Week 1: Introduction to Predictive Modeling Framework

1. Data Mining Process (cyclical)
    a. Business & Data Understanding
    b. Data Preparation
    c. Modeling
    d. Evaluation
    e. Deployment

2. Terminology
    a. Attribute/feature
    b. Target attribute/label
    c. Training/Mining
    d. Testing/Prediction

3. Feature Types
    a. Numeric (Age, Balance)
    b. Categorical (Employed, Yes, No)
    c. Ordinal Categorical (Gold, Silver, Bronze)

4. Classification and class probability estimation
    a. How likely will customer respond to campaign
    b. Diabetes prediction based on medical history
    c. Loan applications

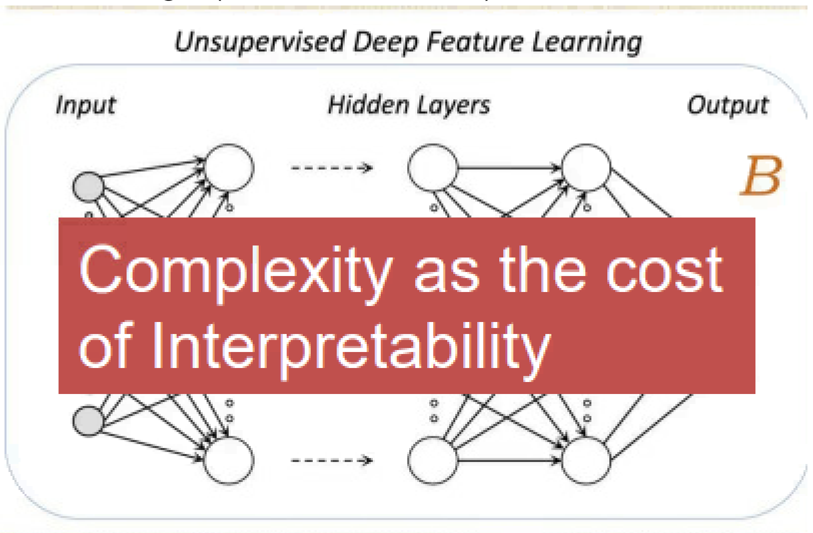
5. Numerical Predictions (**Regression**)
    a. Rate a movie 1-5
    b. Attendance of Mall of America
    c. Customer lifetime value prediction

6. Classification vs Regression
    a. “Will customer purchase services if given incentive?”

        i. Classification (binary target)

   b. "Which service package (A, B, C, None) will customer likely purchase if given incentive?"
      i. Classification (four value target)
   c. "How much will customer use service?"
      i. Regression (numeric target, target is amount of usage for customer)

7. Supervised vs Unsupervised Learning
   a. "Do customers naturally fall into different groups?"
      i. No guarantee results are meaningful or will be useful for any purpose
   b. "Can we find groups of customers who have likelihood of canceling service after contract ends?"
      i. **Specific purpose**
      ii. Meaningful results (usually)
      iii. **Requires data on the target** (the individual's label)


Unsupervised Deep Feature Learning
Input
Hidden Layers
Output
B
Complexity as the cost of Interpretability

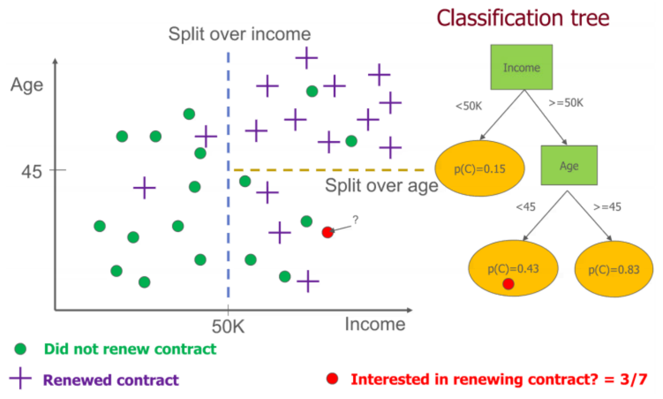
8. Selecting Informative Attributes
   a. Objective: partition customers into subgroups that are **less impure with respect to class**, so that each group has as many instances as possible that belong to the same class
   b. Most common splitting criterion is **information gain** (IG), based on purity measure called *entropy*
   c. Information gain measures change in entropy due to any amount of new information being added

        i. IG(parent, child) = entropy(parent) - [$p(c_1)$ * entropy(c_1) + $p(c_2)$ * entropy(c_2) + …]
        ii. entropy(parent) = - [p(0) * $\log_2$ p(0) + p(1) * $\log_2$ p(1)]
        iii. entropy(child) = - [p(0) * $\log_2$ p(0) + p(1) * $\log_2$ p(1)] (ALL DEPENDS ON THE SPLIT OF DATA)

