Building a Robot Judge: Data Science for Decision-Making

2. Machine Learning Essentials

Instructions before we begin:
(1) Turn on video and set audio to mute
(2) In Participants panel, set zoom name to "Full Name, School / Degree"
(ex: "Leon Smith, ETH Data Science Msc")
(3) If this is your first lecture, say "hi" in the chat

Online Lecture Norms

Let's make the most of online learning!

- Live attendance at lectures is required.
- Keep video on if connection allows.
- Stay muted when not talking.
- To make questions or comments, type in the chat (private or public) or use the "raise hand" function.

Question/Comment Padlets

- We have a dedicated question/comment padlet for each lecture week.
- The URL's will be of the form `bit.ly/BRJ_Padlet#`, where # is the week number, 2 through 13.
  - e.g. this week is `bit.ly/BRJ_Padlet2`, next week is `bit.ly/BRJ_Padlet3`.
- Please post things before class, during class, or during the mid-lecture break – we will go over them at the beginning of each lecture segment, or in the TA sessions.

TA Session

- Any feedback on the first TA session?
  - video is in the recordings folder
- Second TA session is Friday, 2pm-3pm:
  - go over this week's notebook
  - answer questions about the homework
  - setting up Stata.

Course Exam Info

- For those not doing a project, there is a take-home exam distributed in late January.
  - Questions will be based on the slides.
  - Will distribute practice questions beforehand.

Themes from last week's group discussions

https://padlet.com/eash44/xhwxvta7903zujw9

When machines might be more fair: **When humans might be more fair:**

Themes from last week's group discussions

`https://padlet.com/eash44/xhwxvta7903zujw9`

When machines might be more fair:	**When humans might be more fair:**
well-defined, observable characteristics (e.g. race/gender)	poorly defined or modifiable characteristics (e.g. being low income)

Themes from last week's group discussions

https://padlet.com/eash44/xhwxvta7903zujw9

When machines might be more fair:	**When humans might be more fair:**
well-defined, observable characteristics (e.g. race/gender)	poorly defined or modifiable characteristics (e.g. being low income)
when the data is unbiased	when the data is biased

Themes from last week's group discussions

`https://padlet.com/eash44/xhwxvta7903zujw9`

When machines might be more fair:	**When humans might be more fair:**
well-defined, observable characteristics (e.g. race/gender)	poorly defined or modifiable characteristics (e.g. being low income)
when the data is unbiased	when the data is biased
when humans are biased	when humans are unbiased

Group Discussion: Real-World Algorithmic Rating System

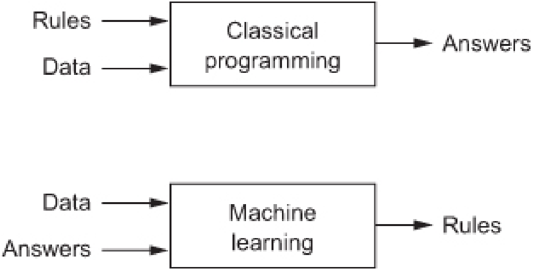
- Based on your breakout room number (to be assigned), discuss one of these articles:
  - Breakout rooms i $\leq$ N/2: `bit.ly/UK-visas` (Visa Algorithm)
  - Breakout rooms i $>$ N/2: `bit.ly/UK-exams` (Grading Algorithm)
- Assignment (8 minutes):
  - 2 minutes: say hello and introduce yourselves.
  - 1 minute: one student should summarize/describe the ML decision system described in the article.
  - 5 minutes: brainstorm at least 2 ways the system could be improved.
  - Write down in the padlet (not writeable for 6 minutes): `https://padlet.com/eash44/qbtf48rk6m5e540s`

Outline

Learning Objectives

1. **Implement and evaluate machine learning pipelines.**
   - **Evaluate (find problems in) existing machine learning pipelines.**
   - **Design a pipeline to solve a given ML problem.**
   - **Implement some standard pipelines in Python.**
2. Implement and evaluate causal inference designs.
3. Understand how (not) to use data science tools (ML and CI) to support expert decision-making.

What is machine learning?



- In classical computer programming, humans input the rules and the data, and the computer provides answers.
- In machine learning, humans input the data and the answers, and the computer learns the rules.

The standard learning problem

- We have a dataset of predictors or features, represented as a big matrix X.
  - e.g., defendant characteristics, criminal history, etc.

The standard learning problem

- We have a dataset of predictors or features, represented as a big matrix X.
  - e.g., defendant characteristics, criminal history, etc.
- The outcome or label to predict, Y
  - e.g., whether a defendant will commit more crimes if released on bail.

The standard learning problem

- We have a dataset of predictors or features, represented as a big matrix X.
  - e.g., defendant characteristics, criminal history, etc.
- The outcome or label to predict, Y
  - e.g., whether a defendant will commit more crimes if released on bail.
- The label is a probabilistic function of the features:

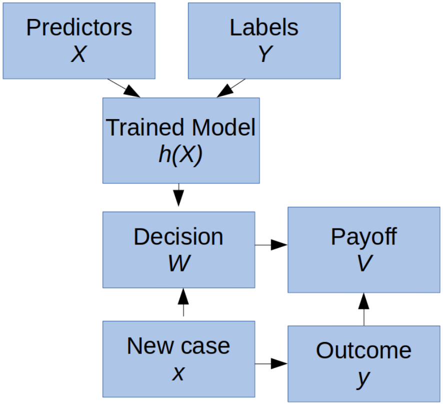
$$Y = h(X)$$

A decision problem

Now consider a decision-maker who has to make a decision W, that will produce some value or benefit, conditional on the value of Y:

$$V = u(W, Y)$$

- the decision-maker only observes X.
  - e.g., whether to grant bail.

