Building Trustworty Models (Draft)

Jiaxuan Wang

<2017-02-27 Monday>

The motivation is modified based on the motivation section from Lauren Naylor's original writeup. The citations are not added in yet. They are placeholders for now.

1 motivation

Machine learning models in healthcare must achieve good predictive performance, but to be used in practice, they also must be interpretable. Interpretability can be defined in many ways depending on the context or setting. It can refer to how well a human can reproduce the calculations of a model, how intuitive the parameters and calculations are, or how well a human can understand how a model's algorithm works, even if they cannot reproduce it by hand `{need cite 8}`. In healthcare, we define an interpretable model as one that is able to provide reasons for its predictions. Past research has shown that decision trees are preferred among physicians because of their high level of interpretability `{need cite 7,10}`. However, this alone is not enough to completely gain their trust.

A model may provide reasons for its predictions, but if the reasons do not agree with what is known to be medically relevant, physicians will not trust it. For example, the lasso penalty, $\lambda \sum_{i=1}^{n} |\theta_i|$ is commonly used to create interpretable models. This penalty induces sparsity in the learned feature weights, so that predictions can be explained by a small number of relevant factors. While this improves interpretability, it does nothing to ensure that the selected features align with physicians' knowledge. If a feature that is known to be relevant is correlated with a feature that is not, the model may use the latter feature to make predictions and discard the former.

This example suggests that in addition to interpretability, we also want the model to be credible, that is to agree with prior knowledge in the field under the constraint that the performance is not worsend. Note that the constraint is important because we still need our model to be consistent with data so that the model is useful.

Maintaining performance while increasing credibility may sound too good to be true: isn't everything come with a cost? Given we already have an accurate model, what credibility does is to filter the reality through a particular point of view and the cost we pay is just providing this viewing lens, which in the medical context is the known risk factors. To illustrate this point, consider an absurd example of trying to predict the number of people drown in a month using season and number of icecreams sold within that month as features. Anyone reasonable would agree that more people drown in summer than in any other season because more people swim in summer. The season should be an obvious relevant feature for a learning algorithm. However, it is very possible that a machine learner would choose number of icecreams sold as a predictive variable over season because more icecream sold implies summer and thus positively affect the number of people drown (the two features are correlated). The learned model is obviously as accurate as the one only using season, but the fact that the model is not aware of the thinking mode of human makes its reasoning cryptic, which can be easily fixed by providing the model with proper knowledge. Thus, this seemingly free lunch property is obtained by leveraging domain knowledge, which in most cases is easy to obtain.

It should be noted that credibility implies interpretability, but the inverse is not true.

The goal of this research is to create credible models in the sense that matches a clinician's medical knowledge. We aim to develop methods for combining the expert-based relevancy of features with a datadriven model. As a case study, we focus on the specific prediction task of predicting a patient's risk of acquiring an infection with C. difficile.

2 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.

3 related work

Credibility and interpretability is usually acheived through feature selection, which can be further breakdown to subset selection, shrinkage, and dimension reduction `{to be cited}`. Subset selection selects subset of features so that when the model is trained on this subset, the tradeoff between model simplicity and increase in loss is balanced. This class of methods include best subset selection and its computationally efficient variant stepwise selection. Shrinkage methods refer to regularization, which is penalty added on the size of model parameters. This is the most often used methods for feature selection due to its non-intrusive nature concerning training pipeline. Dimension reduction methods try to learn the true dimension of the data so that noise is minimized and correlation is removed. Example methods include PCA and ICA. In this work, we focus on regularization methods.

The most commonly used and analyzed regularizations are L_1 (lasso) and L_2 (ridge) norm due to their desirable statistical properties. Each of which can be interpreted as placing a prior distribution on feature weights `{need cite see the adaptive lasso paper}`. The sparseness in feature weights induced by lasso's diamond shaped contour makes it more favorable in the context of eliminating irrelevant features, thus many extensions over it are proposed, including ordered weighted loss (OWL) `{to cite OWL paper}`, adaptive lasso `{to be cited}`, elastic net `{to cite}`, and weighted lasso `{need to cite 19}`. While OWL, elastic net, and weighted lasso are generalizations of lasso, adaptive lasso satisfies the oracle property in the sense that under mild regularity conditions, it identifies the right subset model and is consistent with true parameter (that is converge in distribution to the true underlying feature weights). However, adaptive lasso requires learning another model to set its weight, which makes it more cubersome to use than others.

The most natural extension over the regular lasso penalty is the weighted lasso, which introduces a weight w_i for each feature: $\lambda \sum i = 1^n w_i |\theta_i|$. This penalty is used in `{need to cite 9}` where the feature's weight is the inverse of its relevance. This approach causes the weights of less relevant features correlated with more relevant features to be driven to zero. However, we may not know the relevance of the features that have not been identied as risk factors: there may be undiscovered relationships not mentioned in the literature. If such a feature were correlated with a known risk factor, we would want to throw it out and use the known risk factor, but if it is not correlated with another feature and is predictive, we would like to keep it. Combining expert knowledge with a model is explored in `{need cite 14}`. The model is trained using features identied as relevant, along with the subset of other features from the data that give the most improvement to performance, while creating the least redundancy in the features. This work differs from ours because their list of relevant features is assumed to be known, and their motivation is to increase model performance, not credibility.

4 measuring success

Fixing the level of performance, the task of learning is to allocate weights to features so that desirable structures are kept. We want our model to be consistent with physician's knowledge. 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. We call a model credible if it satisfies the following properties:

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, weights of known risk factors 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 have 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 distance in distribution between each group of correlated features and the known risk factor indicator vector within that group. The metrics used for the correlated feature group space are KL divergence and earth mover's distance. Earth mover's distance measures the amount of work to turn one distribution to the other and is symmetric, while KL divergence is asymmetric in its argument.

Here I give an exmaple 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.

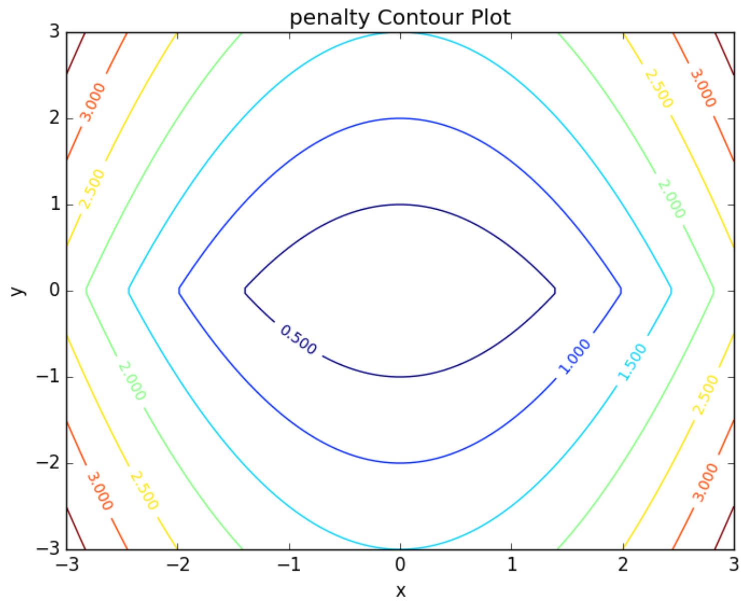
5 method

The most natural approach to encourage sparseness in unknown risk factors while maintaining dense weights in known risk factors is to constrain known risk factors using l_2 norm and unknown risk factors using l_1 norm. Formally, this penalty term can be written as

pena(θ) = α (0.5 * (1-β) $||r \odot \theta||_2^2 + \beta\ ||(1\text{-}r) \odot \theta||_1$)

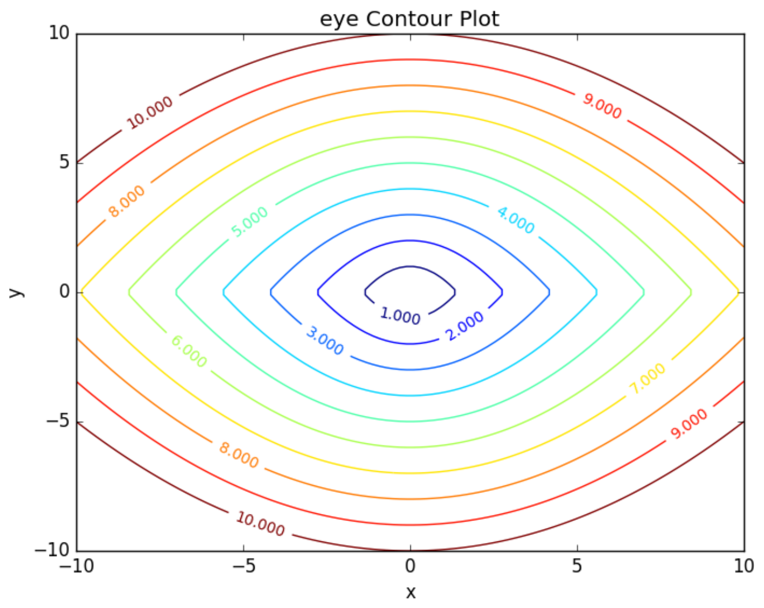
where r $\in \{0,1\}^d$, $\theta \in \mathbb{R}^d$, $\alpha \in \mathbb{R}_+$, $\beta \in [0,1]$

Assuming x is the known risk factor and y is the unknown risk factor, we plot the contour of this penalty:



As the contour plot suggests, this penalty function is nonhomogeneous: that is $f(tx) \neq |t|f(x)$. In the case of perfectly correlated variables, this translate to sensitivity to α: small α will let the model favor unknown risk factor y which is opposite to what we want.