9. Nodes Split
    a. ***Divide-and-conquer*** approach: take data subset and ***recursively*** find best attribute to partition
    b. When do you stop?
        i. Nodes are pure, no more variables, over-fitting

10. Decision Trees
    a. Easy to understand, implement, use
    b. Computationally cheap
    c. Advantages for model comprehensibility (model evaluation, communication to non savvy stakeholders)

Motivation Example



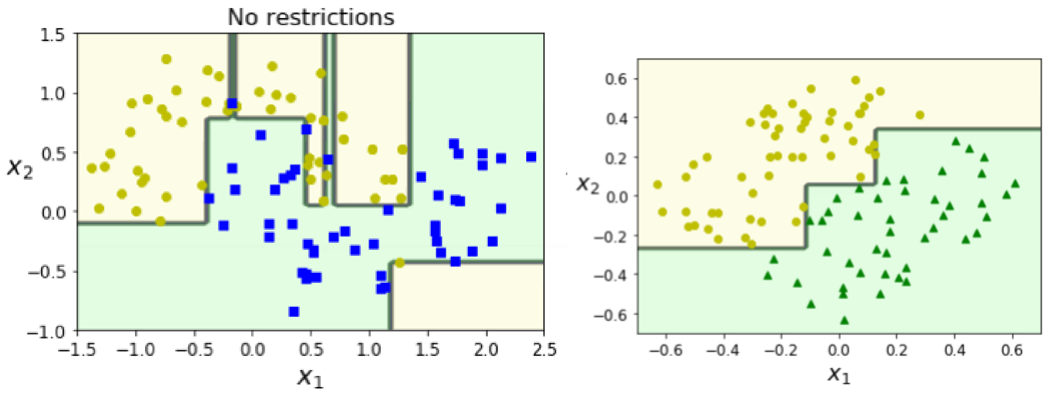
11. Probability Estimation Trees
    a. **Frequency-based estimate**: each member of segment corresponds to tree leaf that has same probability of belonging to that class

b. If leaf contains *n* positive instances and *m* negative instances (binary classification), probability of any new instance being positive is estimated as *n / n +m*

c. Prone to **overfitting**

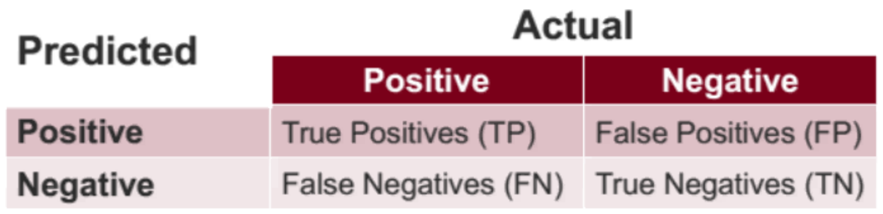
   i. **Laplace Correction** is method for avoid this trap, p(c) = *n* + 1 / *n* + *m* + 2 (*n* belongs to *c*, *m* does not)

12. Overfitting/Instability examples



13. Another classification view
    a. Producing **continuous output** (probability of belonging to certain class)
    b. Classify by applying certain threshold (probability cutoff value) on output
    c. Naive Bayes, DT, NN

14. Confusion Matrix
    a. Problem involving *n* will be an n x n matrix
    b. Columns labeled with actual classes and rows labeled with predicted classes



Predicted	Actual: Positive	Actual: Negative
Positive	True Positives (TP)	False Positives (FP)
Negative	False Negatives (FN)	True Negatives (TN)

c. Accuracy = (TP + TN) / (ALL OBVS)
d. Precision_p = TP / (TP + FP); Precision_n = (TN / TN + FN)
e. Recall_p = TP / (TP + FN);Recall_n = TN / (TN + FP)
f. F-Measure (harmonic mean of P & R) = 2 * (Precision_p * Recall_p / Precision_p + Recall_p)
    i. Always between 0 (BAD) and 1 (YES)

