizations do.

For c) and d) we measure the KL divergence in each group of dependent features.

Here I give a typical exmaples of what I mean by measuring KL divergence in a group of dependent features

assume $r = [1, 1, 0, 0]^T$ and $\theta = [0.1, 0.2, -0.01, 0.02]^T$ (θ excluding b term), we first normalize each vector so that their $||\cdot||_1$ is 1.

$r' = [0.5, 0.5, 0, 0]^T$, $\theta' = [0.32258065, 0.64516129, 0.03225806, 0.06451613]^T$

To avoid 0 appearing in log of KL divergence calculation, a small smooth factor of 1e-6 is added to any vector with 0, renormalizing giving

$r'' = [4.99999000e{-}01, 4.99999000e{-}01, 9.99996000e{-}07, 9.99996000e{-}07]^T$, $\theta'' = [0.32258065, 0.64516129, 0.03225806, 0.06451613]^T$

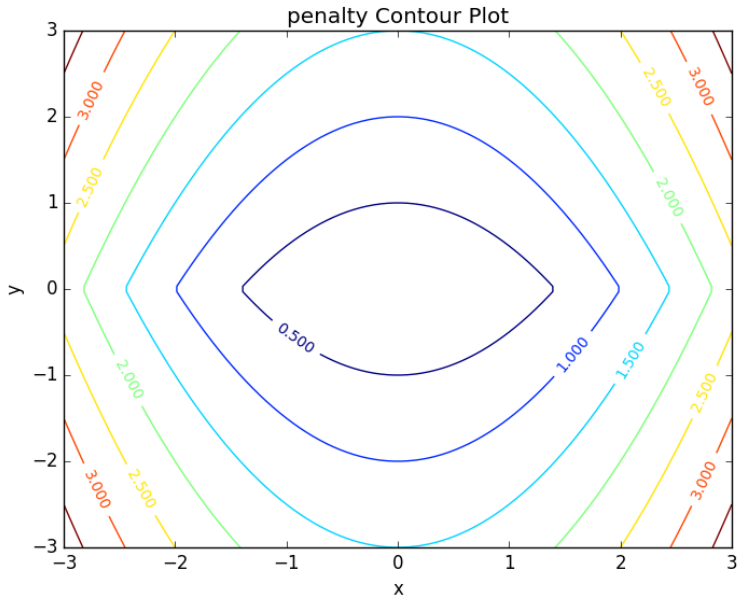
Then $KL(r''||\theta'')$ is the reported result in each dependent group, where $KL(x||y) = \sum_i p(x_i) \log \frac{p(x_i)}{p(y_i)}$

In the case where r is all 0 in relevant feature group, I give $min_{v \in one\ hot\ vectors} KL(v||\theta'')$ as a loss as to encourage sparse feature.

3 approaches taken

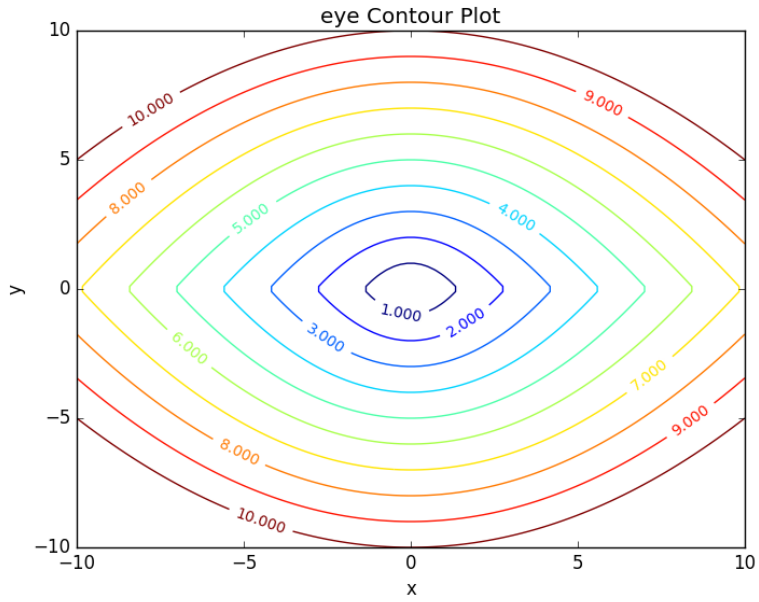
3.1 old approach

0.5 * λ_2 $||\text{r} * \theta||_2^2$ + λ_1 $||(1\text{-r}) * \theta||_1$ where r $\in \{0,1\}^{\text{d}}$, $\theta \in \text{R}^{\text{d}}$

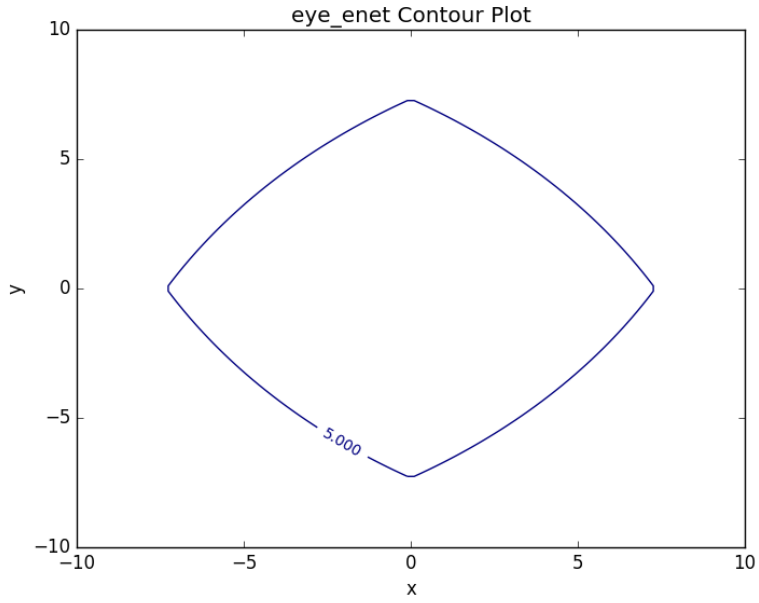


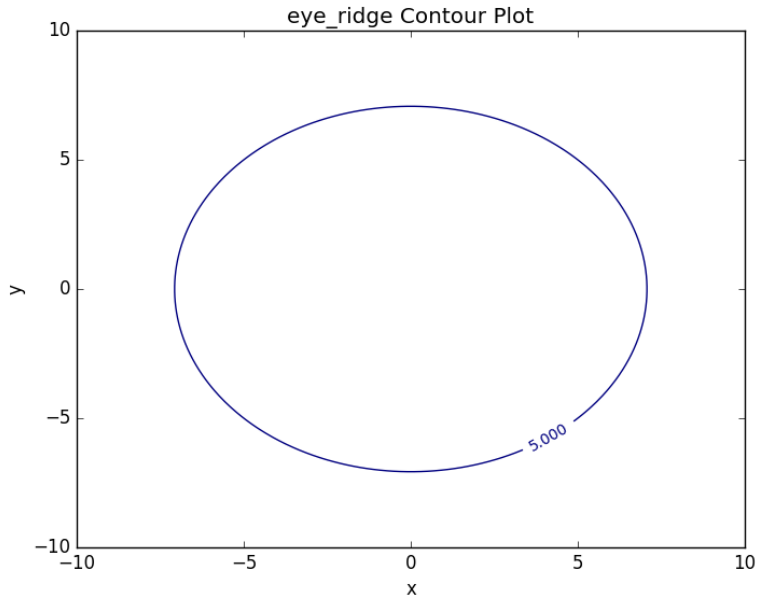
3.2 new approach

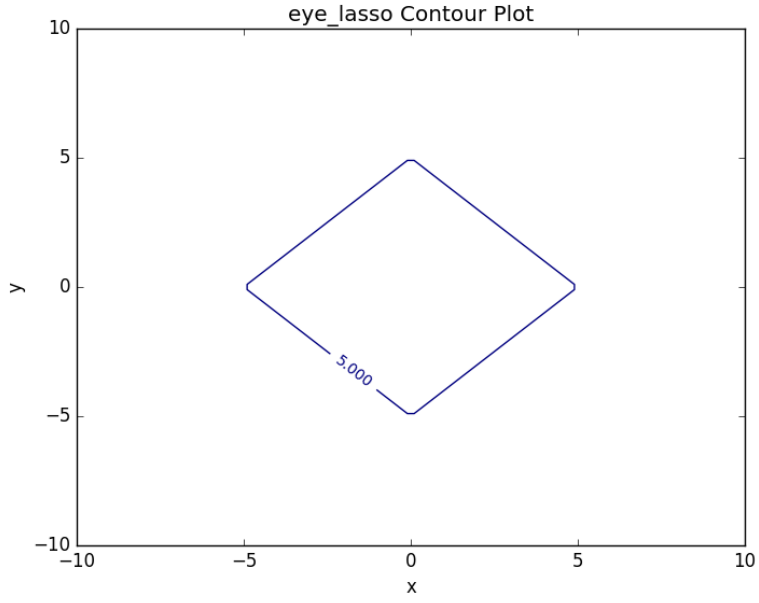
Fix a convex body that have property of the previous contour plot such that the angle at the end point is 45 degree. The following is the contour plot of its induced norm



