*What are important prob. types in DM?
-Finding patterns
-Clustering
-Classification
-Regression
-Ranking
-Density Estimation
*Describe them shortly?write down their notations? give examples?

Fundamentals of Learning

MIT 15.097 Course Notes
Cynthia Rudin

Important Problems in Data Mining

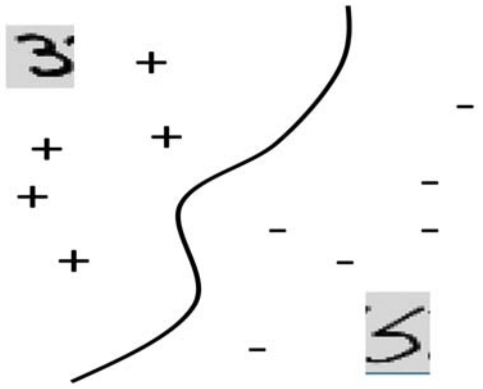
1. Finding patterns (correlations) in large datasets
   -e.g. (Diapers $\rightarrow$ Beer). Use Apriori!

2. Clustering - grouping data into clusters that "belong" together - objects within a cluster are more similar to each other than to those in other clusters.

   - Kmeans, Kmedians
   - Input: $\{x_i\}_{i=1}^m, x_i \in \mathcal{X} \subset \mathbf{R}^n$
   - Output: $f : \mathcal{X} \rightarrow \{1, \dots, K\}$ (K clusters)
   - clustering consumers for market research, clustering genes into families, image segmentation (medical imaging)

3. Classification

   - Input: $\{(x_i, y_i)\}_{i=1}^m$ "examples," "instances with labels," "observations"
   - $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$ "binary"



   - Output: $f : \mathcal{X} \rightarrow \mathbf{R}$ and use $sign(f)$ to classify.
   - automatic handwriting recognition, speech recognition, biometrics, document classification
   - "LeNet"

4. Regression

- Input: $\{(x_i, y_i)\}_{i=1}^m$, $x_i \in \mathcal{X}, y_i \in \mathbf{R}$
- Output: $f : \mathcal{X} \to \mathbf{R}$
- predicting an individual's income, predict house prices, predict stock prices, predict test scores

5. Ranking (later) - in between classification and regression. Search engines use ranking methods

6. Density Estimation - predict conditional probabilities

   - $\{(x_i, y_i)\}_{i=1}^m$, $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$
   - Output: $f : \mathcal{X} \to [0, 1]$ as "close" to $P(y = 1|x)$ as possible.
   - estimate probability of failure, probability to default on loan

Rule mining and clustering are **unsupervised methods** (no ground truth), and classification, ranking, and density estimation are **supervised methods** (there is ground truth). In all of these problems, we do not assume we know the distribution that the data are drawn from!

*What is the input and output in training?
*What is the input and output in the testing?

Training and Testing (in-sample and out-of-sample) for supervised learning

Training: training data are input, and model f is the output.

$$\{(x_i, y_i)\}_{i=1}^m \Longrightarrow \boxed{\text{Algorithm}} \Longrightarrow f.$$

Testing: You want to predict y for a new x, where (x, y) comes from the same distribution as $\{(x_i, y_i)\}_{i=1}^m$.

That is, $(x, y) \sim D(\mathcal{X}, \mathcal{Y})$ and each $(x_i, y_i) \sim D(\mathcal{X}, \mathcal{Y})$.

*What exactly is testing means in more technical terms?

✓ Compute $f(x)$ and compare it to y. How well does $f(x)$ match y? Measure goodness of f using a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$:

$$\begin{aligned} R^{\text{test}}(f) &= \mathbf{E}_{(x,y)\sim D}\ell(f(x), y) \\ &= \int_{(x,y)\sim D} \ell(f(x), y)dD(x, y). \end{aligned}$$

*What are other names of R (test)?

R^{test} is also called the **true risk** or the **test error**.