15. Majority Rule (Naive Bayes)
    a. Classify all records that belong to the majority (most prevalent) class
    b. Accuracy if negative is just TN / ALL OBVS
    c. Precision if negative is usually 1, recall is …

16. Multi-class Confusion Matrix
    a. C1, C2, ... , Ck
    b. Predicted(C1) - points predicted to be C1 by model
    c. Actual(C1) - points that belong to C1
    d. PrecisionC1 = | Predicted(C1) union Actual(C1) | / | Predicted(C1) |

Week 2:

Classification, class probability estimation, ranking

- Logistic Regression

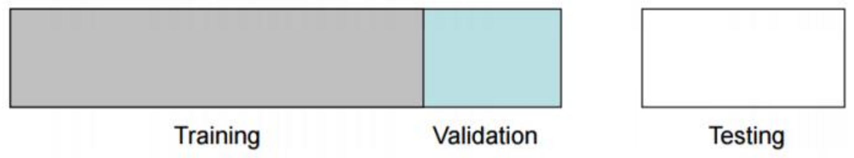
Generalization and issue of overfitting

- Regularization

1. ***k*-NN (*k*-Nearest Neighbor)**
    a. Low values of *k* (1, 3) get local structure of data (but noise as well)
    b. High values of *k* provide smoothing and less noise, but loss some of that local structure
    c. K = | D | data set is naive rule (classifying all records as majority class)

2. **Training-Testing-Validation**
    a. Keep part of labeled data apart as validation data
    b. Evaluate k values based on prediction accuracy of validation data

c. Choose k that minimizes the validation error



Validation can be viewed as another name for testing, but the name ***testing*** is typically reserved for final evaluation purpose, whereas ***validation*** is mostly used for model selection purpose.

3. Normalization Approaches
    a. Scaling/Normalizing standardizes intervals before measuring distance to neighbors
    b. Min-max scaling: numerical attributes get switched to interval between [0, 1]
        i. z = (x - min) / (max - min)

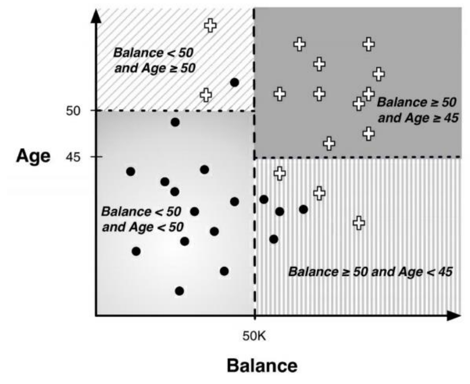
When attributes have different importance

- Weighted distances may be used
- E.g., $d(A,B) = w_1|a_1 - b_1| + \cdots + w_k|a_k - b_k|$

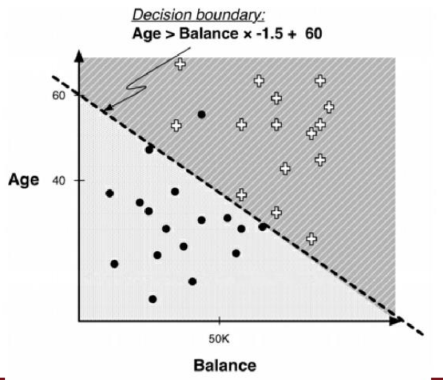
4. Why *k*-NN?
    a. “Lazy” learning approach
        i. No model building -> faster to train but slower to estimate
    b. Enhancement
        i. Weighted distances
    c. Strengths
        i. Easy to implement/use
        ii. Handles noisy data well except for small *k* values
        iii. No assumptions required
        iv. Complex interactions between vars without building models
    d. Weaknesses
        i. More time to perform estimation, computational inefficient
        ii. Requires storage, domain knowledge and blows up dimensionality

5. Logit Reg

Decision Boundaries: Decision Trees



Decision Boundaries: Linear Classifier



6. Logit Reg, contd.
    a. Distinction between classification and regression is value for **target variable is categorical or numeric**
    b. LR produces numeric estimate
    c. Values of target variable in data are categorical
    d. LR is estimating probability of class membership over **categorical class**