In fact, by relaxing the constraint of r over binary to float, we can recover enet (setting r=0.5 * 1). Even without extending r, we can recover ridge (r=1) and lasso (r=0)




eye_ridge Contour Plot
10
5
0
−5
−10
y
−10
−5
0
5
10
x
5.000


eye_lasso Contour Plot
10
5
0
−5
−10
y
−10
−5
0
5
10
x
5.000

4 experiments

4.1 set up

4.1.1 nd data generation (genPartitionData)

Data n = 5000

n relevant groups (nrgroups) = 11

n irrelevant group (nirgroups) = 11

correlated variables pergroup (npergroup) = 10

h_i = Uniform(-3, 1, n)

$\theta_i = 1 \; \forall$ i

$x_{i,j}$ ~ Uniform(1..2) h_i + N(0, 0.2) for i $\in$ [n] for j $\in$ [npergrop]

y = $\frac{\sum_{i=1}^{nrgroups} h_i \theta_i}{\sum_{i=1}^{nrgroups} |\theta_i|} >$ -1

r (known risk factors): for each correlated variable group, putting in one more known risk factor than the previous group

Loss function is the negative loss likelihood of the logistic regression model.

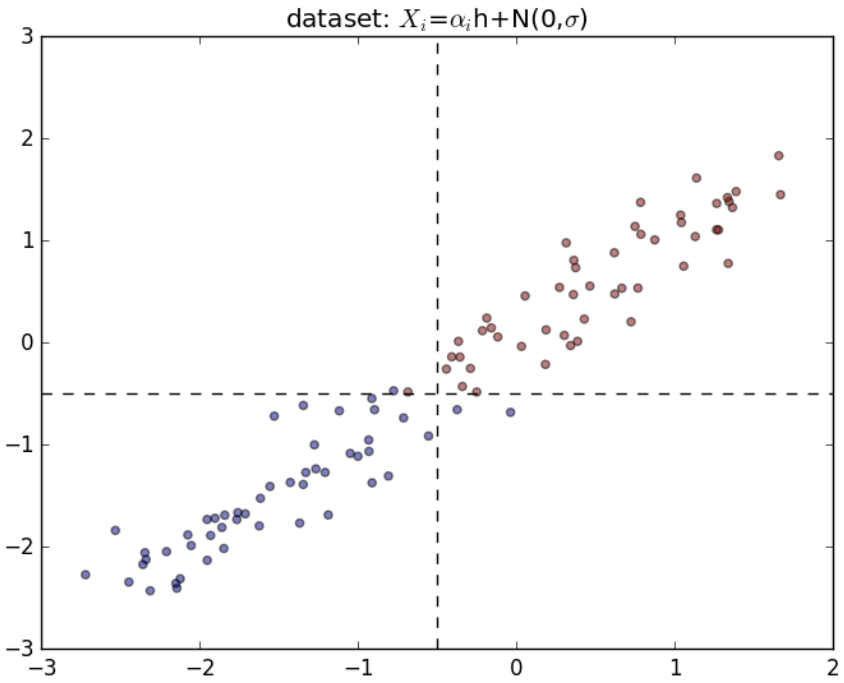
Optimizer: AdaDelta

Number of Epoch: 1000

Regulizers: elastic net, lasso, ridge, OWL, weighted lasso, weighted ridge, eye penalty

4.1.2 2d data generation

Data n = 100:



h = linspace(-2.5, 1, n)

x_0 ~ Uniform(1..4) h + N(0, 0.2)

x_1 ~ Uniform(1..4) h + N(0, 0.2)

y = h > 0.5

r (known risk factors) = [1, 0]

Loss function is the negative loss likelihood of the logistic regression model.

Optimizer: AdaDelta

Number of Epoch: 1000

Regulizers: elastic net, lasso, ridge, OWL, weighted lasso, weighted ridge, penalty, eye penalty

4.1.3 eye penalty

$q(\theta) := 2\ \beta\ ||(1\text{-r}) * \theta||_1 + \$\ (1\text{-}\beta)\ ||r*\theta||_2^2$

$pena(\theta) := \alpha\ q(\theta)$

where $r \in \{0,1\}^d$, $\theta \in R^d$, $\alpha \in R_+$, $\beta \in (0,1)$ (β is also called $l1_{ratio}$ in this text)

For any constant c

$pena(\theta) = c$

is convex because pena is convex (addition of positively weighted norms)

similarly, q(θ) = c is also convex

c can be chosen so that slope in the first quadrant between known risk factor x and unknown risk factor is -1

we define eye norm as a an atomic norm $||\cdot||_A$ as introduced in Venkat et al.

$||x||_A := \inf\{t > 0 | x \in tconv(A)\}$

Let $A = \{x|q(x) = \frac{\beta^2}{1-\beta}\}$, we get the eye penalty

Note that A is already a convex set, equivalently we write

$eye(x) = \inf\{t > 0 | x \in t\{x|q(x) = \frac{\beta^2}{1-\beta}\}\}$

1. derivation

   The main intuition is to set c so that the slope in the first quadrant between known risk factor x and unknown risk factor is -1. Since we only care about this interaction between known and unknown risk factors and that {x|pena(x)=c} is symmetric about origin, WLOG, we let y be the unknown feature and x be the known risk factor with constraint y $\geq$ 0, x $\geq$ 0.

$$\alpha[2\beta y + (1-\beta)x^2] = c \tag{1}$$

$$\rightarrow 2\beta y + (1-\beta)x^2 = \frac{c}{\alpha} \tag{2}$$

$$\rightarrow y = \frac{c}{2\alpha\beta} - \frac{(1-\beta)x^2}{2\beta} \tag{3}$$

$$\rightarrow y = 0 \Rightarrow x = \sqrt{\frac{c}{\alpha(1-\beta)}} \tag{4}$$

$$\rightarrow f'(x) = -\frac{(1-\beta)}{\beta}x \tag{5}$$

$$\rightarrow f'(\sqrt{\frac{c}{\alpha(1-\beta)}}) = -\frac{1-\beta}{\beta}\sqrt{\frac{c}{\alpha(1-\beta)}} = -1 \tag{6}$$

$$\rightarrow c = \frac{\alpha\beta^2}{1-\beta} \tag{7}$$

$$\rightarrow 2\beta y + (1-\beta)x^2 = \frac{\beta^2}{1-\beta} \tag{8}$$

   Thus, we just need q(x) = $\frac{\beta^2}{1-\beta}$

2. properties: a) A is symmetric about origin (x $\in$ A then -x $\in$ A), so this is a norm

   (a) eye(t θ) = |t| eye(θ)

   (b) eye(θ + β) $\leq$ eye(θ) + eye(β)

   (c) eye(θ) = 0 iff θ = 0

   b) β doesn't affect the shape of contour, so no need to search over β

   proof:

   consider the contour B_1 = {x: eye_{β_1}}(x) = t} and B_2 = {x: eye_{β_2}}(x) = t}

   We want to show B_1 is similar to B_2

   case1: t = 0, then B_1 = B_2 = {0} by property a3

   case2: t $\neq$ 0

   we can equivalently write B_1 and B_2 as: (by definition and a1 and q convex)

   B_1 = t {x: x $\in$ {x | q_{β_1}(x) = $\frac{\beta_1^2}{1-\beta_1}$ }}

   B_2 = t {x: x $\in$ {x | q_{β_2}(x) = $\frac{\beta_2^2}{1-\beta_2}$ }}

   let B_1' = {x: x $\in$ {x | q_{β_1}(x) = $\frac{\beta_1^2}{1-\beta_1}$ }} and B_2' = {x: x $\in$ t {x | q_{β_2}(x) = $\frac{\beta_1^2}{1-\beta_2}$ }}