$\boxed{\text{Can we calculate } R^{\text{test}}?}$

✓ We want R^{test} to be small, to indicate that $f(x)$ would be a good predictor ("estimator") of y.

For instance

$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 && \text{least squares loss, or} \\ \ell(f(x), y) &= \mathbf{1}_{[\text{sign}(f(x)) \neq y]} && \text{(mis)classification error}\end{aligned}$$

→ ANN

$\boxed{\text{Which problems might these loss functions correspond to?}}$

How can we ensure $R^{\text{test}}(f)$ is small?

Look at how well f performs (on average) on $\{(x_i, y_i)\}_i$.

$$R^{\text{train}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i), y_i).$$

*What are other names of training error?

R^{train} is also called the **empirical risk** or **training error**. For example,

$$R^{\text{train}}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{[\text{sign}(f(x_i)) \neq y_i]}.$$

(How many handwritten digits did f classify incorrectly?)

Say our algorithm constructs f so that $R^{\text{train}}(f)$ is small. If $R^{\text{train}}(f)$ is small, hopefully $R^{\text{test}}(f)$ is too.

We would like a guarantee on how close R^{train} is to R^{test}. When would it be close to R^{test}?

When Rtrain is close to Rtest?

- If m is large.
- If f is "simple."

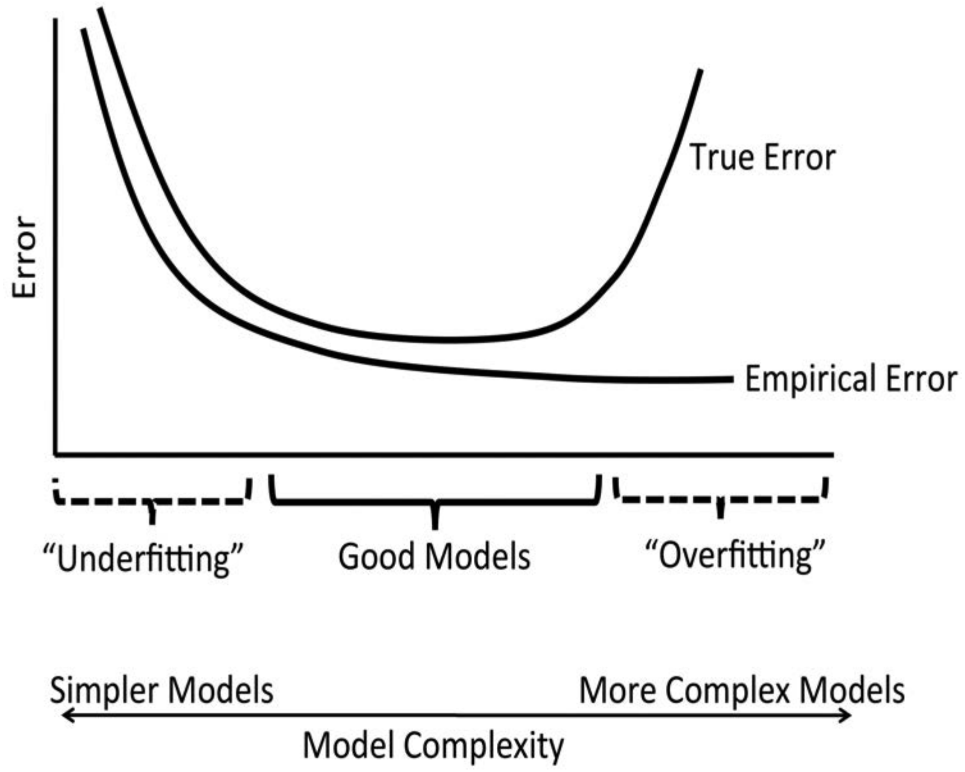
Illustration

In one of the figures in the illustration, f:

- was overfitted to the data
- modeled the noise
- "memorized" the examples, but didn't give us much other useful information
- doesn't "generalize," i.e., predict. We didn't "learn" anything!