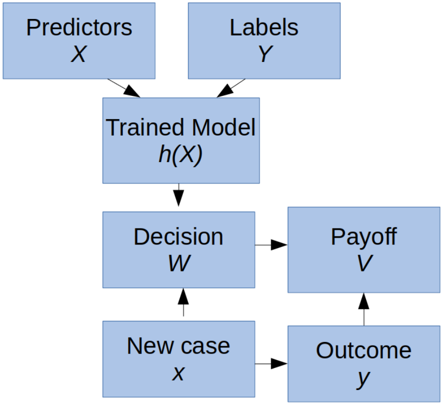


A decision problem

Now consider a decision-maker who has to make a decision W, that will produce some value or benefit, conditional on the value of Y:

$$V = u(W, Y)$$

- the decision-maker only observes X.
  - e.g., whether to grant bail.

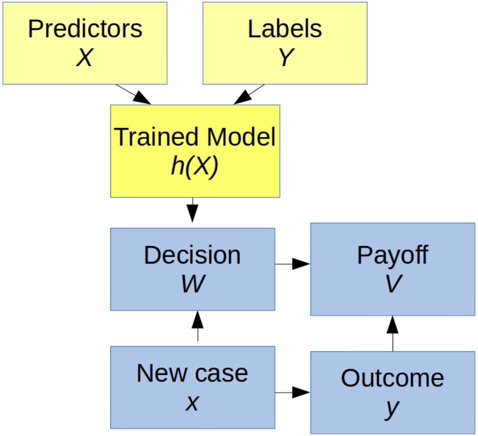


- The decision-maker computes a prediction $\hat{Y} = \hat{h}(X)$ and decides

$$W^*(X) = \arg\max_W u(W, \hat{Y}(X))$$

  - after Y is observed, the payoff is $u(W^*(X), Y)$.

Machine Learning for Decision-Making

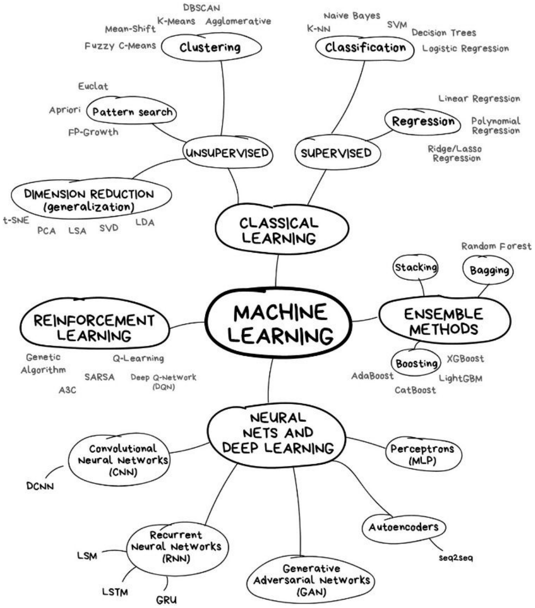


Today we focus on the prediction part:

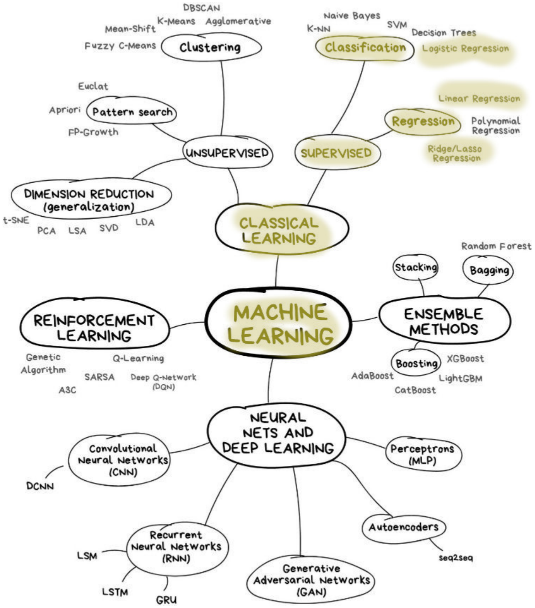
- How should the decision-maker learn $\hat{h}()$ to predict $\hat{Y}_i$ given a new case i with facts X_i.

Outline

The Machine Learning Landscape



What we will do today



A Machine Learning Project, End-to-End

Aurelien Geron, *Hands-on machine learning with Scikit-Learn & TensorFlow*, Chapter 2:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Three Types of (Standard) Machine Learning Problems

Determined by the data type of the outcome variable (or label):

- **Binary classification**: two choices, normalized to zero and one.
  - e.g., guilty or innocent

Three Types of (Standard) Machine Learning Problems

Determined by the data type of the outcome variable (or label):

- **Binary classification**: two choices, normalized to zero and one.
  - e.g., guilty or innocent
- **Regression**: a one-dimensional, continuous, real-valued outcome.
  - e.g., number of days of prison assigned

Three Types of (Standard) Machine Learning Problems

Determined by the data type of the outcome variable (or label):

- **Binary classification**: two choices, normalized to zero and one.
  - e.g., guilty or innocent
- **Regression**: a one-dimensional, continuous, real-valued outcome.
  - e.g., number of days of prison assigned
- **Multinomial Classification**: Three or more discrete, un-ordered outcomes.
  - e.g., predict what judge is assigned to a case: Alito, Breyer, or Cardozo

Poll 2.2: What type of ML Problem is this?