   Claim: B_2' = $\frac{\beta_2(1-\beta_1)}{\beta_1(1-beta_2)}$ B_1'

   It should be clear that if this claim is true then B_1 is similar to B_2 and we are done

   take x $\in$ B_1'

   then q_{β_1}(x) = 2 β_1 ||(1-r) * x||$_1$ + (1-β_1) ||r*x||$_2^2$ = $\frac{\beta_1^2}{1-\beta_1}$

   let x' = $\frac{\beta_2(1-\beta_1)}{\beta_1(1-\beta_2)}$ x

$$q_{\beta_2}(x') = 2\beta_2||(1-r)*x'||_1 + (1-\beta_2)||r*x'||_2^2 \tag{9}$$

$$= \frac{2\beta_2^2(1-\beta_1)}{\beta_1(1-\beta_2)}||(1-r)*x||_1 + \frac{\beta_2^2(1-\beta_1)^2}{\beta_1^2(1-\beta_2)}||r*x||_2^2 \tag{10}$$

$$= \frac{\beta_2^2(1-\beta_1)}{\beta_1^2(1-\beta_2)}(2\beta_1||(1-r)*x||_1 + (1-\beta_1)||r*x||_2^2) \tag{11}$$

$$= \frac{\beta_2^2(1-\beta_1)}{\beta_1^2(1-\beta_2)}\frac{\beta_1^2}{1-\beta_1} \tag{12}$$

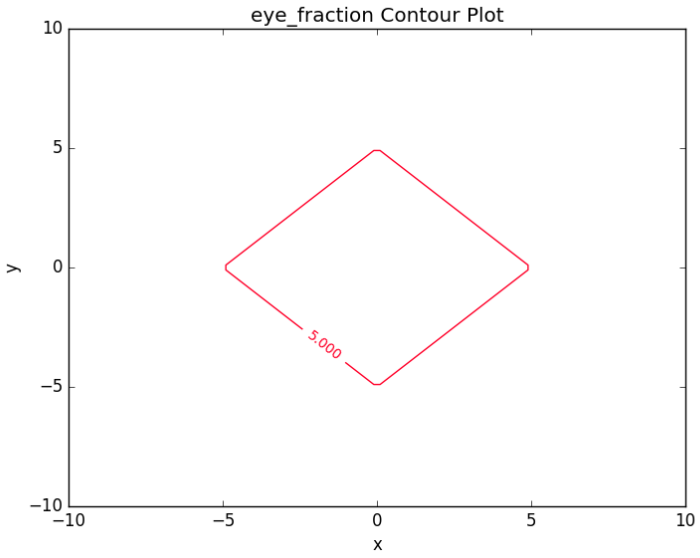
$$= \frac{\beta_2^2}{1-\beta_2} \tag{13}$$

so x' $\in$ $B_{2'}$. Thus $\frac{\beta_2(1-\beta_1)}{\beta_1(1-beta_2)}$ $B_{1'} \subset B_{2'}$. The other direction is similarly proven.

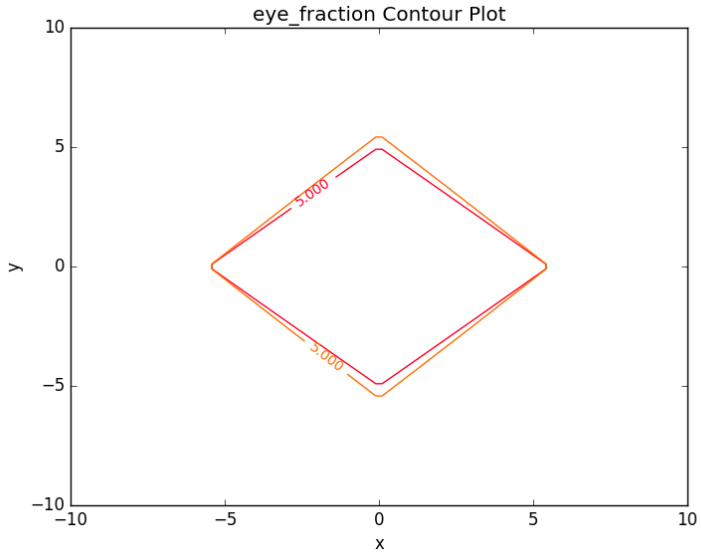
3. extending r to $[0,1]^d$ At times, it makes sense for risk factor to be fractionally weighted (eg. odds ratio in medical documents)

   varying r_1 and r_2

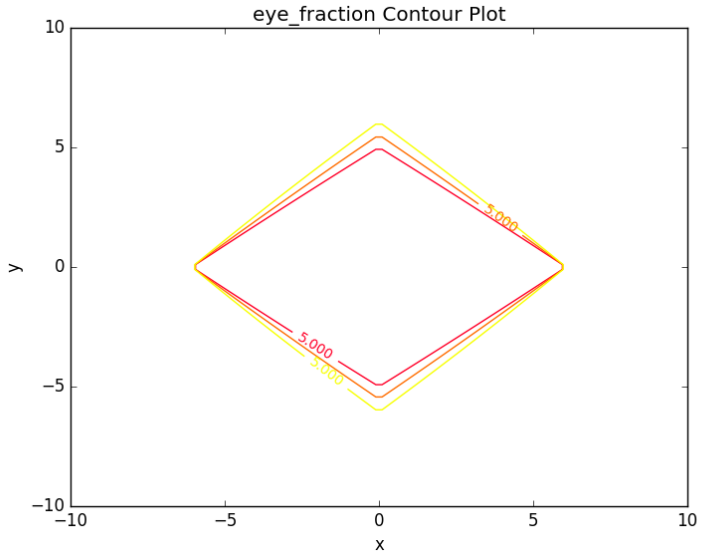
   $r_1 = 0.0$



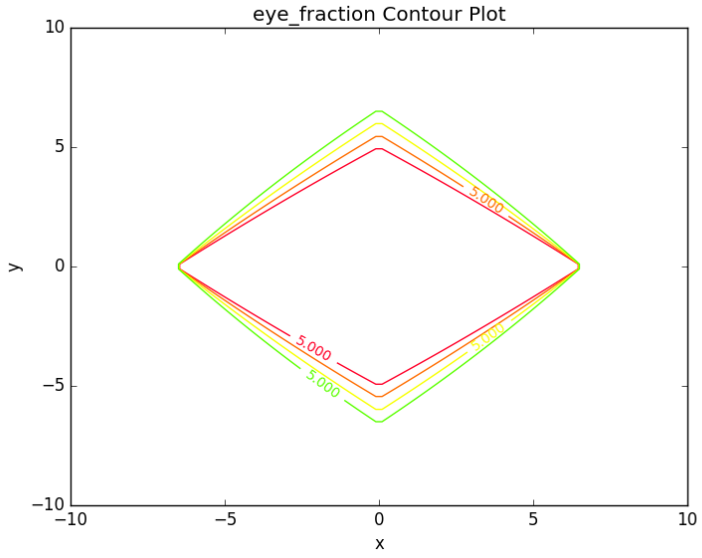
   $r_1 = 0.1$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000

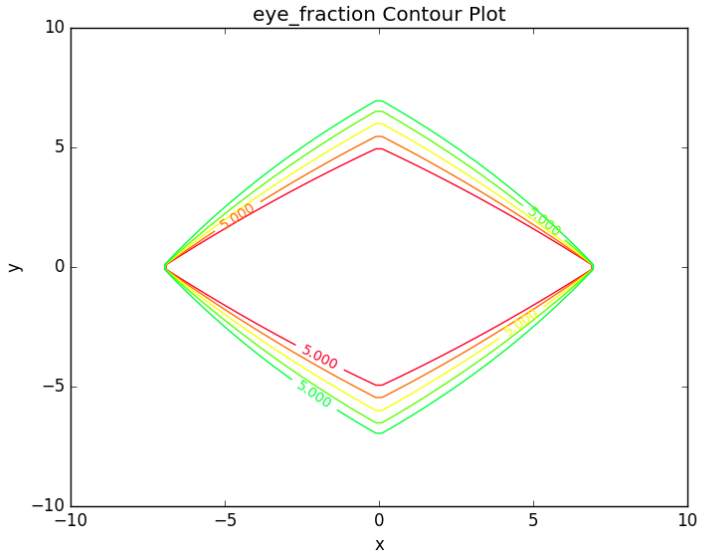
$r_1 = 0.2$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000
5.000