To address this issue, we propose eye penalty which is obtained by fixing a convex body in the contour of pena and scale it for different contour levels. We call the fixed contour as the generating convex body. The generating convex body are chosen by constraining the slope at the right end point in the cross section between known and unknown risk factors to be -1, which forces perfectly correlated features to favor known risk factors.



The new contour plot demonstrate that this new penalty term is homogeneous.

A derivation of this penalty and the proof of its properties can be found in the last section. Here I simply state the result:

5.1 formal definition of eye penalty

q(x) $= 2\ \beta\ ||(1\text{-r}) \odot \text{x}||_1 + (1\text{-}\beta)\ ||\text{r} \odot \text{x}||_2^2$

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

5.2 properties

1. eye is a norm
2. β controls only the scaling factor of the norm This implies that β need not to be grid searched because α also controls scaling factor
3. eye is a generalization of lasso, ridge, and elastic net

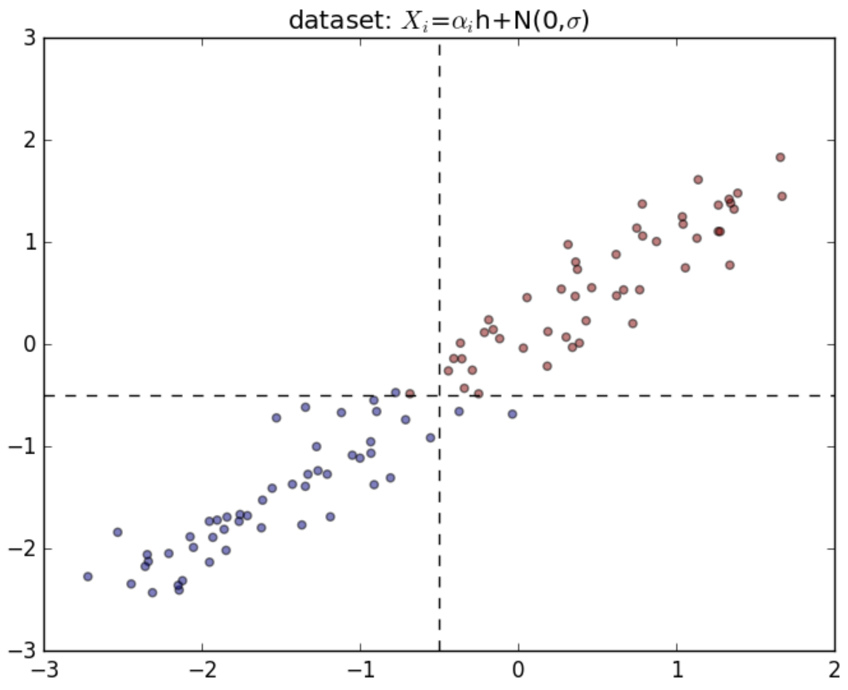
6 experiments

Each experiment is ran with different aim in mind. The first four experiments explore 2d data while the last four experiments explore high dimensional data. The last experiments applies eye penalty to C. difficile prediction.

6.1 generating data

6.1.1 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

6.1.2 nd highly correlated 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

6.1.3 general nd data generation

Data n = 2000

n relevant groups (nrgroups) = 11

n irrelevant group (nirgroups) = 0

correlated variables pergroup (npergroup) = 4

Given a covariance matrix C

Do cholesky decomposition: C = A A^T

h ~ N(0,1,shape=(n,d))

x = h A^T

$\theta_i = 1 \ \forall$ i

y = $\mathbb{1}_{X\theta > 0}$

r (known risk factors): for each dependent group, set half as known, half as unknown

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

6.2 running procedure

6.2.1 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

6.2.2 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

6.2.3 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)

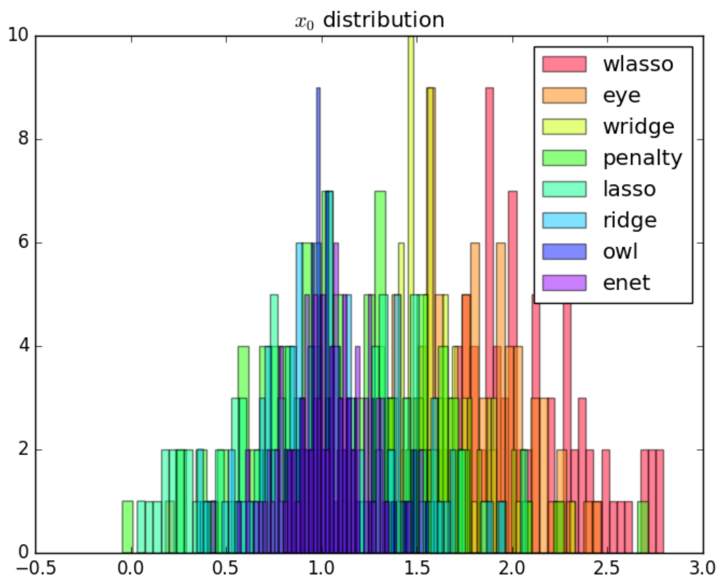
parameter search over the different hyperparams of the regularizers

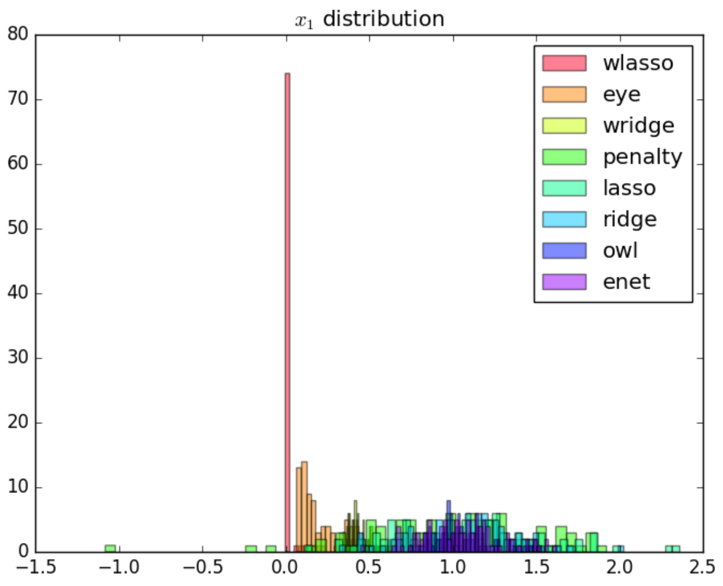
for each regularizer, use the hyperparmeters that acheives the minimal loss

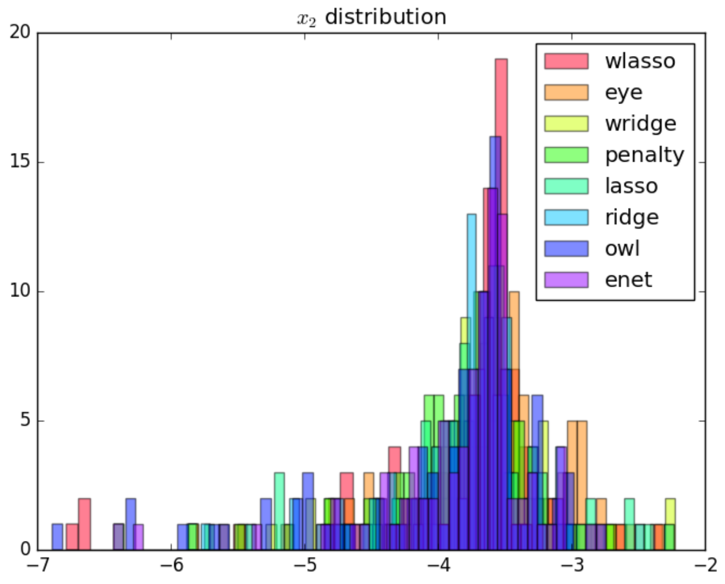
repeat the following 100 times:

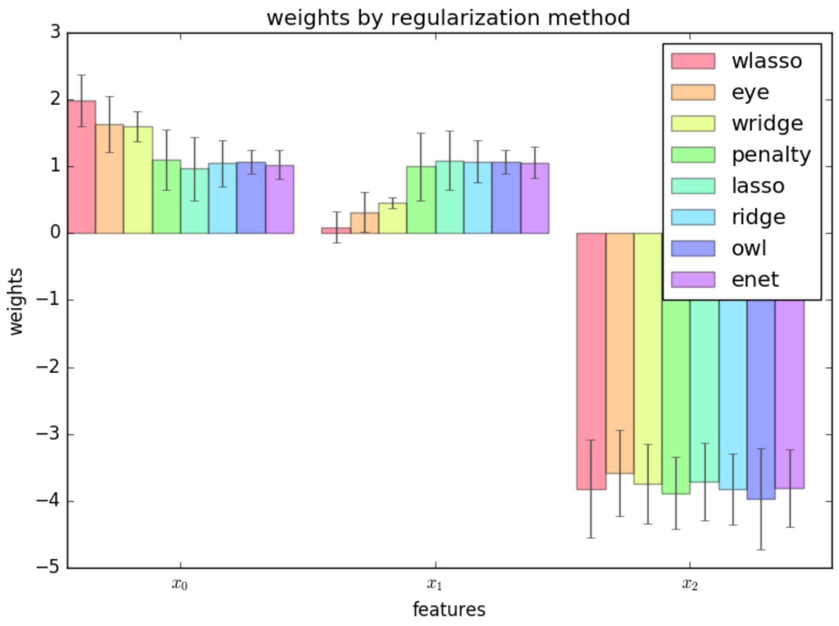
generate data, normalize data, run the selected regularizers, record θ

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


x_0 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet
0
2
4
6
8
10
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0


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





6.2.4 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)

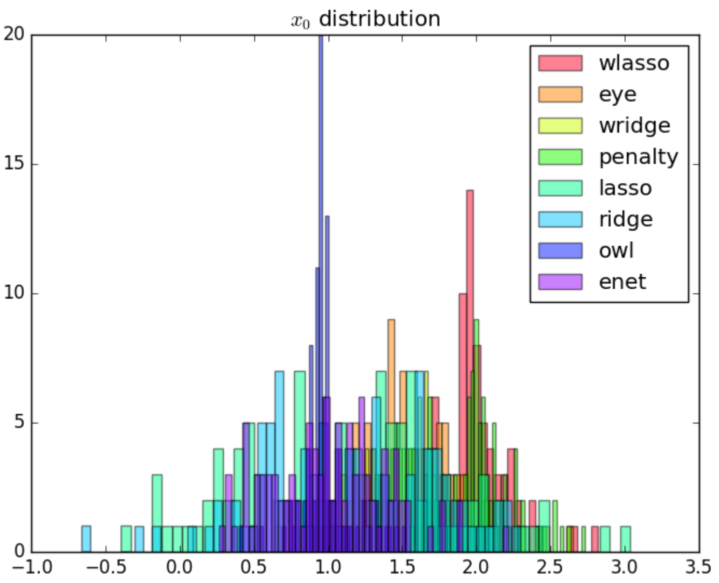
parameter search over the different hyperparams of the regularizers

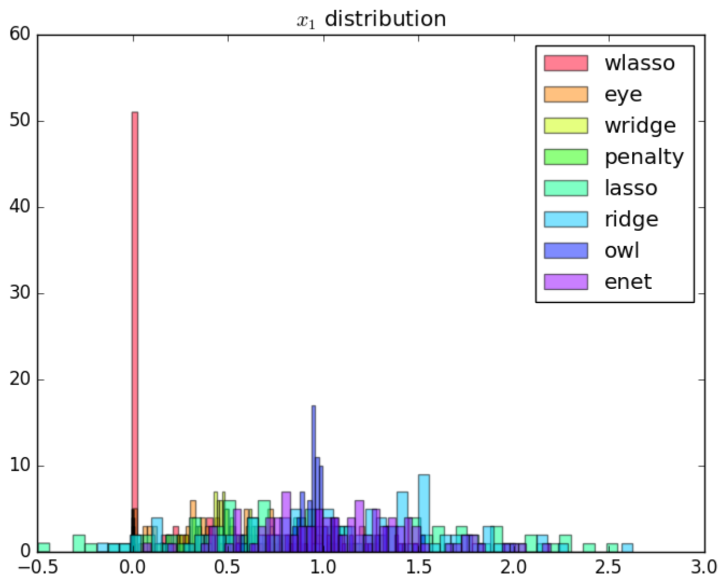
for each regularizer, use the hyperparmeters that acheives the minimal loss

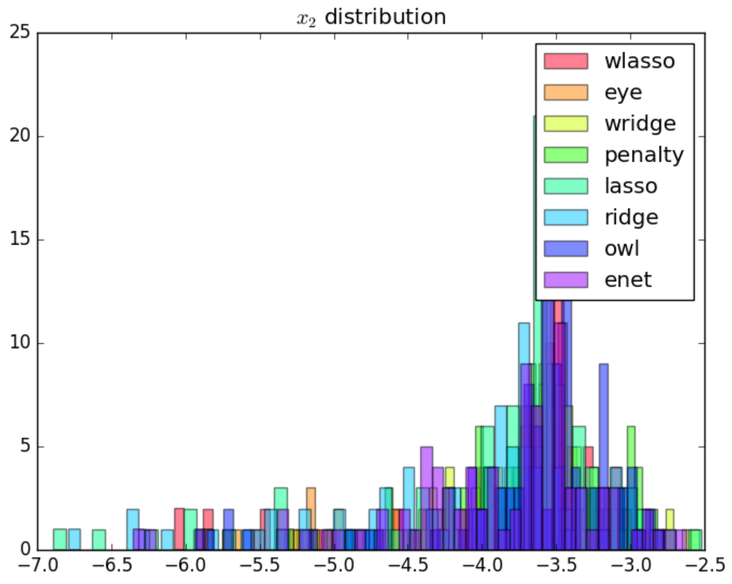
repeat the following 100 times:

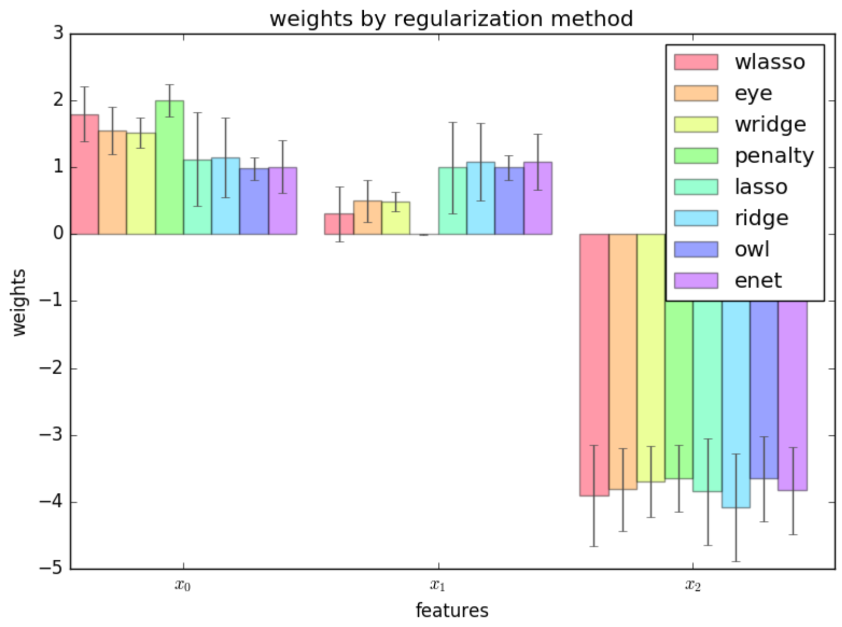
generate data ($\mathrm{x_i}$ = Uniform(0..4) h + N(0,0.2)), normalize data, run the selected regularizers, record θ