- Based on defendant characteristics and the facts of the case, predict which charges the prosecutor will bring:
  - third degree murder (manslaughter)
  - second degree murder (crime of passion)
  - first degree murder (premeditated)

What do ML Algorithms do? Minimize a cost function

- A typical cost function for regression problems is Mean Squared Error (MSE):

$$\text{MSE}(x, h) = \frac{1}{n_D} \sum_{i=1}^{n_D} (h(x_i; \theta) - y_i)^2$$

  - n_D, the number of rows/observations in dataset
  - x, the feature set, with row x_i
  - y, the set of outcomes, with item y_i
  - $h(x_i; \theta) = \hat{y}$ the model prediction (hypothesis)

Loss functions, more generally

- The loss function $L(\hat{y}, y)$ assigns a score based on prediction and truth:
  - Should be bounded from below, with the minimum attained only for cases where the prediction is correct.
- The average loss for the test set is

$$\mathcal{L}(\theta) = \frac{1}{n_D} \sum_{i=1}^{n_D} L(h(\boldsymbol{x}_i; \theta), y_i)$$

  - $\rightarrow$ treats the data as constants; parameters determine the loss.
- The estimated parameter vector θ solves

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)$$

OLS Regression is Machine Learning

- Ordinary Least Squares Regression (OLS) assumes the functional form $h(x;\theta) = x_i'\theta$ and minimizes the MSE

$$\min_{\hat{\theta}} \frac{1}{n_D} \sum_{i=1}^{n_D} (x_i'\hat{\theta} - y_i)^2$$

OLS Regression is Machine Learning

- Ordinary Least Squares Regression (OLS) assumes the functional form $h(x;\theta) = x_i'\theta$ and minimizes the MSE

$$\min_{\hat{\theta}} \frac{1}{n_D} \sum_{i=1}^{n_D} (x_i'\hat{\theta} - y_i)^2$$

- This minimand has a closed form solution:

$$\hat{\theta} = (\boldsymbol{x}'\boldsymbol{x})^{-1}\boldsymbol{x}'\boldsymbol{y}$$

  - most machine learning models do **not** have a closed form solution.

How to solve without a closed-form solution? Gradient Descent

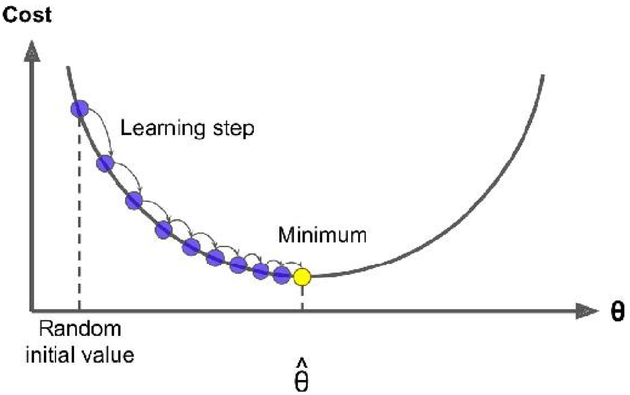


Figure 4-3. Gradient Descent

- Gradient descent measures the local gradient of the error function, and then steps in that direction.
  - Once the gradient equals zero, you have reached a minimum.

$$\text{MSE}(\theta) = \frac{1}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = \frac{2}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.

$$\mathrm{MSE}(\theta) = \frac{1}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \mathrm{MSE}}{\partial \theta_j} = \frac{2}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.
- The *gradient* ∇ gives the vector of these partial derivatives:

$$\nabla_\theta \mathrm{MSE} = \begin{bmatrix} \frac{\partial \mathrm{MSE}}{\partial \theta_1} \\ \frac{\partial \mathrm{MSE}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathrm{MSE}}{\partial \theta_j} \end{bmatrix} = \frac{2}{m}\boldsymbol{X}'(\boldsymbol{X}'\theta - \boldsymbol{y})$$

$$\text{MSE}(\theta) = \frac{1}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = \frac{2}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.
- The *gradient* ∇ gives the vector of these partial derivatives:

$$\nabla_\theta \text{MSE} = \begin{bmatrix} \frac{\partial \text{MSE}}{\partial \theta_1} \\ \frac{\partial \text{MSE}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \text{MSE}}{\partial \theta_j} \end{bmatrix} = \frac{2}{m}\boldsymbol{X}'(\boldsymbol{X}'\theta - \boldsymbol{y})$$

- Gradient descent nudges θ against the gradient (the direction that reduces MSE):

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta \text{MSE}$$

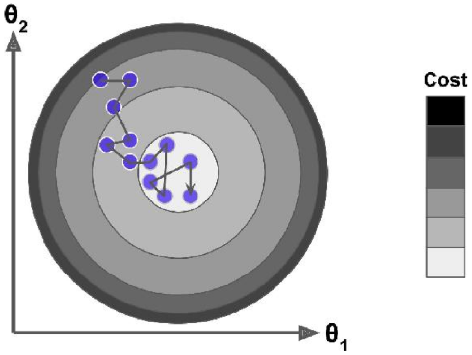
  - η = learning rate

Stochastic Gradient Descent

- SGD picks a random instance in the training set and computes the gradient only for that single instance.
  Nice tutorial implementing SGD from scratch to solve OLS:
  https://neuraspike.com/blog/linear-regression-stochastic-gradient-descent-python/

Stochastic Gradient Descent

- SGD picks a random instance in the training set and computes the gradient only for that single instance.
  Nice tutorial implementing SGD from scratch to solve OLS:
  `https://neuraspike.com/blog/linear-regression-stochastic-gradient-descent-python/`



- SGD is much faster to train, but bounces around even after it is close to the minimum.
  - Compromise: mini-batch gradient descent, selects a sample of rows (a "mini-batch") for gradient compute.