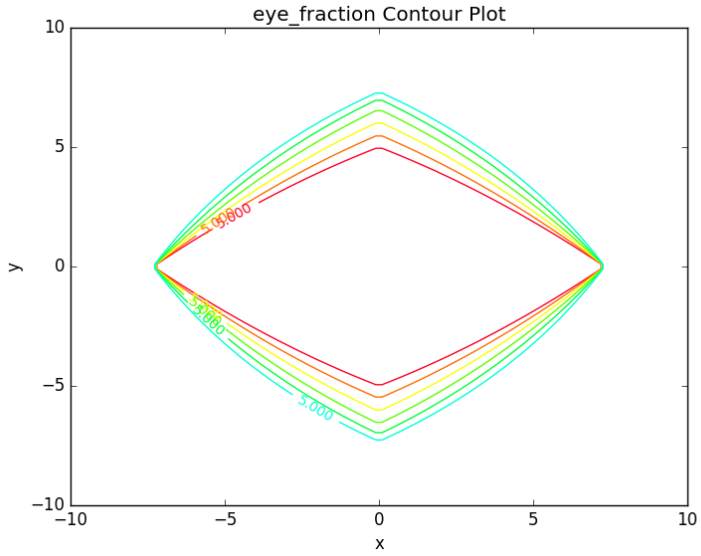
$r_1 = 0.3$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000
5.000
5.000
5.000

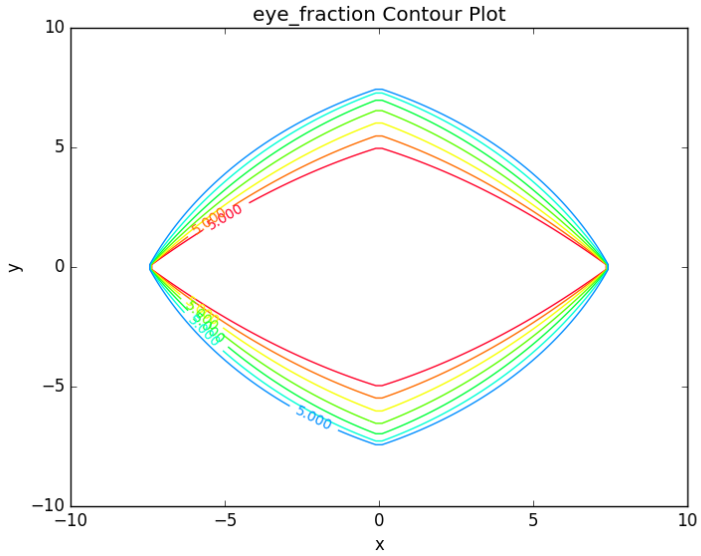
$r_1 = 0.4$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000
5.000
5.000
5.000

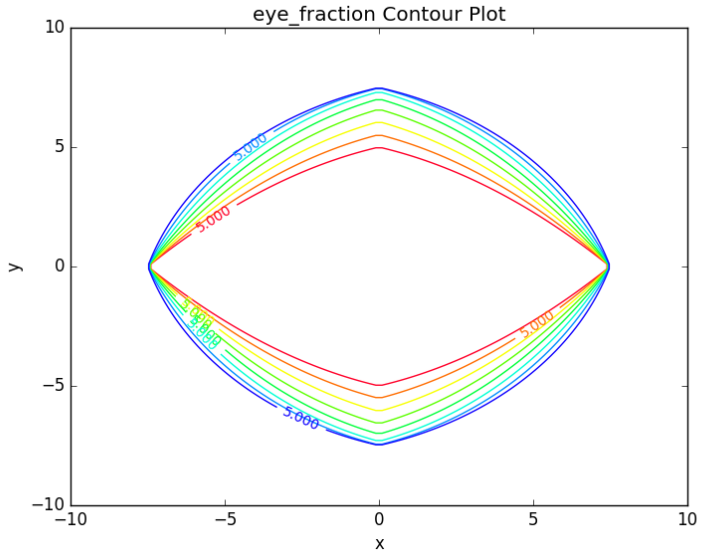
$r_1 = 0.5$


eye_fraction Contour Plot
10
5
0
−5
−10
y
−10
−5
0
5
10
x
5.000

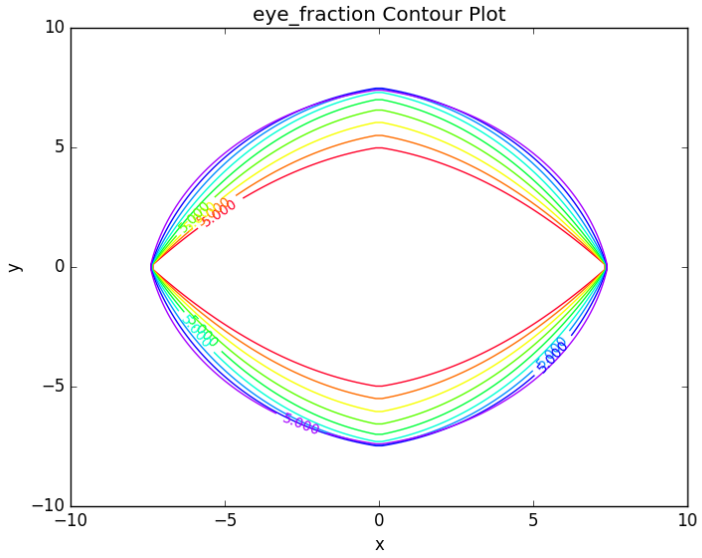
$r_1 = 0.6$


eye_fraction Contour Plot
10
5
0
−5
−10
y
−10
−5
0
5
10
x
5.000

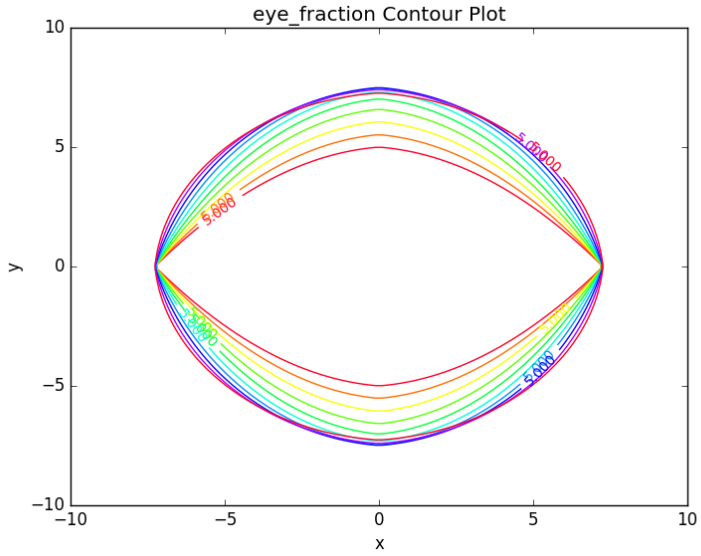
$r_1 = 0.7$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000
5.000
5.000
5.000

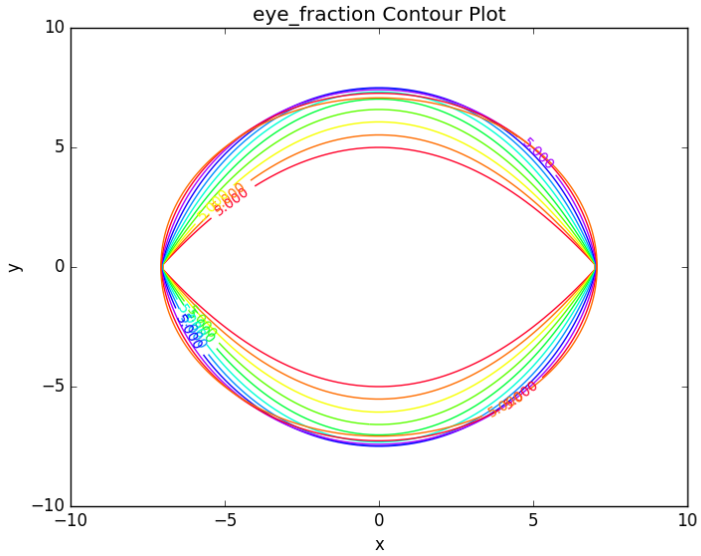
$r_1 = 0.8$


eye_fraction Contour Plot
10
5
0
−5
−10
−10
−5
0
5
10
x
y
5.000
5.000
5.000
5.000
5.000

