H2O Training

1. Introduction to ML
2. Introduction to H2O
3. H2O Data Munging
4. Machine Learning Concepts
5. H2O Algorithms
6. Productionalizing Models

Supervised Learning in H2O

- **Regression and Classification Models** Exponential distributions including Poisson, Gamma, and Tweedie are available in addition to Bernoulli, Multinomial, and Gaussian distributions.
- **Fast and CPU Efficient** Parallel and distributed computation across multiple nodes and many cores.
- **Grid Search** Hyperparameter optimization allows the user to run through many parameters before selecting the best models.
- **Early Stopping** The user can specify the metric and the incremental change in the metric as convergence.
- **Stochastic** User can specify the sample rate that the algorithm will sample the column and row by for better generalization.
- **Model Output** The model is exportable as Java code and if you find the model overfitted after a certain number of trees, it is easy to reduce the number of trees in a POJO before putting it in production without rerunning model build.

Supervised Learning:

GENERALIZED LINEAR MODEL

What is the Generalized Linear Model (GLM)?

- Class of models that relate X (inputs) to Y (output)

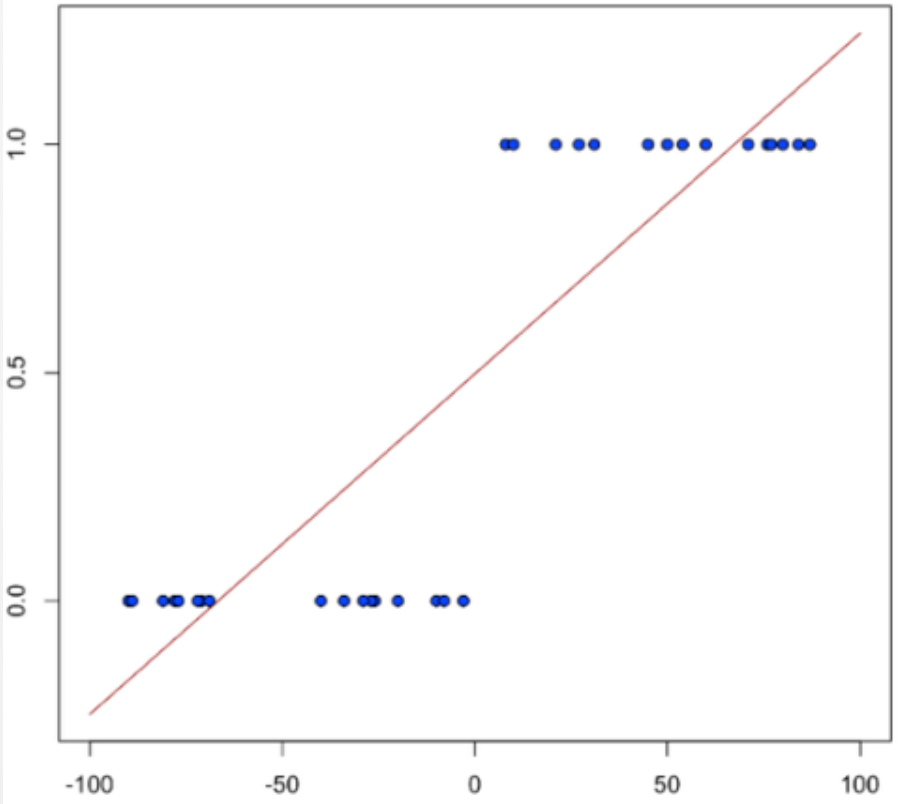
$$\mathrm{E}(\mathbf{Y}) = \boldsymbol{\mu} = g^{-1}(\mathbf{X}\boldsymbol{\beta})$$

$$\mathrm{Var}(\mathbf{Y}) = \mathrm{V}(\boldsymbol{\mu}) = \mathrm{V}(g^{-1}(\mathbf{X}\boldsymbol{\beta})).$$