The choosing criteria is loss




x_1 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet

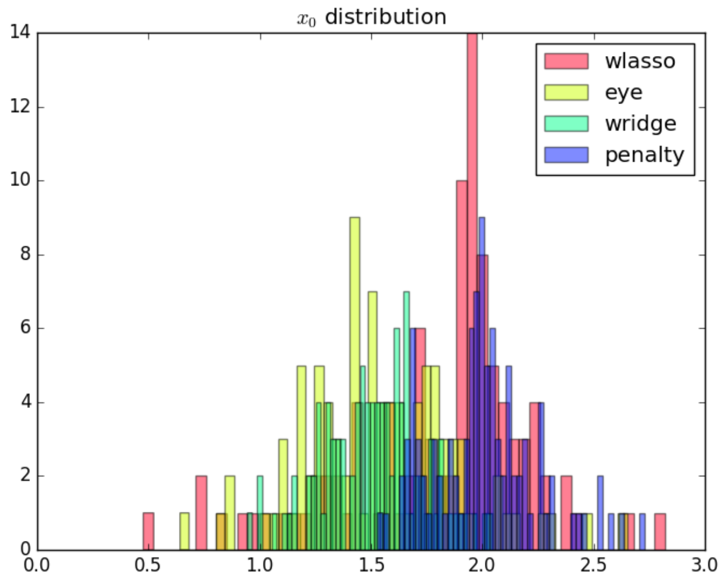

x_2 distribution
wlasso
eye
wridge
penalty
lasso
ridge
owl
enet

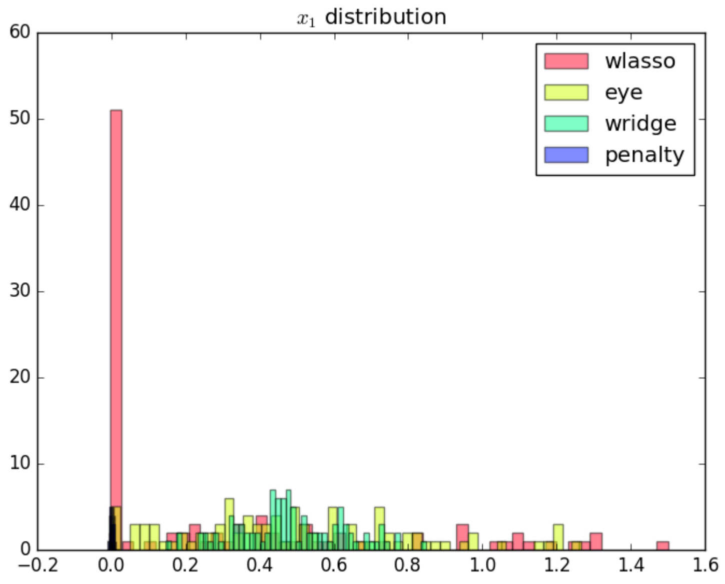


hyper parameter used:

- 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


x_0 distribution
wlasso
eye
wridge
penalty
0
2
4
6
8
10
12
14
0.0
0.5
1.0
1.5
2.0
2.5
3.0


x_1 distribution
wlasso
eye
wridge
penalty
0
10
20
30
40
50
60
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

6.2.5 fifth run (nd data, sweep r, fix correlation of 0.04, fix theta to 1)

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

kl/emd$_{\text{metricvisual}}$ (generated using gen$_{\text{result}}$.py:gen$_{\text{ndlosscsv}}$, is in .pages format so assumes mac, included in attachment)

6.2.6 TODO sixth run (sweep corelation, fix r, fix theta to 1)

construct a covariance matrix with 10 different blocks on diagnal with variables in each block having a different covariance value. This experiment is to discover the relationship between noise level and credibility.

6.2.7 TODO seventh run (sweep fractional r, fix correlation, fix theta)

To extend r to be fractional, we consider setting r according to parametrized functions: log, exp, sigmoid, and linear.

6.2.8 TODO eighth run (sweep theta, fix r, fix correlation)

Try different theta in data generation. I expect this will not make a difference in dependent groups compared to run 5, 6, and 7.

6.2.9 TODO real data

After graduating from simulated data, we will apply eye penalty to C. difficile prediction.

7 summary of regularizations used in this work

7.0.1 eye penalty

q(θ) := 2 β $||$(1-r) $\odot$ $\theta||_1$ + (1-β) $||$r $\odot$ $\theta||_2^2$

pena(θ) := α q(θ)

where r $\in$ $\{0,1\}^{\mathrm{d}}$, $\theta \in \mathbb{R}^d$, $\alpha \in \mathbb{R}_+$, β $\in$ (0,1) (β is also called l1$_{\mathrm{ratio}}$ in this text)

For any constant c

pena(θ) = 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, adding in scaling factor α, equivalently we write

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

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.

$$\begin{align}
&\alpha[2\beta y + (1-\beta)x^2] = c \tag{1}\\
&\to 2\beta y + (1-\beta)x^2 = \frac{c}{\alpha} \tag{2}\\
&\to y = \frac{c}{2\alpha\beta} - \frac{(1-\beta)x^2}{2\beta} \tag{3}\\
&\to y = 0 \Rightarrow x = \sqrt{\frac{c}{\alpha(1-\beta)}} \tag{4}\\
&\to f'(x) = -\frac{(1-\beta)}{\beta}x \tag{5}\\
&\to f'(\sqrt{\frac{c}{\alpha(1-\beta)}}) = -\frac{1-\beta}{\beta}\sqrt{\frac{c}{\alpha(1-\beta)}} = -1 \tag{6}\\
&\to c = \frac{\alpha\beta^2}{1-\beta} \tag{7}\\
&\to 2\beta y + (1-\beta)x^2 = \frac{\beta^2}{1-\beta} \tag{8}
\end{align}$$

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$_{\beta_1}$}(x) = t} and B$_2$ = {x: eye$_{\beta_2}$}(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$_{\beta_1}$(x) = $\frac{\beta_1^2}{1-\beta_1}$ }}

B$_2$ = t {x: x $\in$ {x | q$_{\beta_2}$(x) = $\frac{\beta_2^2}{1-\beta_2}$ }}

let B$_{1'}$ = {x: x $\in$ {x | q$_{\beta_1}$(x) = $\frac{\beta_1^2}{1-\beta_1}$ }} and B$_{2'}$ = {x: x $\in$ t {x | q$_{\beta_2}$(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$_{\beta_1}$(x) = 2 β_1 ||(1-r) * x||$_1$ + (1-β_1) ||r*x||$_2^2$ = $\frac{\beta_1^2}{1-\beta_1}$

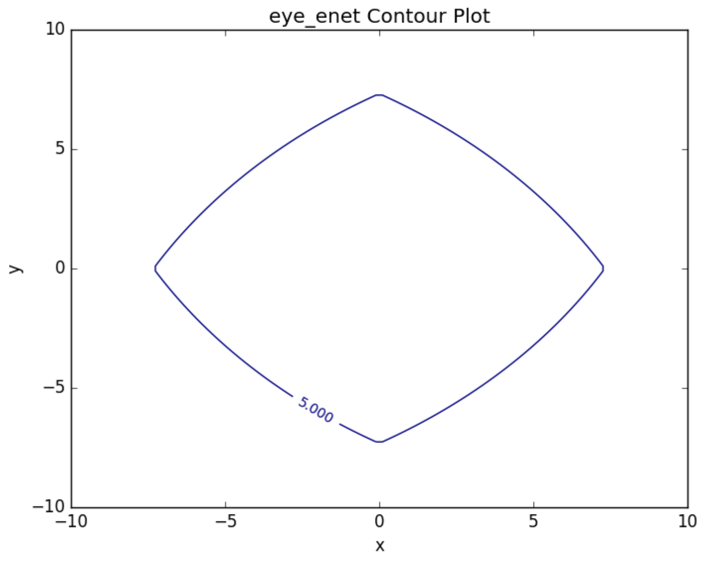
let x' = $\frac{\beta_2(1-\beta_1)}{\beta_1(1-\beta_2)}$ x