$r_1 = 0.9$

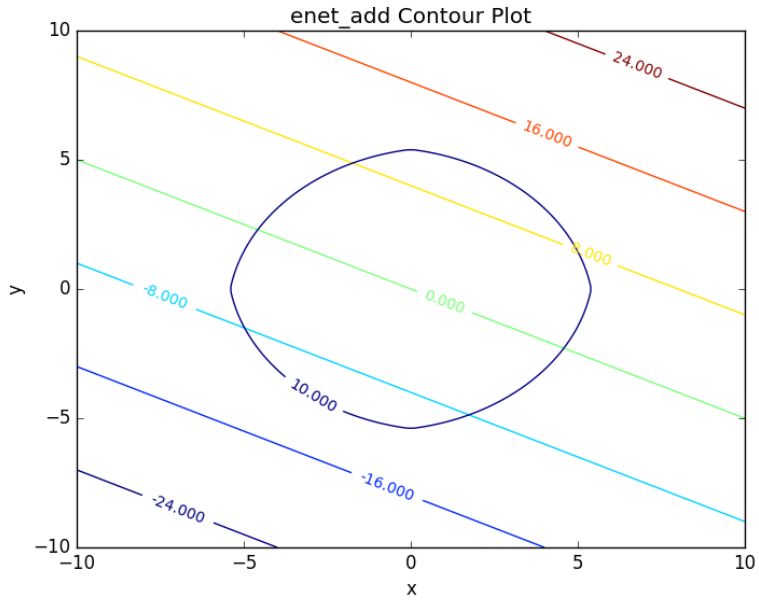


$r_1 = 1.0$



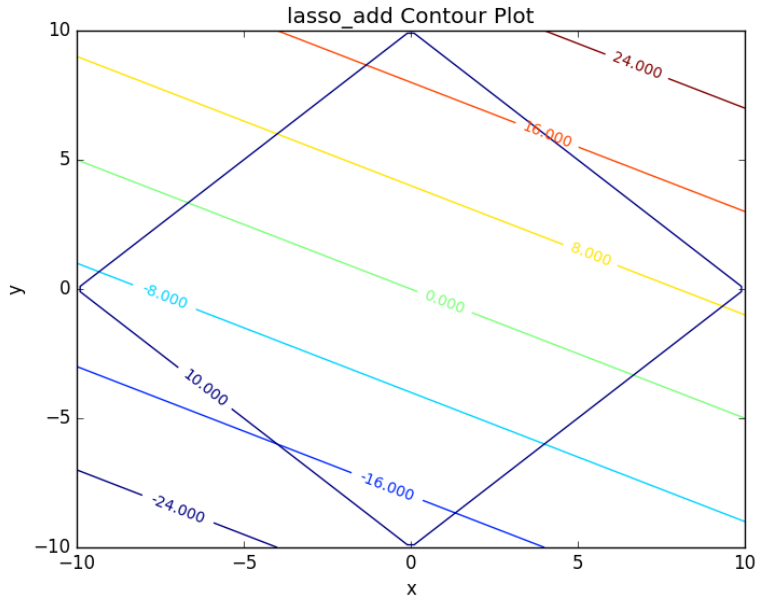
4.1.4 elastic net

α * (c * $||\theta||_1$ + 0.5 * (1 - c) * $||\theta||_2^2$) where c is a scaler



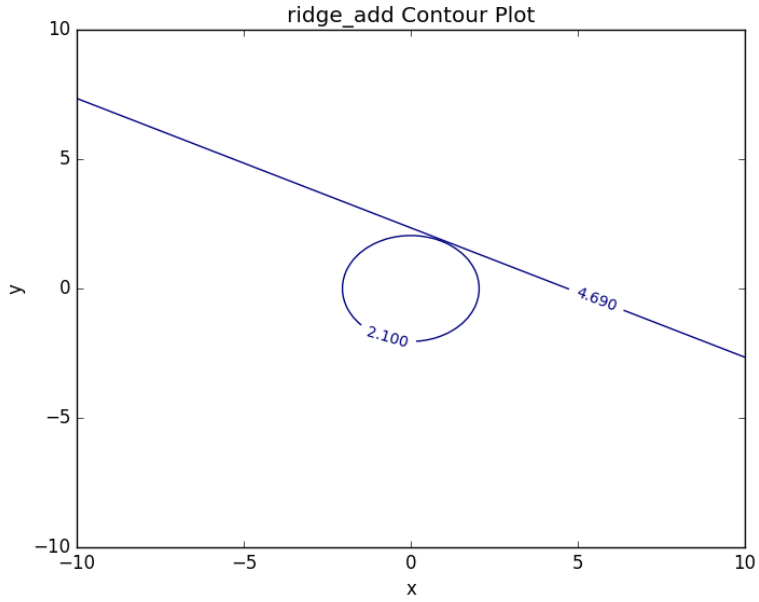
4.1.5 lasso

$\alpha * ||\theta||_1$



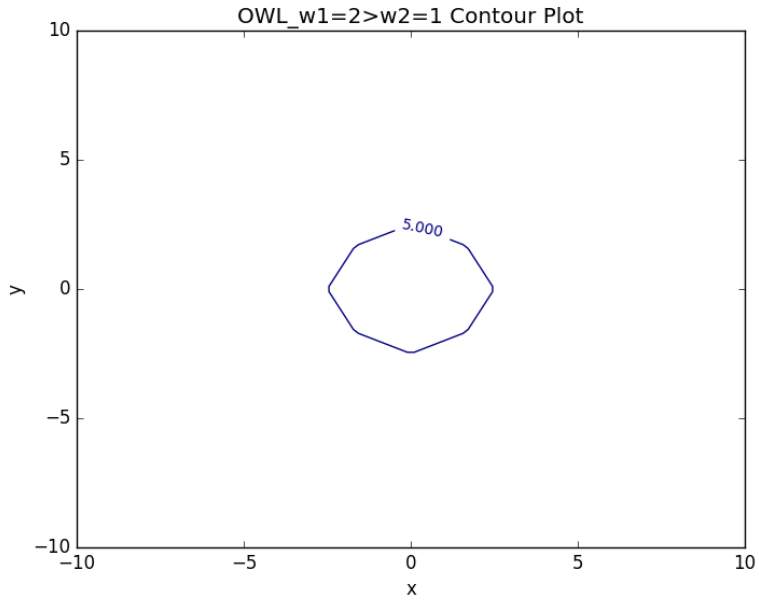
4.1.6 ridge

$0.5 * \alpha * ||\theta||_2^2$

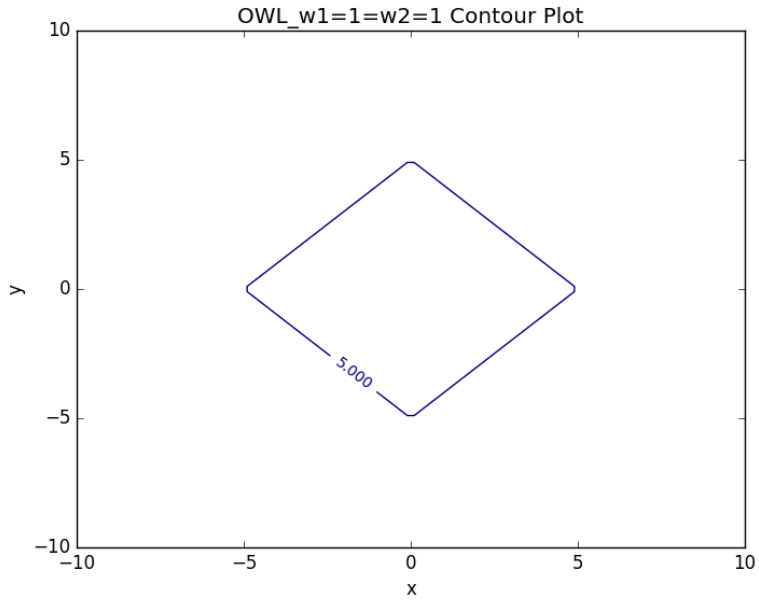


4.1.7 OWL

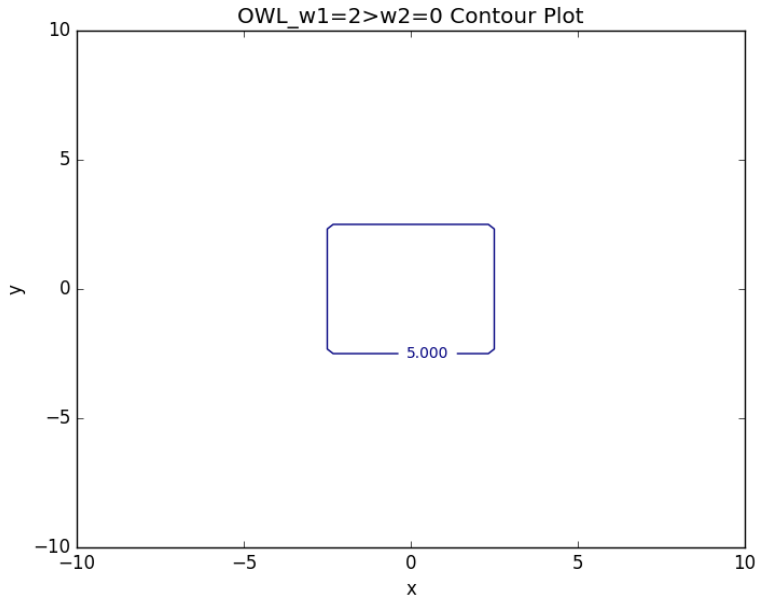
$\alpha * \sum_{i=1}^{n} w_i \, |x|_{[i]}$ where $w \in K_{m+}$ (monotone nonnegative cone)



degenerated case: back to lasso



degenerated case: back to l_{inf}



some properties:

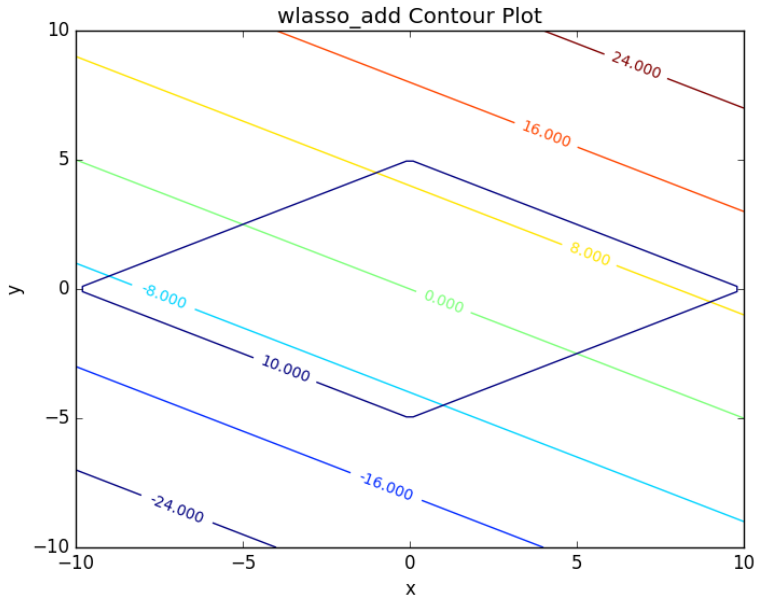
generalization of OSCAR norm

symmetry with respect to signed permutations

in the regular case, the minimal atomic set for this norm is known (the corners are easily calculated)

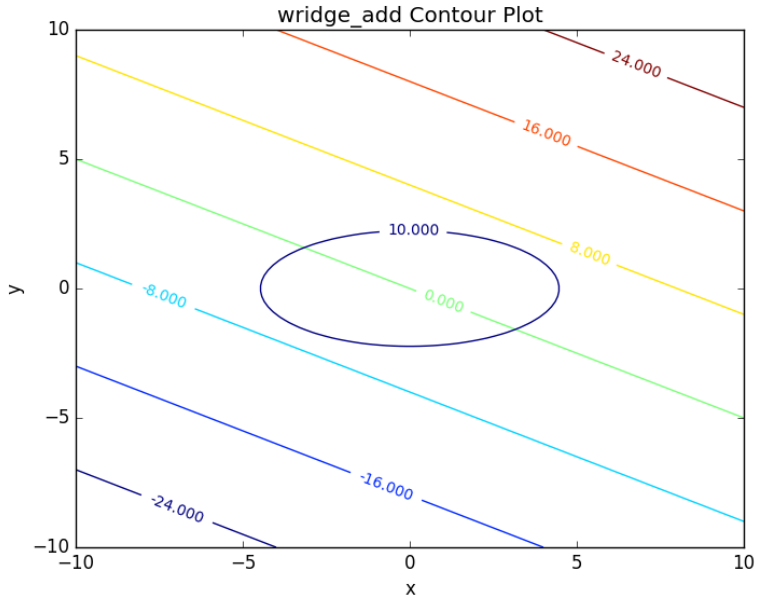
4.1.8 weighted lasso

α * ||w * θ||$_1$ where w $\in$ R_+$^{\mathrm{d}}$



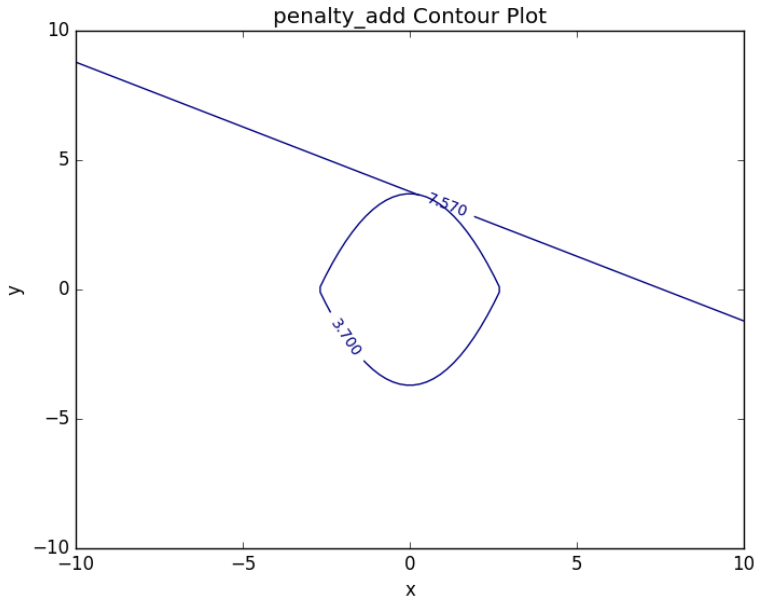
4.1.9 weighted ridge

$0.5 * \alpha * ||\mathrm{w} * \theta||_2^2$ where $\mathrm{w} \in \mathrm{R}_+^{\mathrm{d}}$



4.1.10 old penalty

α * (0.5 * (1-c) * $||$r * $\theta||_2^2$ + c * $||$(1-r) * $\theta||_1$) where r $\in \{0,1\}^d$, $\theta \in R^d$, α $\in$ R, c $\in$ R



4.2 running procedure

4.2.1 first run (regularized b)

b regularized

fix hyperparmeters to predefined value

repeat the following 100 times:

generate data (x2 = 2x1), run the selected regularizers, record θ

4.2.2 second run (unregularized b, validation)

b unregularized

generate two datasets (x2 = 2x1), one for training, one for validation

parameter search over the different hyperparams of the regularizers

for each regularizer, use the hyperparmeters that acheives the minimal loss

repeat the following 100 times:

generate data, run the selected regularizers, record θ

4.2.3 third run (data normalized, eye penalty)

b unregularized

generate two datasets (x2 = 2x1), one for training, one for validation

normalize the data to 2 mean and 2 variance (validaton data is normalized using mean and variance for the training data)