7. Classification Process
    a. Model produces estimated probability of being class “1”
        i. Coefficients (B) derived through iterations
    b. Converts to classification by establishing cutoff level
        i. Maximize accuracy, minimize false positives and negatives, minimize costs)
    c. If probability > cutoff, label it “1”

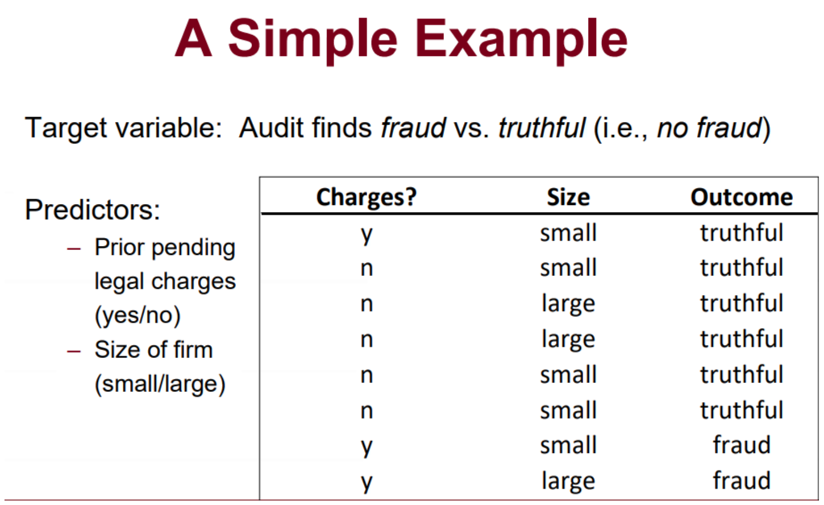
8. Logit vs Tree Induction
    a. Classification trees use decision boundaries that are perpendicular / orthogonal
    b. Linear classifiers can use boundaries of any direction / orientation (by using weighted combination of all attributes)

    c. Linear classifier places single decision surface through entire space
    d. Tree can cut up instance space into small regions

9. Ordered Logit
    a. Multiclass Classification -> Ordinal Categorical Target
        i. Rating systems (poor, fair, good, excellent)
        ii. Employment (unemployed, part time, full time)
        iii. Grades (A, B, C, D, F)

$$y=\begin{cases} 0 & if\ \ y^* \leq threshold_1 \\ 1\ if threshold_1 < y^* \leq threshold_2 \\ 2\ if threshold_2 < y^* \leq threshold_3 \\ \quad\quad\quad\quad \ldots \end{cases}$$

10. Objective Functions
    a. “Best” line depends on **objective (loss) function**, which represents the goal
    b. Loss function determines how much penalty should be assigned to instance based on error in predicted value
    c. **Linear regression**, **logistic regression**, **support vector machines** are similar instances that fit a linear model to data using a **different objective function**
11. Naive Bayes
    a. Handles categorical and numeric data well
    b. Simple and computationally efficient (for large data sets)
    c. Robust to irrelevant attributes (handles noise)
    d. Actual probability estimates may not be accurate, but probability rankings are, so predictions can be accurate

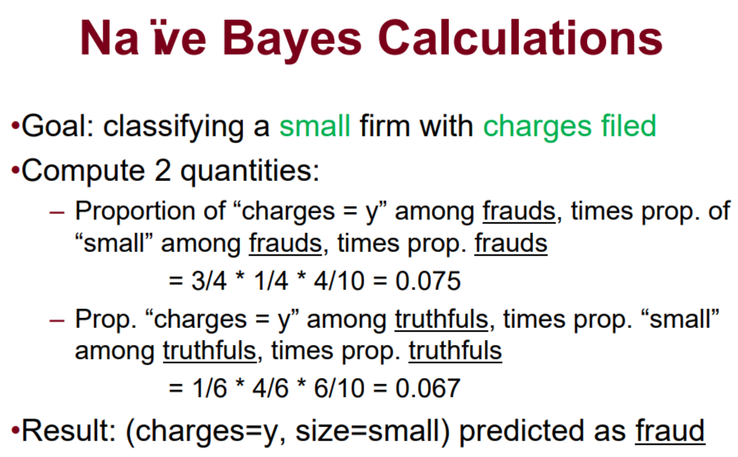


A Simple Example

Target variable: Audit finds *fraud* vs. *truthful* (i.e., *no fraud*)