*What is structural risk minimization shortly?explain on graph?

Computational Learning Theory, a.k.a. Statistical Learning Theory, a.k.a., learning theory, and in particular, Vapnik's **Structural Risk Minimization** (SRM) addresses generalization. Here's SRM's classic picture:



Which is harder to check for, overfitting or underfitting?

bunun ustunde dusn?

Computational learning theory addresses how to construct probabilistic guarantees on the true risk. In order to do this, it quantifies the class of "simple models."

Bias/Variance Tradeoff is related to learning theory (actually, bias is related to learning theory).

Inference Notes - Bias/Variance Tradeoff

Why is Bias Variance Tradeoff?
If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data

In supervised learning, underfitting happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kind of models are very simple to capture the complex patterns in data like Linear and logistic regression.In supervised learning, overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.

Regularized Learning Expression

✓ Structural Risk Minimization says that we need some bias in order to learn/generalize (avoid overfitting). Bias can take many forms:

- "simple" models $f(x) = \sum_j \lambda_j x^{(j)}$ where $\|\lambda\|_2^2 = \sum_j \lambda_j^2 < C$
- "prior" in Bayesian statistics
- connectivity of neurons in the brain

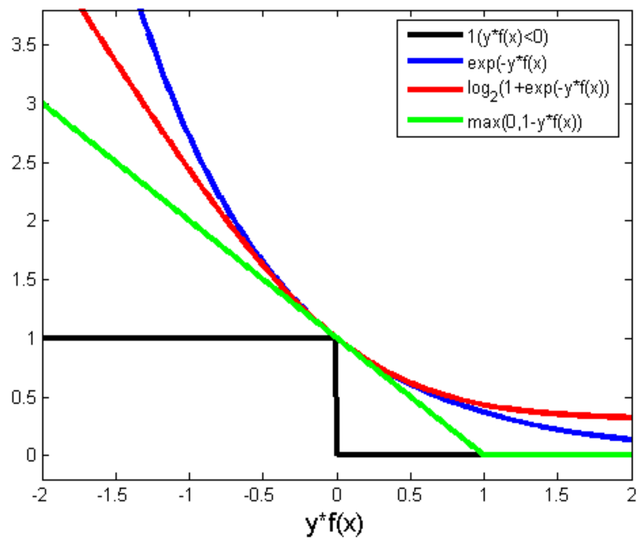
Regularized Learning Expression:

$$\sum_i \ell(f(x_i), y_i) + CR^{\text{reg}}(f)$$

This expression is kind of omnipresent. This form captures many algorithms: SVM, boosting, ridge regression, LASSO, and logistic regression.

In the regularized learning expression, the loss $\ell(f(x_i), y_i)$ could be:

- ✓ "least squares loss" $(f(x_i) - y_i)^2$
- ✓ "misclassification error" $\mathbf{1}_{[y_i \neq \text{sign}(f(x_i))]} = \mathbf{1}_{[y_i f(x_i) \leq 0]}$
  - Note that minimizing $\sum_i \mathbf{1}_{[y_i f(x_i) \leq 0]}$ is computationally hard.



- ✓ "logistic loss" $\log_2\left(1 + e^{-y_i f(x_i)}\right) \Longleftarrow$ logistic regression

✓ - "hinge loss" $\max(0, 1 - y_i f(x_i)) \Longleftarrow$ SVM
✓ - "exponential loss" $e^{-y_i f(x_i)} \Longleftarrow$ AdaBoost

In the regularized learning expression, we define a couple of options for $R^{\text{reg}}(f)$. Usually f is linear, $f(x) = \sum_j \lambda_j x^{(j)}$. We choose $R^{\text{reg}}(f)$ to be either:

- $\|\lambda\|_2^2 = \sum_j \lambda_j^2 \Longleftarrow$ ridge regression, SVM
- $\|\lambda\|_1 = \sum_j |\lambda_j| \Longleftarrow$ LASSO, approximately AdaBoost