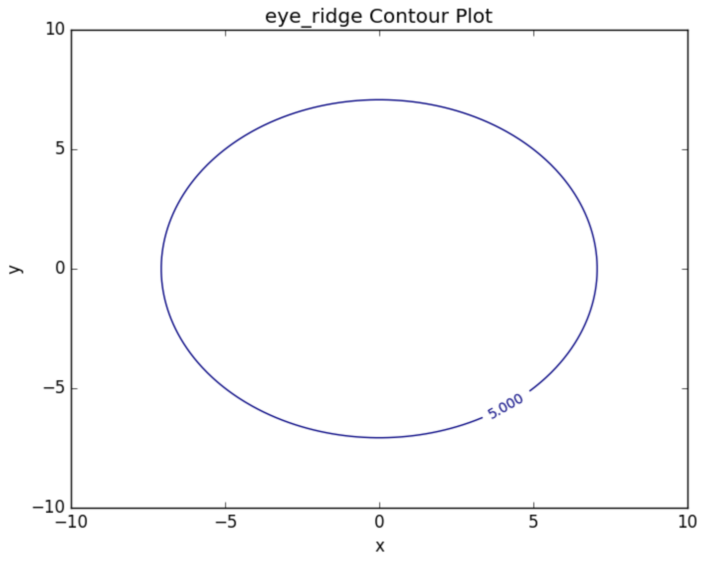
$$\begin{aligned}
q_{\beta_2}(x') &= 2\beta_2||(1-r)*x'||_1 + (1-\beta_2)||r*x'||_2^2 && (9)\\
&= \frac{2\beta_2^2(1-\beta_1)}{\beta_1(1-\beta_2)}||(1-r)*x||_1 + \frac{\beta_2^2(1-\beta_1)^2}{\beta_1^2(1-\beta_2)}||r*x||_2^2 && (10)\\
&= \frac{\beta_2^2(1-\beta_1)}{\beta_1^2(1-\beta_2)}(2\beta_1||(1-r)*x||_1 + (1-\beta_1)||r*x||_2^2) && (11)\\
&= \frac{\beta_2^2(1-\beta_1)}{\beta_1^2(1-\beta_2)}\frac{\beta_1^2}{1-\beta_1} && (12)\\
&= \frac{\beta_2^2}{1-\beta_2} && (13)
\end{aligned}$$

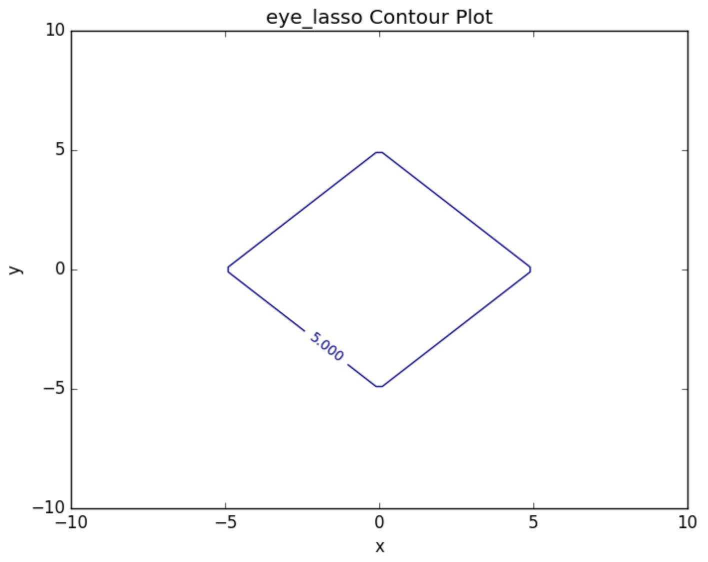
so x' $\in$ B$_{2'}$. Thus $\frac{\beta_2(1-\beta_1)}{\beta_1(1-beta_2)}$ B$_{1'}$ $\subset$ B$_{2'}$. The other direction is similarly proven.

c) eye as a generalization of elastic net, lasso, and ridge

By relaxing the constraint of r from binary to float, we can recover elastic net (setting r=0.5 * **1**). Even without extending r, we can recover ridge (r= **1**) and lasso (r= **0**)


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

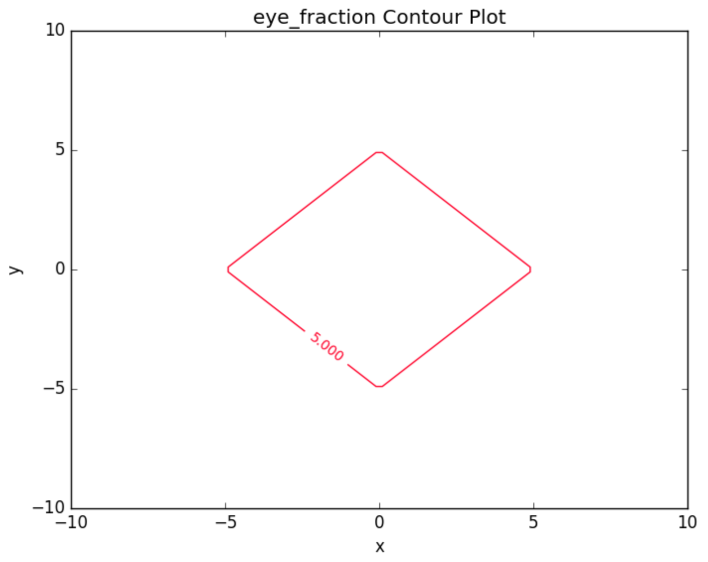

eye_ridge Contour Plot
10
5
0
−5
−10
y
−10
−5
0
5
10
x
5.000



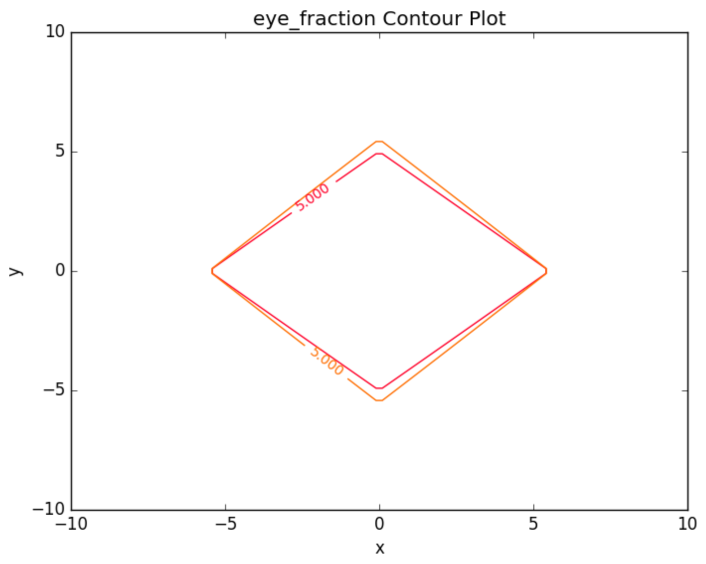
3. extending r to $[0,1]^d$ At times, it makes sense for risk factor to be fractionally weighted (eg. odds ratio in medical documents).

   varying r_1 and r_2 (in the following plot, r_2 are sweep from 0 up to r_1 with stepsize of 0.1)