Predictors:
- Prior pending legal charges (yes/no)
- Size of firm (small/large)

Charges?	Size	Outcome
y	small	truthful
n	small	truthful
n	large	truthful
n	large	truthful
n	small	truthful
n	small	truthful
y	small	fraud
y	large	fraud



Na ïve Bayes Calculations

•Goal: classifying a small firm with charges filed

•Compute 2 quantities:
- Proportion of "charges = y" among frauds, times prop. of "small" among frauds, times prop. frauds

  = 3/4 * 1/4 * 4/10 = 0.075
- Prop. "charges = y" among truthfuls, times prop. "small" among truthfuls, times prop. truthfuls

  = 1/6 * 4/6 * 6/10 = 0.067

•Result: (charges=y, size=small) predicted as fraud

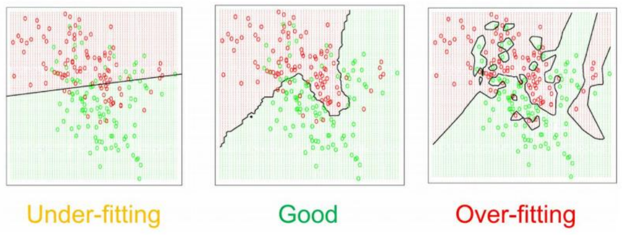
12. Overfitting

a. Tendency of data mining to tailor models to training data *at the expense of generalization* to unseen data points

b. Model should apply to general population from where training data came from

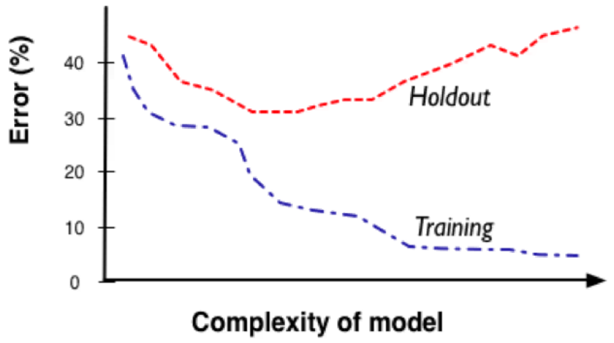
c. All DM have tendency to overfit (some more than others)

d. No single choice to eliminate overfitting, so manage the complexity in *principled* way



Over-fitting in tree induction

Fitting Graph



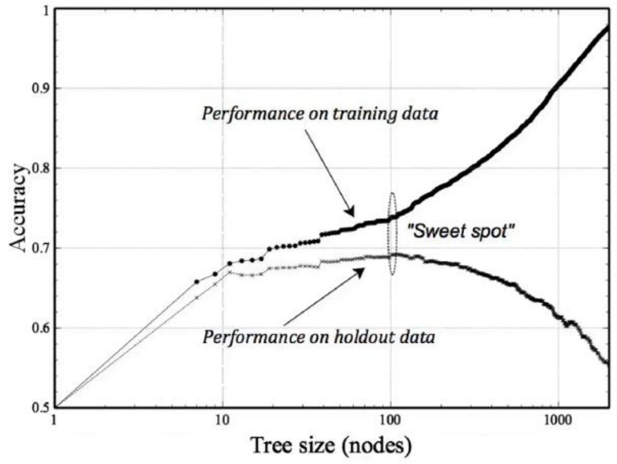


Figure 5-1. A typical fitting graph. Each point on a curve represents an accuracy estimation of a model with a specified complexity (as indicated on the horizontal axis). Accuracy estimates on training data and testing data vary differently based on how complex we allow a model to be. When the model is not allowed to be complex enough, it is not very accurate. As the models get too complex, they look very accurate on the training data, but in fact are overfitting—the training accuracy diverges from the holdout (generalization) accuracy.