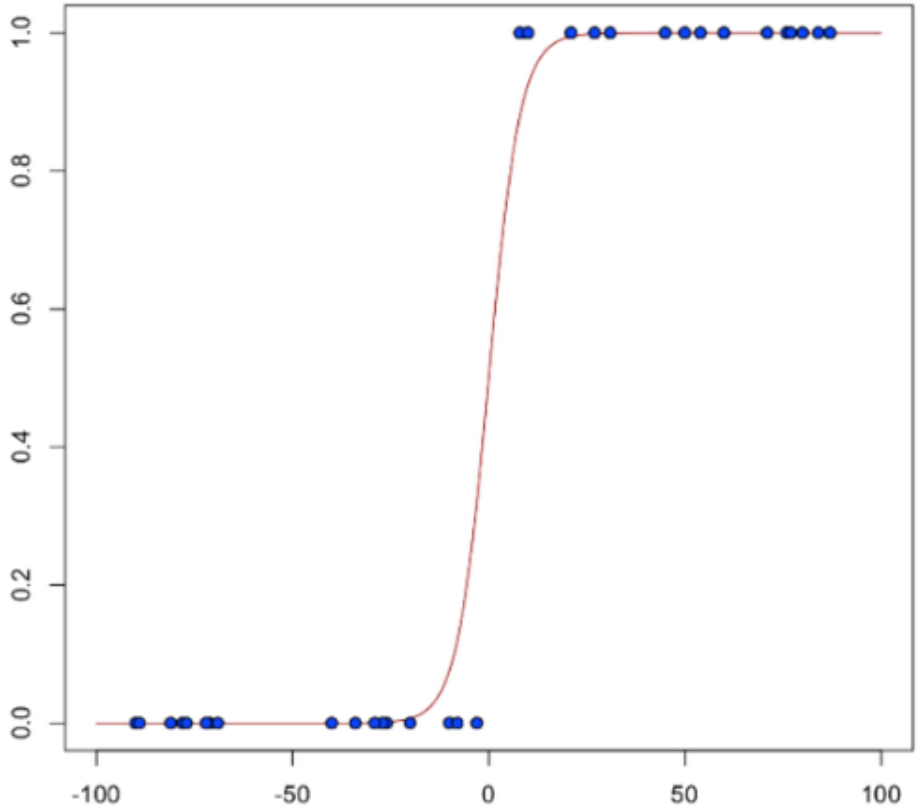
- Allows for a unification of models that have errors of the following form:
  - Normal (Gaussian)
  - Poisson
  - Gamma
  - Tweedie
  - Binomial (Logistic)
  - Multinomial
- MLE is found by iteratively reweighted least squares

Same predictors, different family and link functions

Linear Regression fit
(family=gaussian,link =identity)



Logistic Regression fit
(family=binomial,link = logit)



Pros and Cons of GLM's

Pros

- Fast to fit
- Easy to interpret
- Well understood statistical properties
- Works for continuous, categorical, and count targets
- Transparent scoring functions

Cons

- Dependent of external feature engineering
- Difficult to model non-linearities
- Difficult to model interaction effects
- Ill-defined with multicollinearity
  - Non-unique solutions