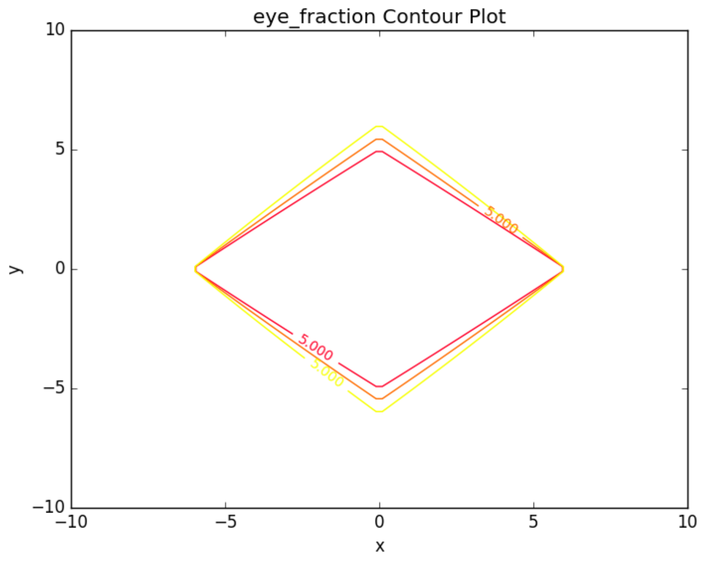
   $r_1 = 0.0$



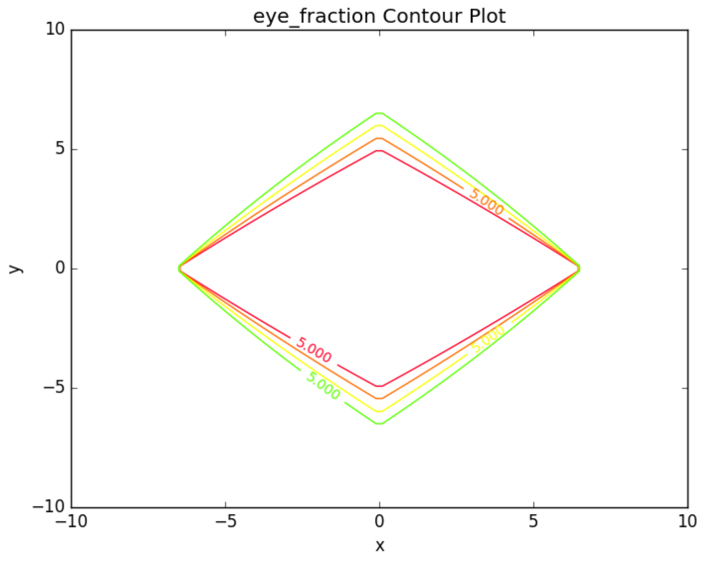
$r_1 = 0.1$



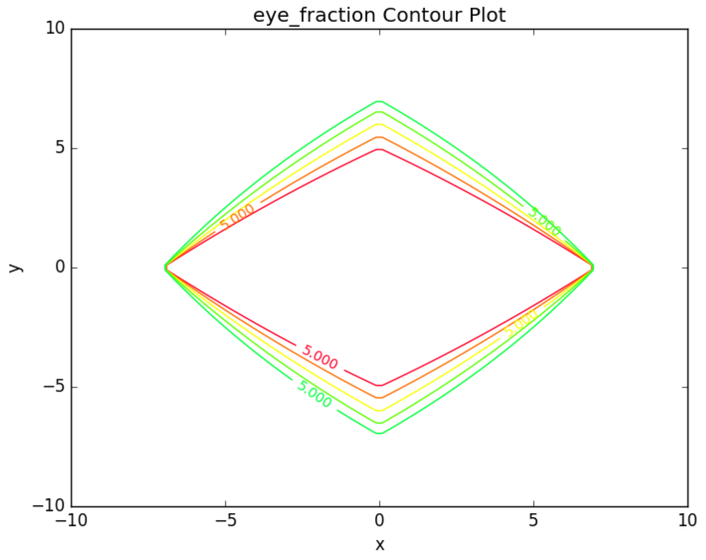
$r_1 = 0.2$



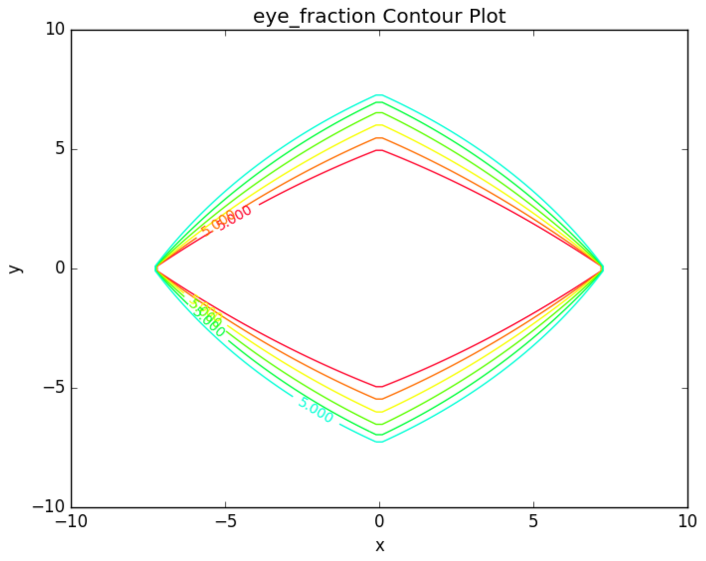
$r_1 = 0.3$


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

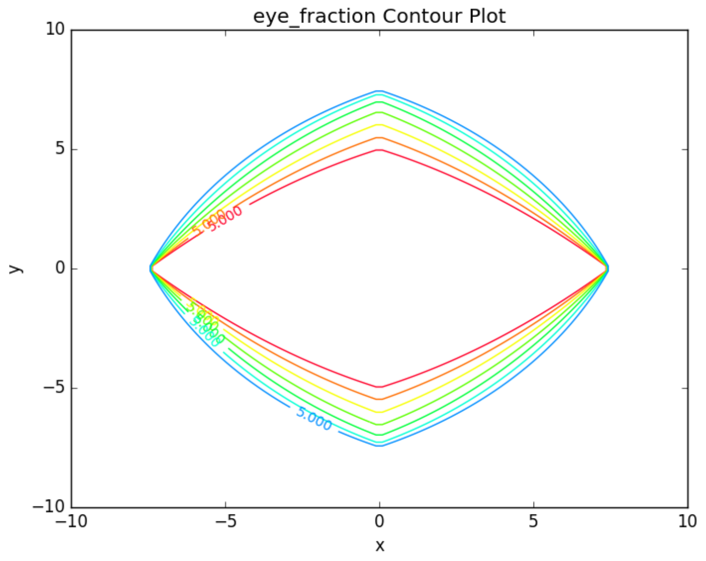
$r_1 = 0.4$


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

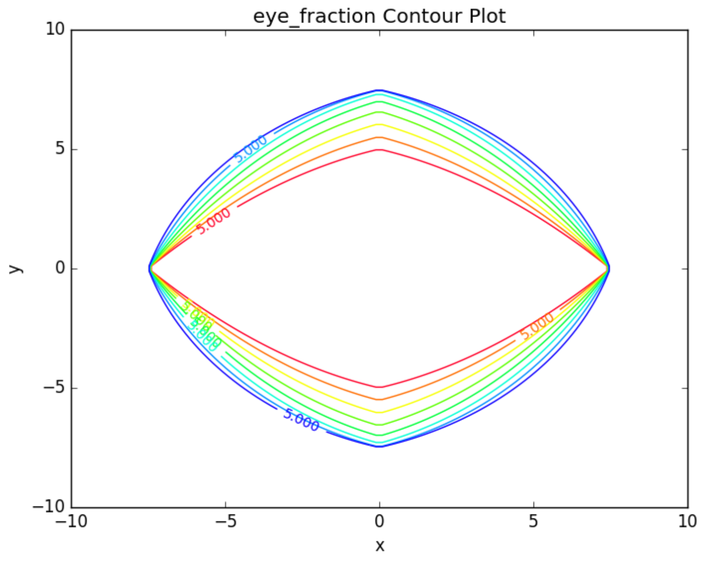
$r_1 = 0.5$


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

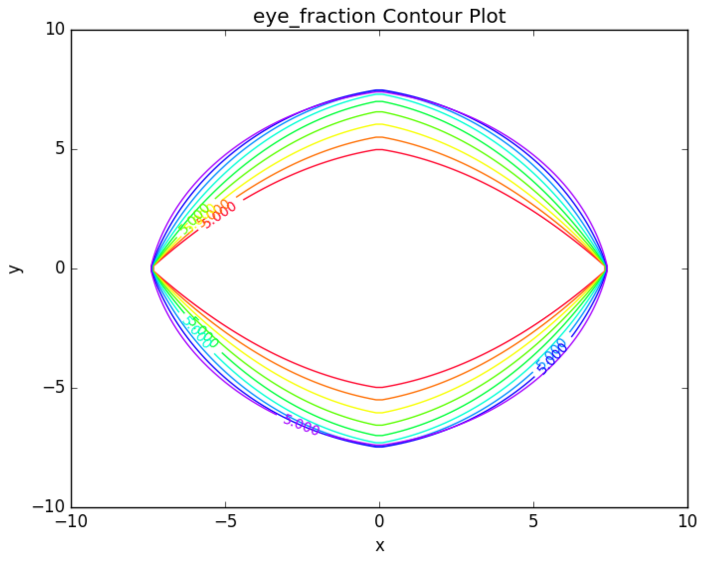
$r_1 = 0.6$


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

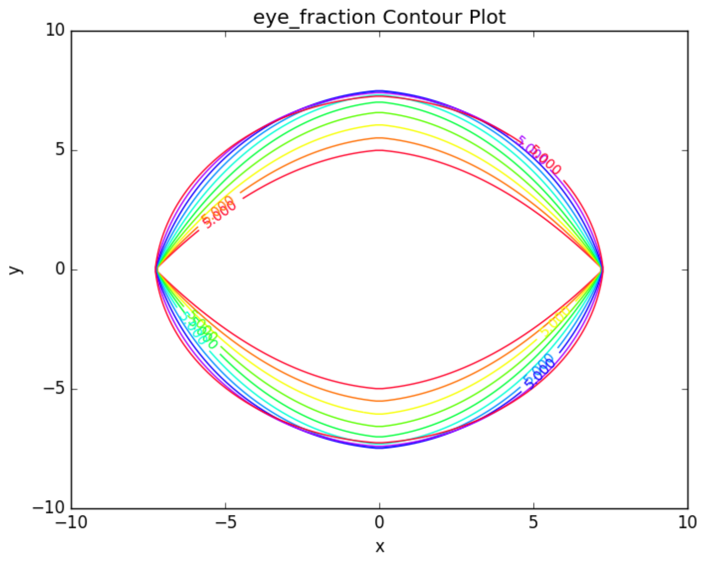
$r_1 = 0.7$


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