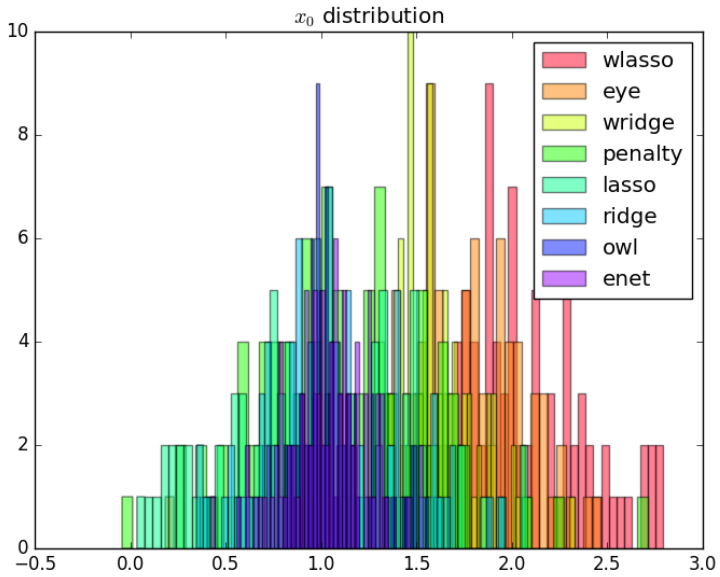
parameter search over the different hyperparams of the regularizers

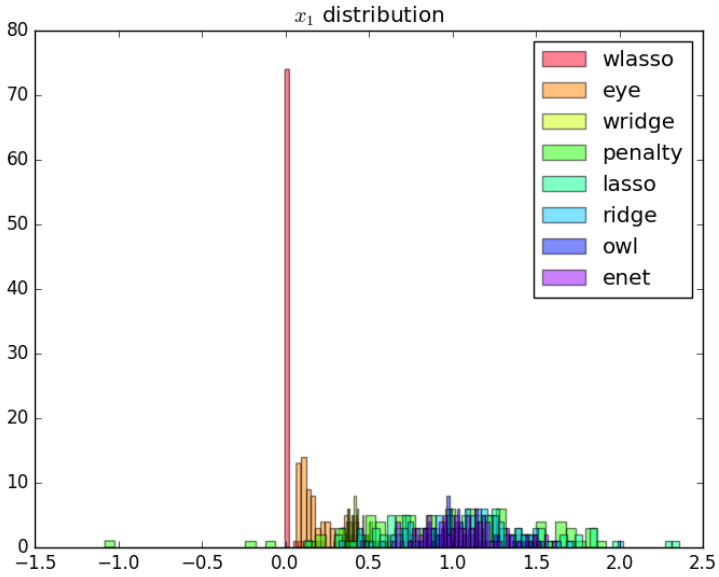
for each regularizer, use the hyperparmeters that acheives the minimal loss

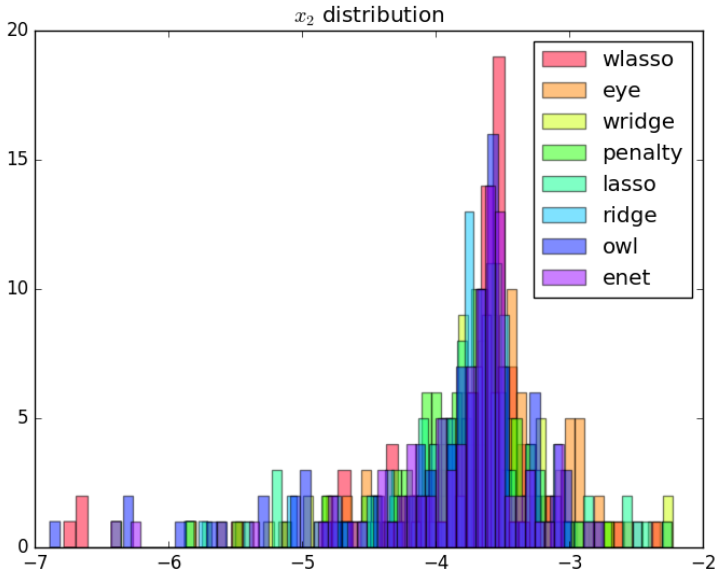
repeat the following 100 times:

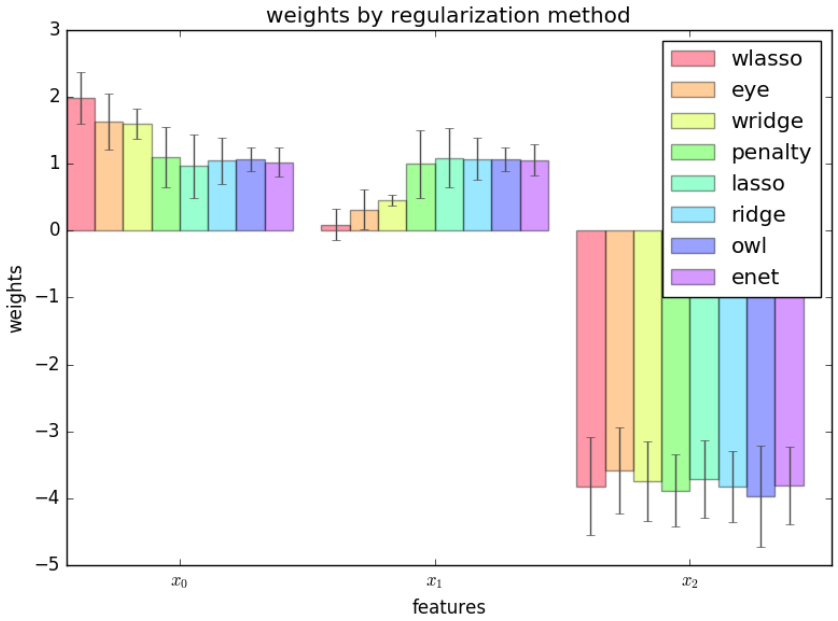
generate data, normalize data, run the selected regularizers, record θ

The choosing criteria is still loss b/c AUROC is always going to be 1 in the deterministic case:




x_1 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet
80
70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


x_2 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet
20
15
10
5
0
−7
−6
−5
−4
−3
−2



4.2.4 Fourth run (noise added)

b unregularized

generate two datasets, one for training, one for validation

normalize the data to 2 mean and 2 variance (validaton data is normalized using mean and variance for the training data)

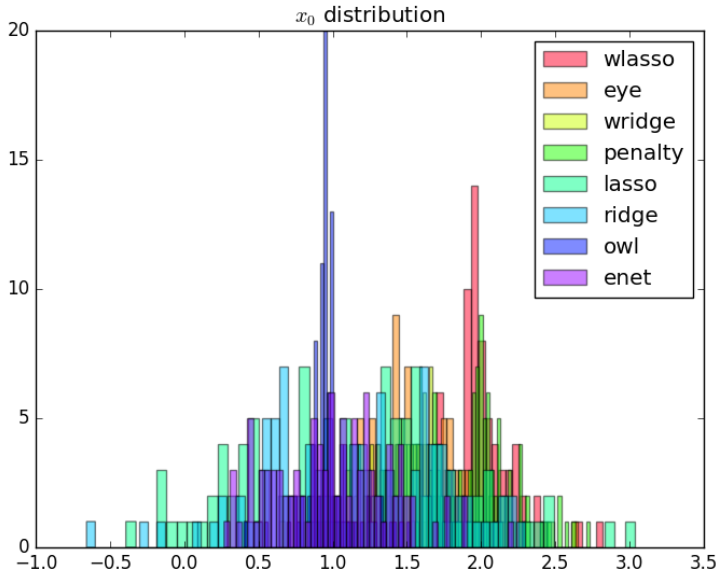
parameter search over the different hyperparams of the regularizers

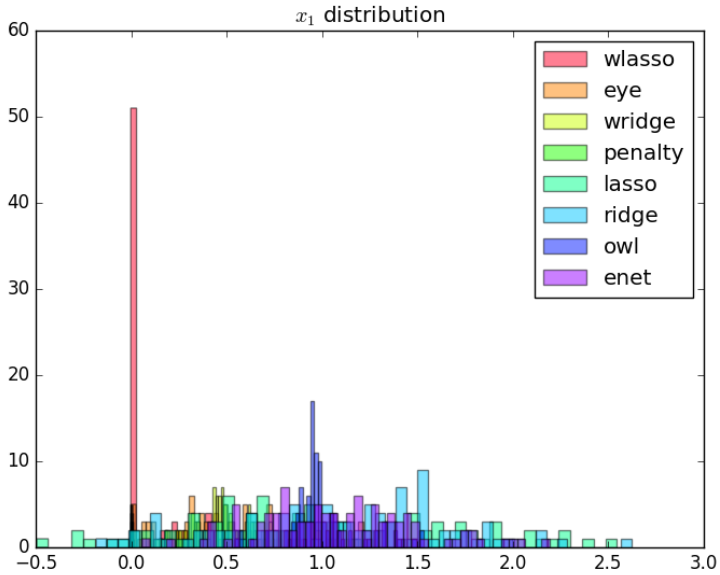
for each regularizer, use the hyperparmeters that acheives the minimal loss