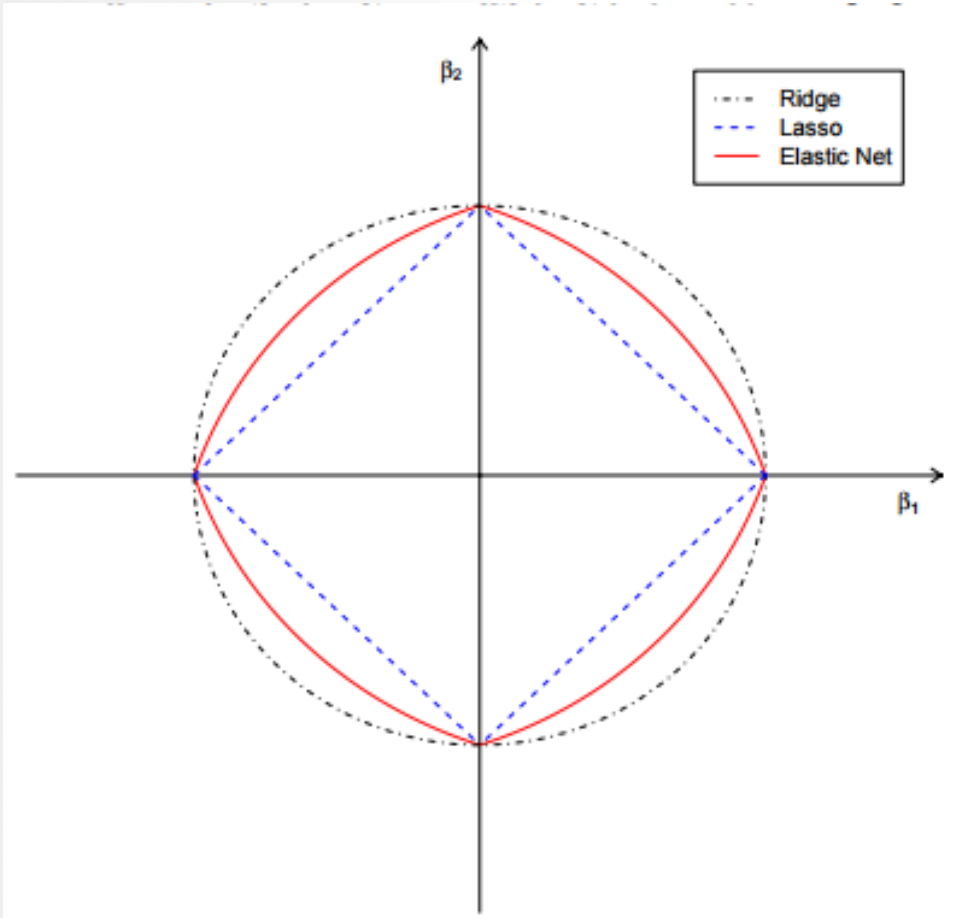
Supervised Learning:

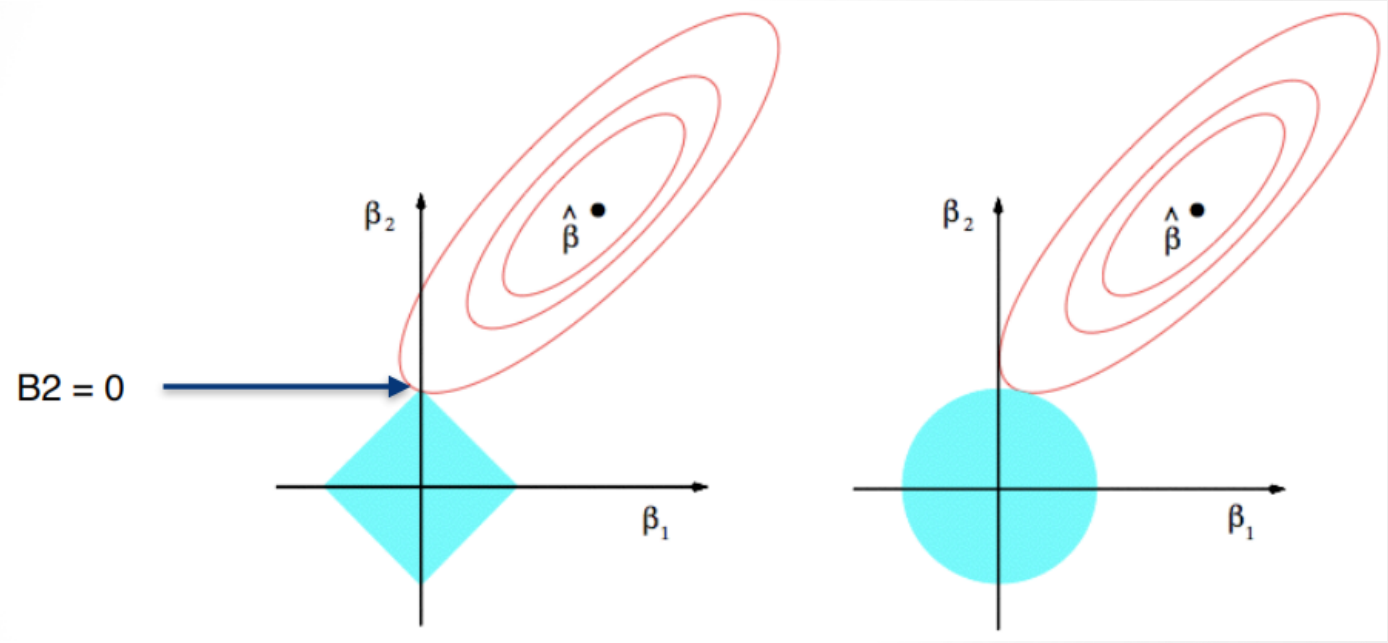
PENALIZED GENERALIZED LINEAR MODEL(LASSO, RIDGE, ELASTIC NET)