5.000

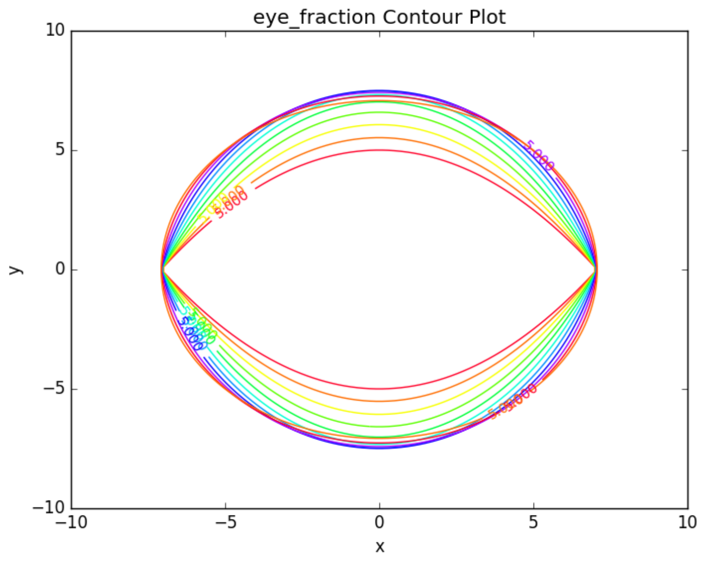
$r_1 = 0.8$


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
5.000

$r_1 = 0.9$

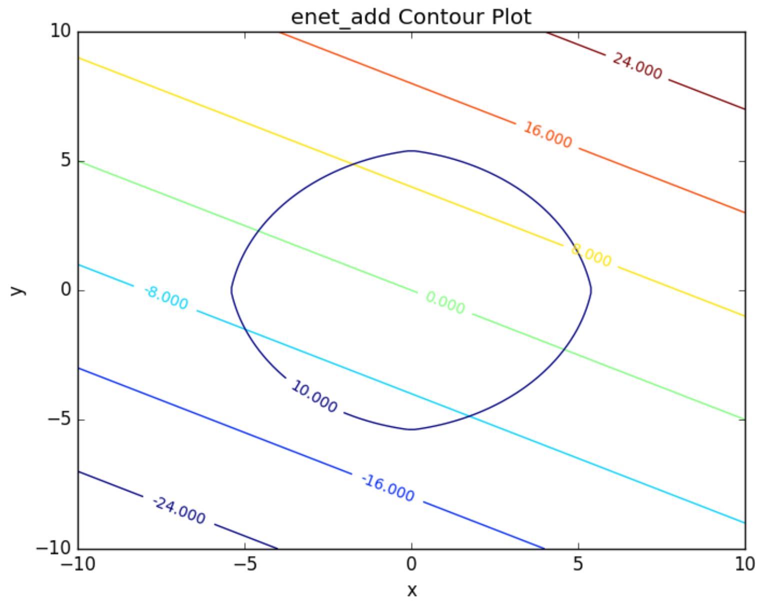


$r_1 = 1.0$



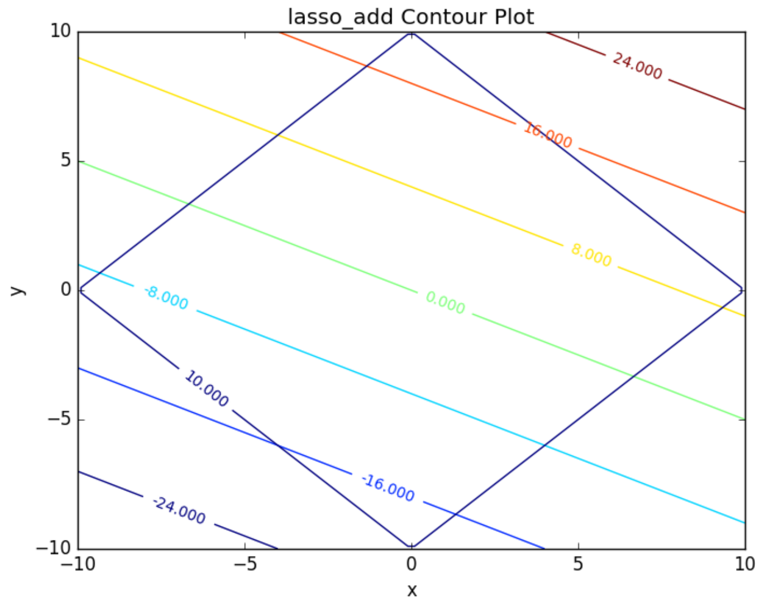
7.0.2 elastic net

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



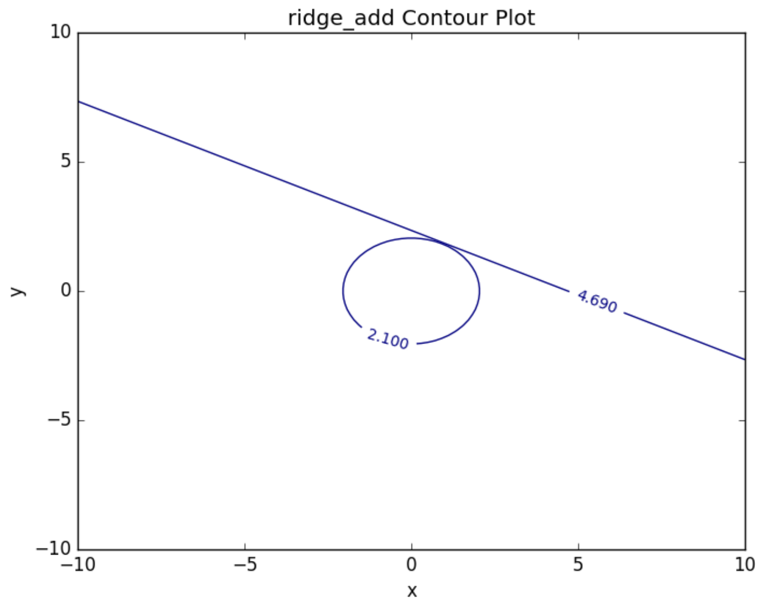
7.0.3 lasso

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



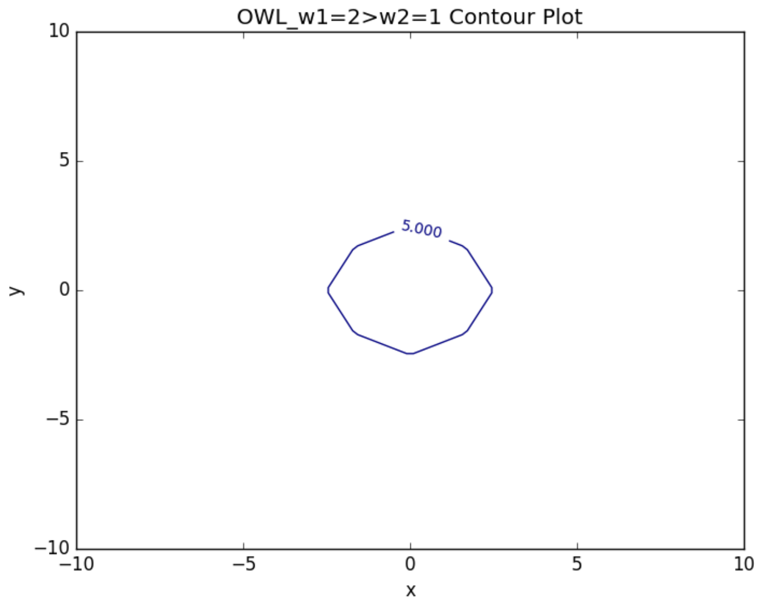
7.0.4 ridge

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

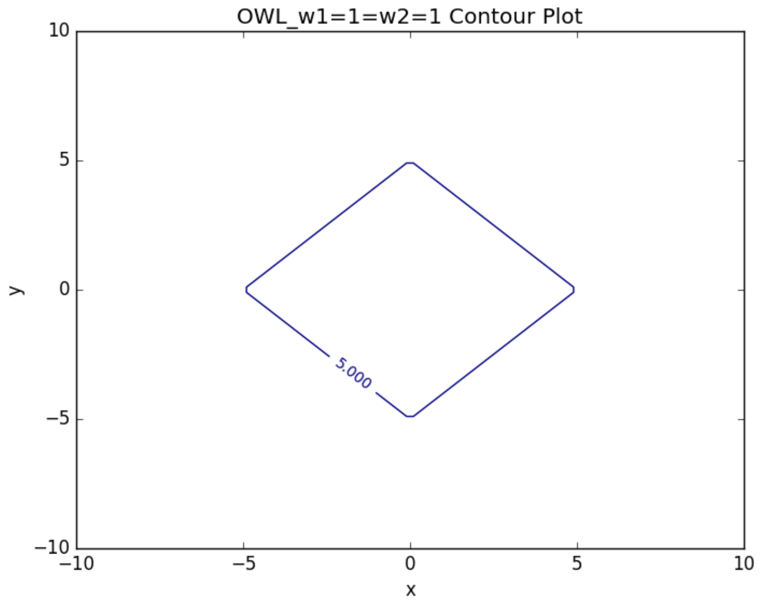


7.0.5 OWL

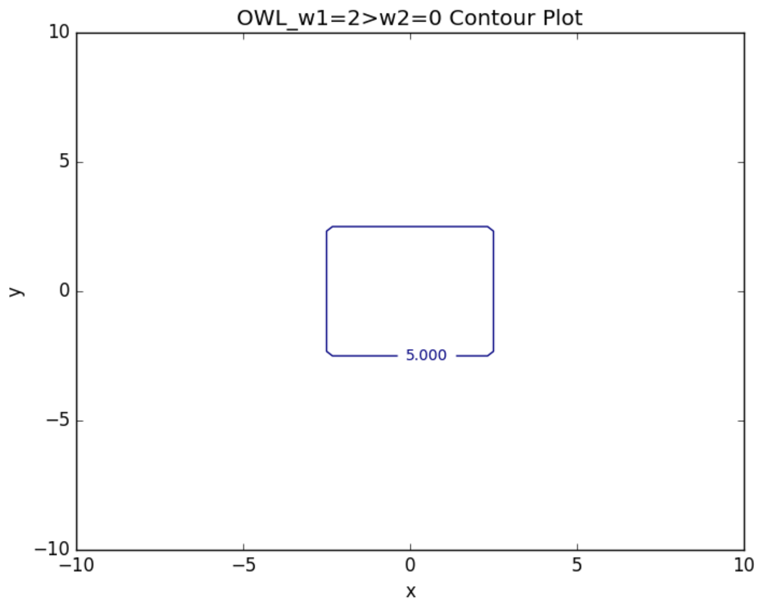