Training and Testing

- Machine learning models can achieve arbitrarily high accuracy in-sample.
  - performance should be evaluated out-of-sample.

Training and Testing

- Machine learning models can achieve arbitrarily high accuracy in-sample.
  - performance should be evaluated out-of-sample.
- standard approach:
  - randomly sample 80% training dataset to learn parameters
  - form predictions in 20% testing dataset for evaluating performance.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y)
```

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.
- Best practice:
  - **reproducible data pipeline**.

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.
- Best practice:
  - **reproducible data pipeline**.
  - if you want a "clean" evaluation in the test set, you have to **do data prep in the training set**.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer (preprocessor)*: An object that transforms a data set.
  - e.g. `preprocessing.StandardScaler`
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer (preprocessor)*: An object that transforms a data set.
  - e.g. `preprocessing.StandardScaler`
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - e.g. `LinearRegression`, after training
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer (preprocessor)*: An object that transforms a data set.
  - e.g. `preprocessing.StandardScaler`
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - e.g. `LinearRegression`, after training
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)

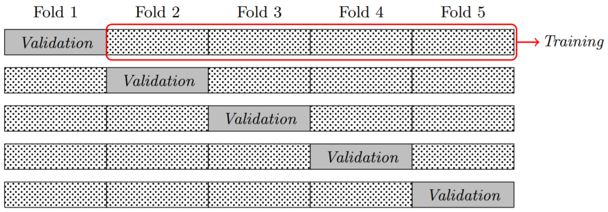
Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer (preprocessor)*: An object that transforms a data set.
  - e.g. `preprocessing.StandardScaler`
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - e.g. `LinearRegression`, after training
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)
  - **Non-proliferation of classes:** Use native Python data types; existing building blocks are used as much as possible.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters
  - e.g. `linear_models.LinearRegression`
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer (preprocessor)*: An object that transforms a data set.
  - e.g. `preprocessing.StandardScaler`
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - e.g. `LinearRegression`, after training
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)
  - **Non-proliferation of classes:** Use native Python data types; existing building blocks are used as much as possible.
  - **Sensible defaults:** Provides reasonable default values for hyperparameters – easy to get a good baseline up and running.

Use Cross-Validation During Model Training



- Within the training set:
  - Use **`cross_val_score`** method to get model performance across subsets of data.
  - Use `GridSearchCV` or `RandomizedSearchCV` to automate search over parameter space.
- Find the best hyperparameters for out-of-fold prediction in the training set.
  - then evaluate model performance in the test set.

Outline

Regression models $\leftrightarrow$ Continuous outcome

- If the outcome is continuous (e.g., Y = tax revenues collected, or criminal sentence imposed in months of prison):

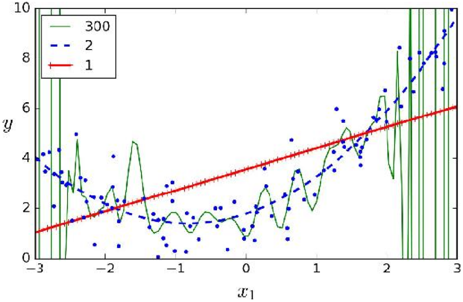


Figure 4-14. High-degree Polynomial Regression

- Need a regression model. Problems with OLS:
  - tends to over-fit training data.
  - cannot handle multicollinearity.

Regression models ↔ Continuous outcome

- If the outcome is continuous (e.g., $Y =$ tax revenues collected, or criminal sentence imposed in months of prison):

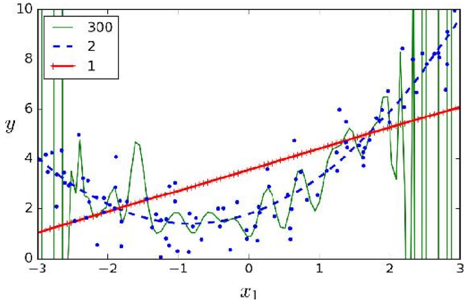


Figure 4-14. High-degree Polynomial Regression

- Need a regression model. Problems with OLS:
  - tends to over-fit training data.
  - cannot handle multicollinearity.
- Machine learning models are evaluated by the fit in held-out data (the test set)
  - "Regularization" refers to ML model training methods designed to reduce/prevent over-fitting of the training set
  - (and hopefully better fit in the test set).

Regularization

- Minimizing the loss L directly usually results in over-fitting. It is standard to add regularization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{n_D} \sum_{i=1}^{n_D} L(h(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i) + \lambda R(\theta)$$