Regularized Regression

- Want to build a parsimonious but interpretable model
- Shrink the number of predictors and/or size by imposing penalties on the estimated coefficients (L1, L2)
  - LASSO: $\min_{\beta_0,\beta} \left\{ \frac{1}{N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 \right\}$ subject to $\sum_{j=1}^{p} |\beta_j| \leq t.$
    - picks one correlated variable, others discarded. Sparse.
  - Ridge: minimize $\sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^{\mathsf{T}} \mathbf{z}_i)^2$ s.t. $\sum_{j=1}^{p} \beta_j^2 \leq t$
    - correlated variables coefficients are pushed to the same value
  - Elastic Net: $\arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1$
    - sparse solution, correlated variables grouped, enter/ leave the model together

Constraints for L1, L2 and L1&L2





LASSO Regression vs. Ridge Regression

GLM Best Practices

- Regularization Selection
  - Explore a few values for alpha, e.g. 0, 0.25, 0.5, 0.75, 1
- Wide Data Sets (10K+ columns)
  - Iteratively Reweighted Least Squares (IRLS) fails with lambda = 0
    - IRLS requires p x p Hessian matrix, where p = # of coefficients
    - Could use Limited-memory BFGS (L-BFGS)
  - IRLS + lambda search works and is recommended
    - Use alpha >> 0
    - Can produce 1K+ non-zero coefficients
  - L-BFGS + L2 penalty works
  - L-BFGS + L1 penalty works, but may take a long time

GLM Example

scikit-learn-like interface for modeling

```
y = "income"
x = ["age", "workclass", "fnlwgt", "education", "marital-status", "occupation", "relationship",
     "race", "sex", "capital-gain", "capital-loss","hours-per-week", "native-country"]
```

```
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
glm_0 = H2OGeneralizedLinearEstimator(family = "binomial", lambda_search = True,
                                      nfolds = 5, seed = 123)
glm_0.train(x = x, y = y, training_frame = census_data, model_id = "income_glm_0")
```

```
glm Model Build progress: |████████████████████████████████████████████████| 100%
```

```
from h2o.grid.grid_search import H2OGridSearch
glm_hyper_parameters = {"alpha": [0.5, 0.75, 1]}
glm_grid = H2OGridSearch(H2OGeneralizedLinearEstimator(family = "binomial", lambda_search = True,
                                                       nfolds = 5, seed = 123),
                         glm_hyper_parameters)
glm_grid.train(x = x, y = y, training_frame = census_data, grid_id = "income_glm_grid")
```

```
glm Grid Build progress: |████████████████████████████████████████████████| 100%
```