$\alpha * \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w_i}\ |\mathrm{x}|_{[\mathrm{i}]}$ where $\mathrm{w} \in \mathrm{K}_{\mathrm{m}+}$ (monotone nonnegative cone)



degenerated case: back to lasso



degenerated case: back to l_{inf}



some properties:

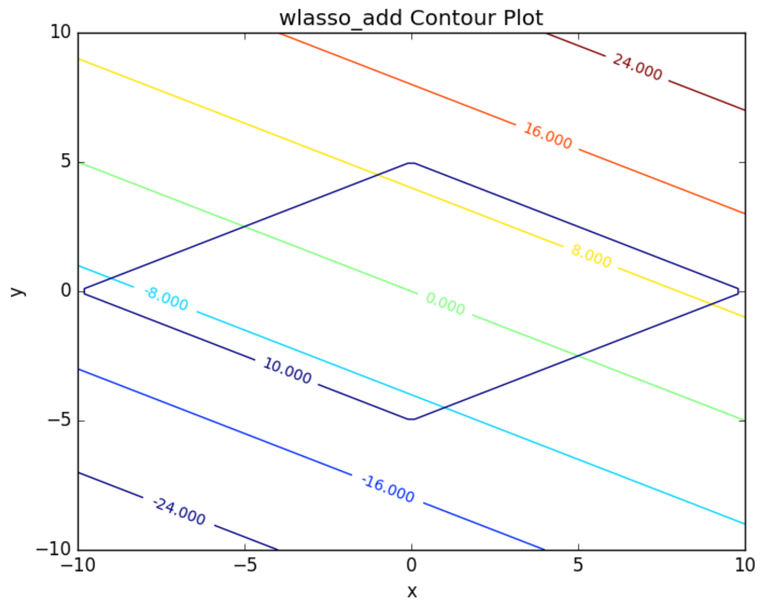
generalization of OSCAR norm

symmetry with respect to signed permutations

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

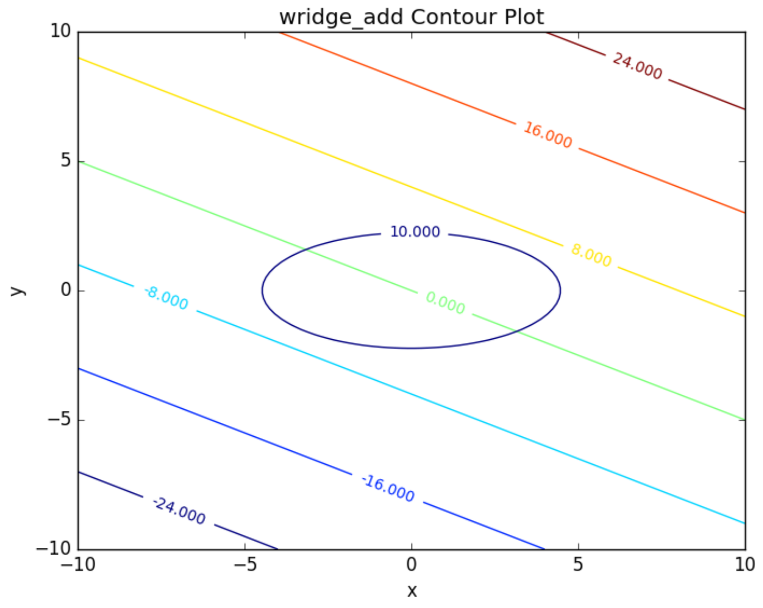
7.0.6 weighted lasso

α * $||$w * $\theta||_1$ where w $\in$ \$R\$_$+^{\mathrm{d}}$



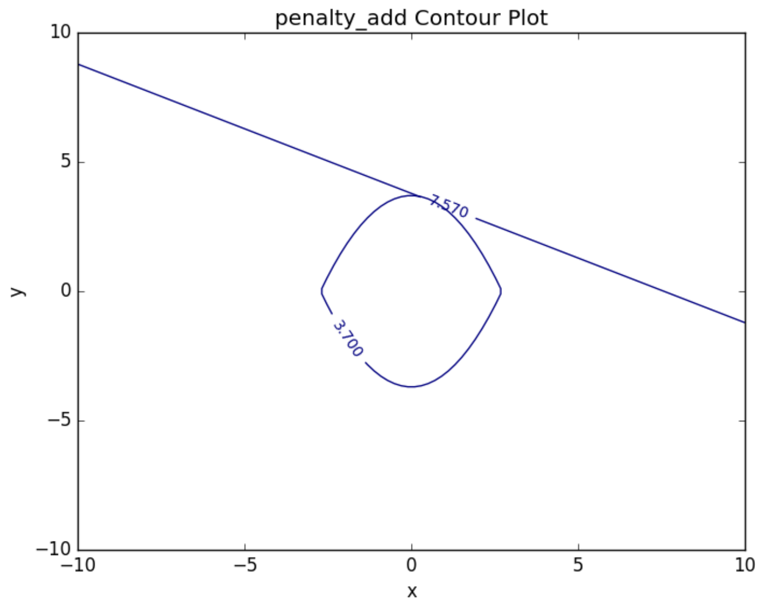
7.0.7 weighted ridge

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



7.0.8 old penalty

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


penalty_add Contour Plot
x
y