  - $R(\theta)$ is a "regularization function" or "regularizer", designed to reduce over-fitting.
  - λ is a hyperparameter where higher values increase regularization.

Regularization

- Minimizing the loss L directly usually results in over-fitting. It is standard to add regularization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{n_D} \sum_{i=1}^{n_D} L(h(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i) + \lambda R(\theta)$$

  - $R(\theta)$ is a "regularization function" or "regularizer", designed to reduce over-fitting.
  - λ is a hyperparameter where higher values increase regularization.

In particular:

- "Lasso" (or L1) penalty:

$$R_1 = \|\theta\|_1 = \sum_{j=1}^{n_x} |\theta_j|$$

  - automatically performs feature selection and outputs a sparse model.

Regularization

- Minimizing the loss L directly usually results in over-fitting. It is standard to add regularization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{n_D} \sum_{i=1}^{n_D} L(h(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i) + \lambda R(\theta)$$

  - $R(\theta)$ is a "regularization function" or "regularizer", designed to reduce over-fitting.
  - λ is a hyperparameter where higher values increase regularization.

In particular:

- "Lasso" (or L1) penalty:

$$R_1 = \|\theta\|_1 = \sum_{j=1}^{n_x} |\theta_j|$$

  - automatically performs feature selection and outputs a sparse model.
- "Ridge" (or L2) penalty:

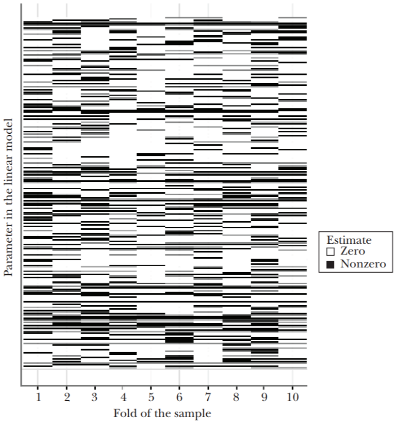
$$R_2 = \|\theta\|_2^2 = \sum_{j=1}^{n_x} (\theta_j)^2$$

  - shrinks coefficients toward zero and helps select between collinear predictors.

Does lasso pick the “true” model?

Lasso prediction of house prices with 150 variables – which variables are “selected” (non-zero coefficients) by lasso, in ten models trained on separate data subsamples (Mullainathan and Spiess 2017):

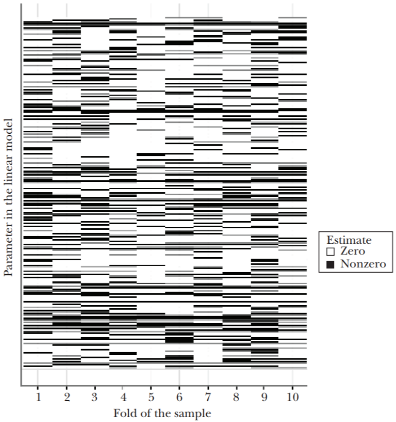
Selected Coefficients (Nonzero Estimates) across Ten LASSO Regressions



Does lasso pick the "true" model?

Lasso prediction of house prices with 150 variables – which variables are "selected" (non-zero coefficients) by lasso, in ten models trained on separate data subsamples (Mullainathan and Spiess 2017):

Selected Coefficients (Nonzero Estimates) across Ten LASSO Regressions



- The set of lasso-selected variables changes across folds in the data
- $\rightarrow$ Lasso does not pick the "correct" predictors.
  - It just learns the correct $\hat{h}(X)$
  - when predictors are correlated with each other, they are substitutable.

Elastic Net = Lasso + Ridge

The Elastic Net cost function is:

$$
\begin{aligned}
L(\theta) &= \mathrm{MSE}(\theta) + \lambda_1 R_1 + \lambda_2 R_2 \\
&= \mathrm{MSE}(\theta) + \lambda_1 \sum_{j=1}^{n_x} |\theta_j| + \lambda_2 \sum_{j=1}^{n_x} (\theta_j)^2
\end{aligned}
$$

- $\lambda_1, \lambda_2 =$ strength of L1 (Lasso) penalty and L2 (Ridge) penalty, respectively.

Elastic Net = Lasso + Ridge

The Elastic Net cost function is:

$$\begin{aligned} L(\theta) &= \text{MSE}(\theta) + \lambda_1 R_1 + \lambda_2 R_2 \\ &= \text{MSE}(\theta) + \lambda_1 \sum_{j=1}^{n_x} |\theta_j| + \lambda_2 \sum_{j=1}^{n_x} (\theta_j)^2 \end{aligned}$$

- λ_1, λ_2 = strength of L1 (Lasso) penalty and L2 (Ridge) penalty, respectively.

The derivative for feature j is

$$\frac{\partial L}{\partial \theta_j} = \frac{2}{n_D} \sum_{i=1}^{n_D} (\boldsymbol{x}_i' \hat{\theta} - y_i) x_i^j + \lambda_1^* + 2\lambda_2 \theta_j$$

where replace $\lambda_1^* = -\lambda_1$ if $\theta_j < 0$.

Elastic Net = Lasso + Ridge

The Elastic Net cost function is:

$$
\begin{aligned}
L(\theta) &= \mathrm{MSE}(\theta) + \lambda_1 R_1 + \lambda_2 R_2 \\
&= \mathrm{MSE}(\theta) + \lambda_1 \sum_{j=1}^{n_x} |\theta_j| + \lambda_2 \sum_{j=1}^{n_x} (\theta_j)^2
\end{aligned}
$$

- λ_1, λ_2 = strength of L1 (Lasso) penalty and L2 (Ridge) penalty, respectively.

The derivative for feature j is

$$
\frac{\partial L}{\partial \theta_j} = \frac{2}{n_D} \sum_{i=1}^{n_D} (\boldsymbol{x}_i' \hat{\theta} - y_i) x_i^j + \lambda_1^* + 2\lambda_2 \theta_j
$$

where replace $\lambda_1^* = -\lambda_1$ if $\theta_j < 0$.

In scikit-learn, e-net penalties are parametrized as "`alpha`" = total penalty, and "`l1_ratio`" = proportion of penalty to L1.