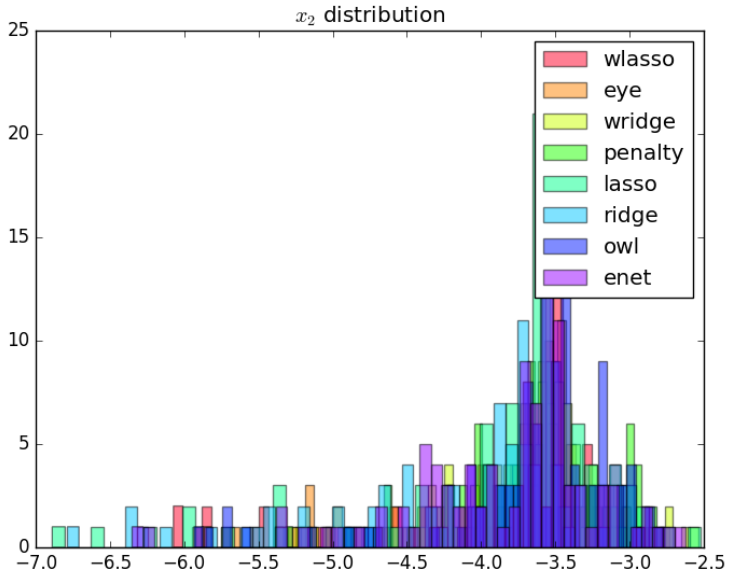
repeat the following 100 times:

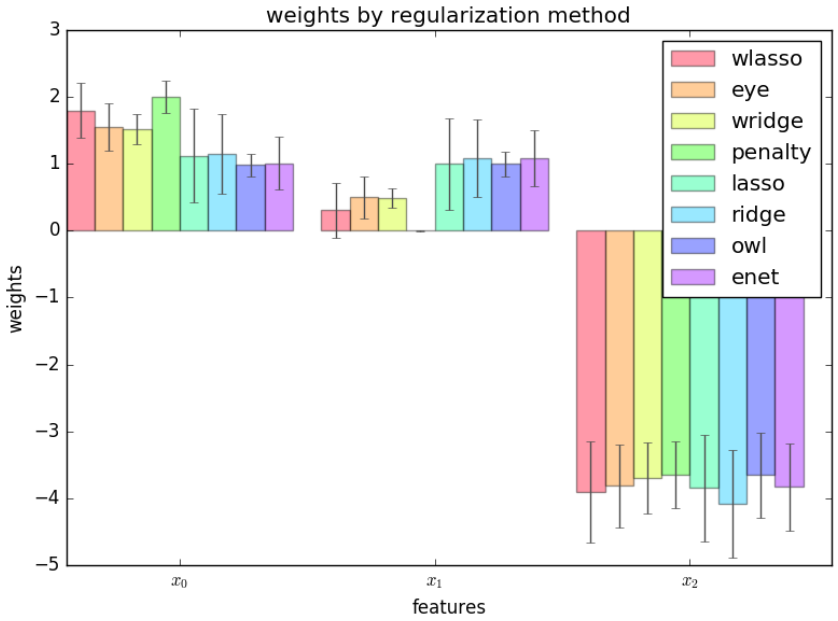
generate data (x_i = Uniform(0..4) h + N(0,0.2)), normalize data, run the selected regularizers, record θ

The choosing criteria is loss


x_0 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet
20
15
10
5
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


x_1 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet
60
50
40
30
20
10
0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0



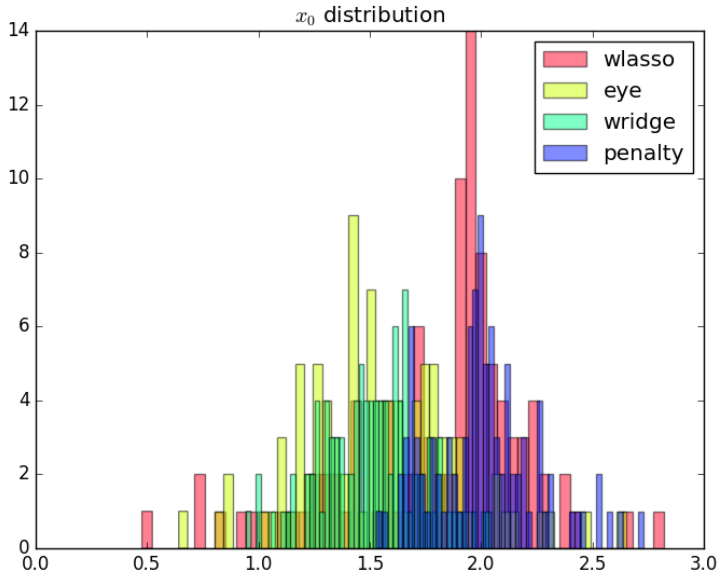


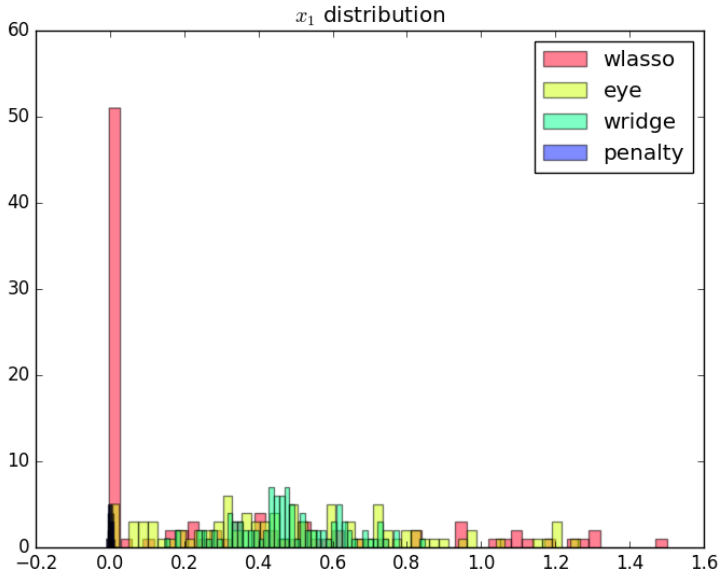
hyper parameter used:

- enet(0.01, 0.2)
- eye(array([1., 0.]), 0.01, 0.4)

- lasso(0.0001)
- OWL([2, 1], 0.01)
- penalty(array([1., 0.]), 0.1, 1.0)
- ridge(0.001)
- weightedLasso(array([1., 2.]), 0.01)
- weightedRidge(array([1., 2.]), 0.01)

The sparsity in penalty can be explained as I placed no constraint on known risk factor (l1 ratio is 1), so it only regularizes x_1 not x_0





4.2.5 fifth run (nd data)

b unregularized

generate two datasets, one for training, one for validation

normalize the data to 2 mean and 2 variance (validaton data is normalized using mean and variance for the training data)

parameter search over the different hyperparams of the regularizers (each of the final candidate has loss around 0.083)

for each regularizer, use the hyperparmeters that acheives the minimal loss

repeat the following 10-20 times:

generate data (detailed in nd data generation section), normalize data, run the selected regularizers, record θ

The choosing criteria is loss

KL divergence metric filtering for relevant features:

eye: 2.5722261048

wlasso: 5.18104309657

wridge: 6.8364694347

lasso: 18.9613782735

ridge: 12.7547711529

owl: 13.5265637342

enet: 17.7231341012
KL divergence metric including irrelevant features:
eye: 13.1307145901
wlasso: 7.55507729218
wridge: 11.5881850514
lasso: 31.1710069808
ridge: 16.9635832109
owl: 17.5479982613
enet: 30.2439873411
$\text{kl}_{\text{metricvisual}}$ (generated using $\text{gen}_{\text{result}}$.py:$\text{gen}_{\text{ndlosscsv}}$)

5 issues encountered

the validation sweep tend to pick very small regularization which essentially unconstrained (maybe increase the noise could help?)

6 next

1. develop a more general metric (related to 4)
2. look for all unknown but different weights (balanced data)
3. extend risk to fractional
4. generate data in a more diverse manner: eg. x3=h1+h2