```
from sklearn.linear_model import ElasticNet
enet = ElasticNet(alpha=2.0, l1_ratio = .75) # L1 = 1.5, L2 = 0.5
enet.fit(X,y)
```

Selecting Elastic Net Hyperparameters

- Elastic net hyperparameters should be selected to optimize out-of-sample fit (measured by mean squared error or MSE).
- "Grid search" scans over the hyperparameter space ($\lambda_1 \geq 0$,$\lambda_2 \geq 0$), computes out-of-sample MSE for all pairs (λ_1, λ_2), and selects the MSE-minimizing model.

```
from sklearn.model_selection import GridSearchCV
param_grid = [{'alpha':  [0.1, 1, 10]; 'l1_ratio=[0,.5,1]}]
grid = GridSearchCV(enet, param_grid)
grid.fit(X_train, y_train)
```

Evaluating Regression Models: R^2

- Mean squared error is good for comparing regression models, but the units depend on the outcome variable and therefore are not interpretable.
  - Better to use R^2 in the test set, which has same ranking as MSE but it more interpretable.

```
from sklearn.metrics import r2_score
r2_score(y_test,y_test_pred)
```

Breakout Rooms: Regression Practice

- See zoom chat for links to colab notebooks.
- if you get stumped, google it.

Outline

Binary Outcome $\leftrightarrow$ Binary Classification

- Binary classifiers try to match a boolean outcome $y \in \{0, 1\}$.
  - The standard approach is to apply a transformation (e.g. sigmoid/logit) to normalize $\hat{y} \in [0, 1]$.
  - Prediction rule is 0 for $\hat{y} < .5$ and 1 otherwise.

Binary Outcome $\leftrightarrow$ Binary Classification

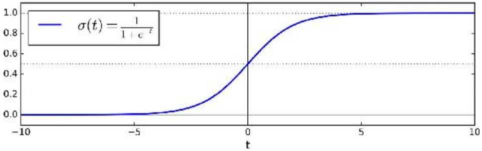
- Binary classifiers try to match a boolean outcome $y \in \{0, 1\}$.
  - The standard approach is to apply a transformation (e.g. sigmoid/logit) to normalize $\hat{y} \in [0, 1]$.
  - Prediction rule is 0 for $\hat{y} < .5$ and 1 otherwise.
- The binary cross-entropy (or log loss) is:

$$L(\theta) = \underbrace{-\frac{1}{n_D}}_{\text{negative}} \sum_{i=1}^{n_D} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{y}_i)}_{\text{log prob}y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{y}_i)}_{\text{log prob}y_i=0}]$$

Logistic Regression

- In logistic regression (also in the output layer of binary neural net classifiers) we use a sigmoid transformation:

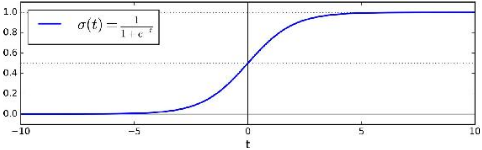
$$\hat{y} = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$



Logistic Regression

- In logistic regression (also in the output layer of binary neural net classifiers) we use a sigmoid transformation:

$$\hat{y} = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$



- Plugging into the binary-cross entropy loss gives the logistic regression cost objective:

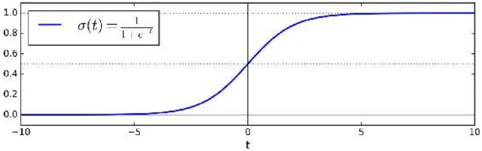
$$\min_{\theta} \sum_{i=1}^{n_D} -y_i \log(\text{sigmoid}(\boldsymbol{x}_i \cdot \theta)) - [1 - y_i] \log(1 - \text{sigmoid}(\boldsymbol{x}_i \cdot \theta))$$

  - does not have a closed form solution, but it is convex (guaranteeing that gradient descent will find the global minimum).

Logistic Regression

- In logistic regression (also in the output layer of binary neural net classifiers) we use a sigmoid transformation:

$$\hat{y} = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$



- Plugging into the binary-cross entropy loss gives the logistic regression cost objective:

$$\min_{\theta} \sum_{i=1}^{n_D} -y_i \log(\text{sigmoid}(\boldsymbol{x}_i \cdot \theta)) - [1 - y_i] \log(1 - \text{sigmoid}(\boldsymbol{x}_i \cdot \theta))$$

  - does not have a closed form solution, but it is convex (guaranteeing that gradient descent will find the global minimum).

- The gradient is

$$\frac{\partial L(\theta)}{\partial \theta_j} = \frac{1}{n_D} \sum_{i=1}^{n_D} [\underbrace{(\text{sigmoid}(\boldsymbol{x}_i \cdot \theta) - y_i)}_{\text{error for obs } i} \underbrace{x_i^j}_{\text{input } j}]$$

  - SGD will follows this at learning rate η until convergence.

Regularized Logistic Regression

- Like linear regression, logistic regression can be regularized with L1 or L2 penalties
- e.g., ridge penalty

$$L_2(\theta) = \mathrm{L}(\theta) + \lambda_2 \sum_{j=1}^{n_x} \theta_j^2$$

with gradient

$$\frac{\partial L_2(\theta)}{\partial \theta_j} = \frac{1}{n_D} \sum_{i=1}^{n_D} [(\text{sigmoid}(\boldsymbol{x}_i \cdot \theta) - y_i) x_i^j + 2\lambda_2 \theta_j]$$

Regularized Logistic Regression

- Like linear regression, logistic regression can be regularized with L1 or L2 penalties
- e.g., ridge penalty

$$L_2(\theta) = \mathrm{L}(\theta) + \lambda_2 \sum_{j=1}^{n_x} \theta_j^2$$

with gradient

$$\frac{\partial L_2(\theta)}{\partial \theta_j} = \frac{1}{n_D} \sum_{i=1}^{n_D} [(\mathrm{sigmoid}(\boldsymbol{x}_i \cdot \theta) - y_i) x_i^j + 2\lambda_2 \theta_j]$$

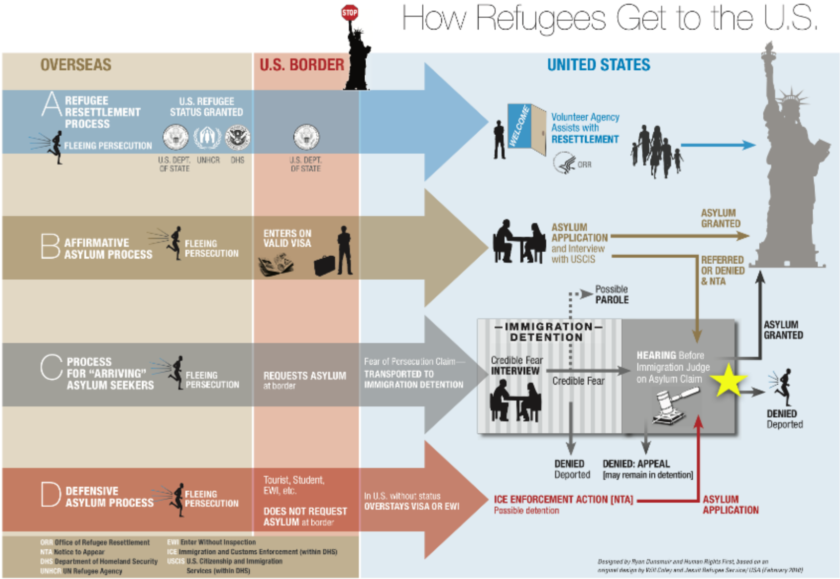
- In scikit-learn, by default LogisticRegression uses the ridge penalty, where C is the inverse of λ:

```
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression(penalty='l2', C = 2.0) # lambda = 1/2
logit.fit(X,y)
```

Outline

Outline

Asylum in the U.S.



Source: rcusa.org.

Dunn, Sagun, Sirin, and Chen (2017): Asylum Courts

- Data:
  - universe of asylum court cases, 1981-2013
  - 492,903 decisions, 336 courts, 441 judges

Dunn, Sagun, Sirin, and Chen (2017): Asylum Courts

- Data:
  - universe of asylum court cases, 1981-2013
  - 492,903 decisions, 336 courts, 441 judges
- High stakes: denial of asylum results in deportation.
- Average grant rate: 35%.

Zoom Poll 2.3: What type of ML Problem is this?

Predicting U.S. Asylum Court Decisions

		Predicted	
		Denied	Granted
True	Denied	195,223	65,798
	Granted	73,269	104,406

Accuracy = 68.3%, F1 = 0.60

- Prediction App (Beta): `https://floating-lake-11821.herokuapp.com/`
  - predictions made using logistic regression with L2 regularization, penalty selected by cross-validation grid search.

Judge Identity is Most Predictive Factor

Model	Accuracy	ROC AUC
Judge ID	0.71	0.74
Judge ID & Nationality	0.76	0.82
Judge ID & Opening Date	0.73	0.77
Judge ID & Nationality & Opening Date	0.78	0.84
Full model at case completion	0.82	0.88

- Predictions from random forest classifier, with parameters selected by cross-validated grid search.
  - Training/test split 482K/120K.

Judge Variation in Predictability

- Some judges are highly predictable, always granting or rejecting.
  - suggests they use heuristics or stereotypes rather than considering cases carefully.

Judge Variation in Predictability

- Some judges are highly predictable, always granting or rejecting.
  - suggests they use heuristics or stereotypes rather than considering cases carefully.
- There is significant variation in predictability by judge, conditional on grant rate.
  - suggests disagreement about circumstances contributing to asylum decision.

Outline

Poll 2.4: What type of ML Problem is this?

http://bit.ly/BRJ_bonica

Abstract: *This article develops a generalized supervised learning methodology for inferring roll-call scores from campaign contribution data. Rather than use unsupervised methods to recover a latent dimension that best explains patterns in giving, donation patterns are instead mapped onto a target measure of legislative voting behavior. Supervised models significantly outperform alternative measures of ideology in predicting legislative voting behavior. Fundraising prior to entering office provides a highly informative signal about future voting behavior. Impressively, forecasts based on fundraising as a nonincumbent predict future voting behavior as accurately as in-sample forecasts based on votes cast during a legislator's first 2 years in Congress. The combined results demonstrate campaign contributions are powerful predictors of roll-call voting behavior and resolve an ongoing debate as to whether contribution data successfully distinguish between members of the same party.*

Outline

Course Project Logistics

https://bit.ly/BRJ_proj

- If you are signed up for the credits, the focus of your work in this course should be on the project.
  - Can be done individually or in small groups (up to 4 students).
  - Do an original analysis using methods learned in the course, and write a paper about it.

Course Project Logistics

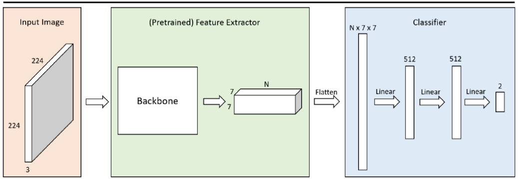
`https://bit.ly/BRJ_proj`

- If you are signed up for the credits, the focus of your work in this course should be on the project.
  - Can be done individually or in small groups (up to 4 students).
  - Do an original analysis using methods learned in the course, and write a paper about it.
- Deliverables:
  - description of topic (October 26th 10% of grade)
  - proposal/outline (November 23rd, 10% of grade)
  - Poster session (early December, 10% of grade).
  - Rough draft with data/methods/results (January 4th 2021, 20% of grade)
  - Final draft (February 1st 2021, 50% of grade)

Last Year's Projects (1)

Dominik Borer: Predicting Candidate Party from Political Television Ads

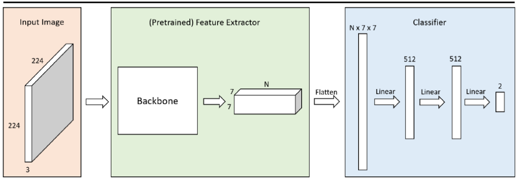
Figure 3: Overview of Model Architecture



Last Year's Projects (1)

Dominik Borer: Predicting Candidate Party from Political Television Ads

Figure 3: Overview of Model Architecture



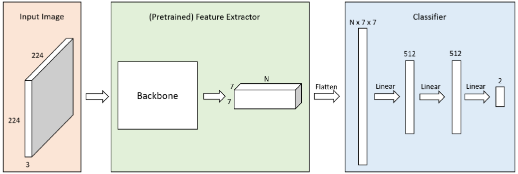
Panel C. Confusion matrix for test set

	Predicted Democratic	**Predicted Republican**
Actual Democratic	44.88% (793)	7.98% (141)
Actual Republican	15.73% (278)	31.41% (555)

Last Year's Projects (1)

Dominik Borer: Predicting Candidate Party from Political Television Ads

Figure 3: Overview of Model Architecture



Panel C. Confusion matrix for test set

	Predicted Democratic	Predicted Republican
Actual Democratic	44.88% (793)	7.98% (141)
Actual Republican	15.73% (278)	31.41% (555)

Panel A: News Show Images with Highest Democrat Slant

Panel B: News Show Images with Highest Republican Slant

Last Year's Projects (2)

Philip Nikolaus: Deep Instrumental Variables for Causal Effects of Judicial Text

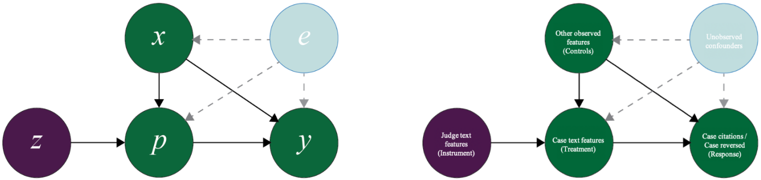


Figure 1: (Left) Directed graph with arrows representing causal effects (Hartford et al. 2017). (Right) Directed graph showing the causal relationship structure for this work.

Last Year's Projects (2)

Philip Nikolaus: Deep Instrumental Variables for Causal Effects of Judicial Text

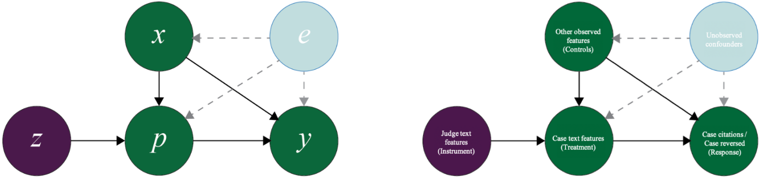


Figure 1: (Left) Directed graph with arrows representing causal effects (Hartford et al. 2017). (Right) Directed graph showing the causal relationship structure for this work.

Top 20 Words in Cluster	**Feature Importance (Deep-IV – OLS)**
cognitive, concentrating, moderate, cognition, functioning, concentration, cope, psychomotor, ability, impulsivity, socialization, difficulty, auditory, interact, mood, pace, irritability, learning, impaired, distractibility	0.245
misappropriation, breach, tortious, tort, misappropriate, fiduciary, enrich, enrichment, interference, secret, wrongful, misappropriated, meruit, trade, conversion, dealing, contractual, contract, defamation, negligent	0.0416
consonant, embody, embrace, rigid, traditional, hew, engraft, dictate, eschew, adherence, incompatible, recognize, animate, jurisprudential, universal, concurring, modern, antithetical, comport, override	0.0314
complicate, depend, crucial, illustrate, elusive, focus, important, straightforward, elide, critical, underscore, differ, complicated, nuance, conceptual, precise, central, grapple, particular, murky	0.0309
coverage, insured, insurer, insure, indemnity, indemnification, insurance, indemnify, reinsurer, uninsured, subrogation, endorsement, cover, umbrella, policyholder, policy, covered, indemnitee, reinsure, insuring	0.0234
conclusory, Twombly550, plausible, Iqbal556, bare, true, allegation, twombly127, formulaic, plausibility, speculative, averment, survive, enough, mere, threadbare, bald, twombly550, recital, suffice	0.0229

Last Year's Projects (3)

- One of the groups began building a legal research application for Swiss lawyers:
  - feature-rich legal search engine.
  - MSWord plugin that would suggest cite references as you type.
  - Already partnered with Homberger law firm to test it out.

Last Year's Projects (3)

- One of the groups began building a legal research application for Swiss lawyers:
  - feature-rich legal search engine.
  - MSWord plugin that would suggest cite references as you type.
  - Already partnered with Homberger law firm to test it out.
- Another group partnered with a local company to build out their environmental-regulation analytics
  - won an Innosuisse grant.

Last Year's Projects (3)

- One of the groups began building a legal research application for Swiss lawyers:
  - feature-rich legal search engine.
  - MSWord plugin that would suggest cite references as you type.
  - Already partnered with Homberger law firm to test it out.
- Another group partnered with a local company to build out their environmental-regulation analytics
  - won an Innosuisse grant.
- *Note: These projects were above expectation.*

New Project Ideas

- We have a list of potential project ideas, will share with interested students.