13. Learning Curves
    a. Plots the generalization performance against the amount of training data
    b. Performance only on testing data, plotted against *amount of training data* used

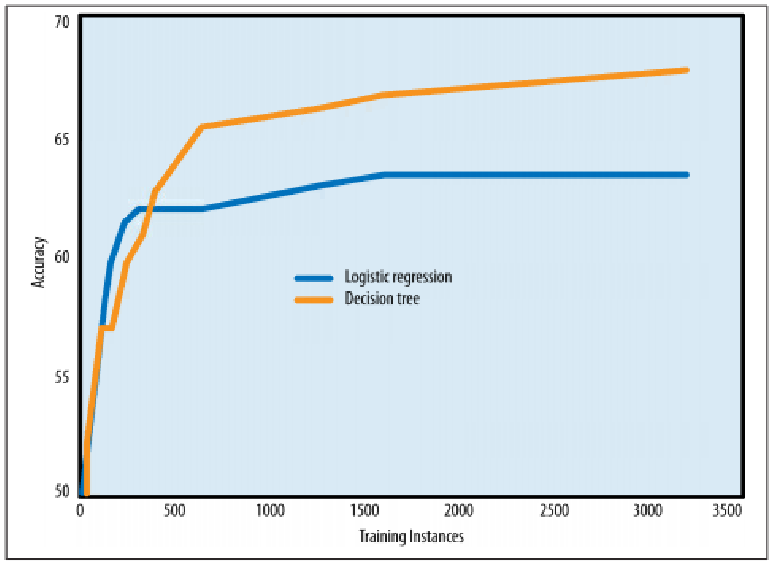
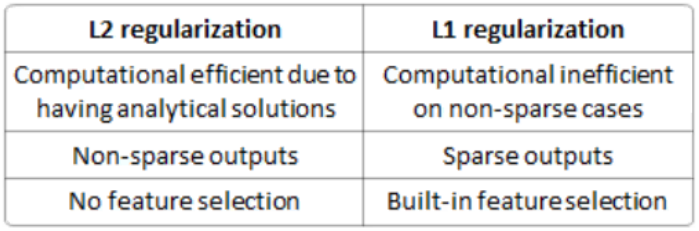


Figure 5-11. Learning curves for tree induction and logistic regression for the churn problem. As the training size grows (x axis), generalization performance (y axis) improves. Importantly, the improvement rates are different for the two induction technique, and change over time. Logistic regression has less flexibility, which allows it to overfit less with small data, but keeps it from modeling the full complexity of the data. Tree induction is much more flexible, leading it to overfit more with small data, but to model more complex regularities with larger training sets.

14. Tree Pruning (technique for Overfitting)
    a. Some branches represent outliers (from noise in data) and results in overfitting
    b. Pruning techniques address this problem
        i. Pre-pruning: halting tree induction early
            1. Threshold on data points in each node
            2. Threshold on number of tree nodes
            3. Threshold on attribute metrics
        ii. Post-pruning
            1. Performed after building entire tree

        2. Calculate expected error rates with/without nodes (using internal training/validation split) and selecting the best choice

15. Regularization
    a. “L2-norm”
        i. Sum of *squares* of weights
        ii. L2-norm + standard least-squares linear regression = **ridge regression**
    b. “L1-norm”
        i. Sum of *absolute values* of weights
        ii. L1-norm + standard least-squares linear regression = **lasso regression**
        iii. Automatic feature selection



L2 regularization	**L1 regularization**
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

16. Regularization, contd.
    a. General approach to control model complexity
        i. Regressions
        ii. SVM
        iii. NN and DNN
        iv. XGBoost

17. Summary:
    a. Fundamental trade-off between model complexity and possibility of overfitting
        i. Complex model may be necessary if phenomenon producing data is complex itself, but complex models may overfit training data
    b. All model types can be overfit, hence the use of fitting graphs and learning curves
    c. Regularization
        i. Techniques: tree pruning, feature selection, employing explicit complexity penalties in objective functions used for modeling

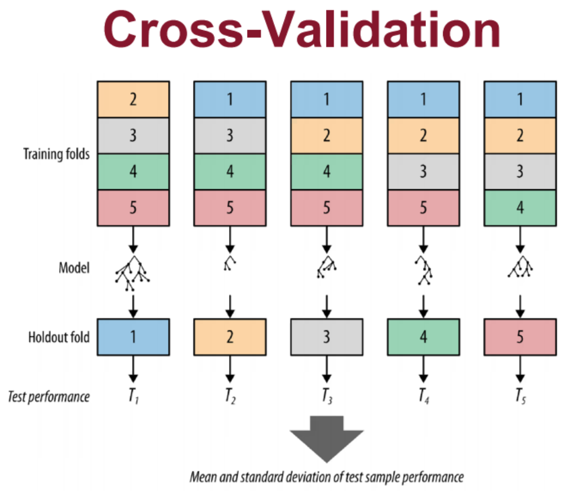
Week 3:

In-depth view of classifier performance and evaluation

- Cross-Validation
- Visualization of Predictive Performance

1. Generalization Performance
    a. Different modeling procedures may have different performance on same data
    b. Different training sets may result in different generalization performance
    c. Different test sets may result in different estimates of generalization performance

2. Cross-Validation
    a. Not only estimate of generalization performance, but **statistics on estimated performance** like the *mean* and *variance*
    b. Better used of limited data set (computing estimates over *all* the data)


Cross-Validation
Training folds
Model
Holdout fold
Test performance
T_1 T_2 T_3 T_4 T_5
Mean and standard deviation of test sample performance

3. Stratified Cross Validation
    a. Proportions of classes in training/test sets to be similar to real data

MegaTelCo: Cross-Validation Example

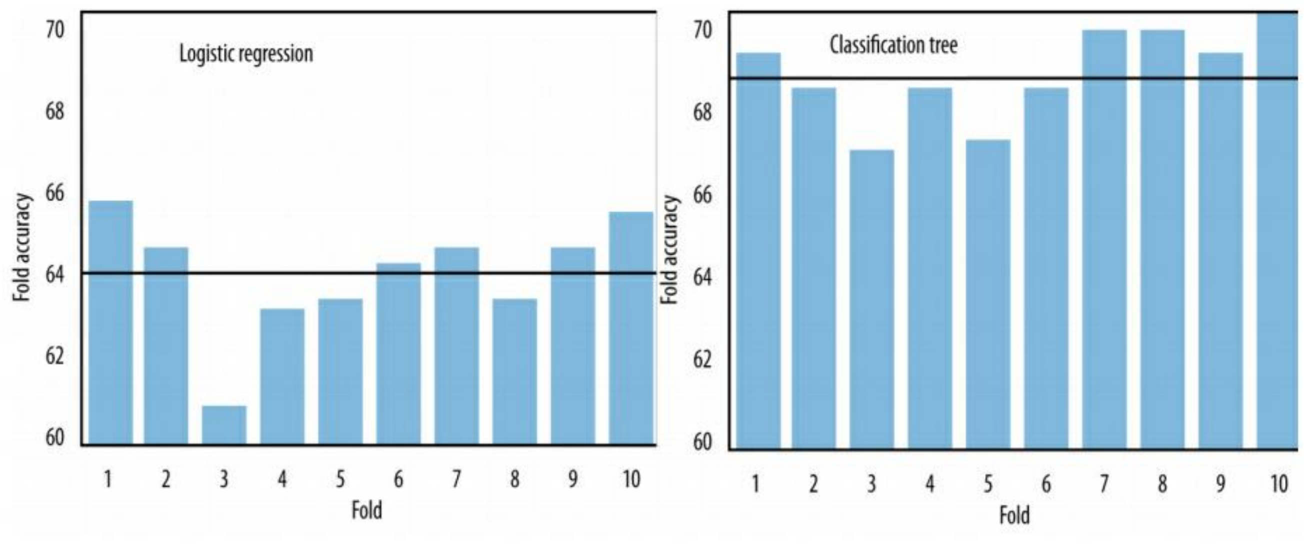
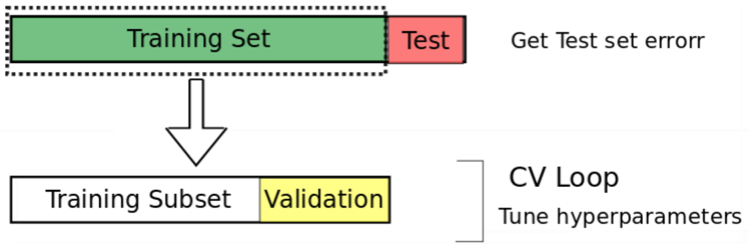


Figure 5-10. Fold accuracies for cross-validation on the churn problem. At the top are accuracies of logistic regression models trained on a dataset of 20,000 instances divided into ten folds. At the bottom are accuracies of classification trees on the same folds. In each graph the horizontal line shows the average accuracy of the folds. (Note the selection of the range of the y axis, which emphasizes the differences in accuracy.)

4. Hyperparameter tuning

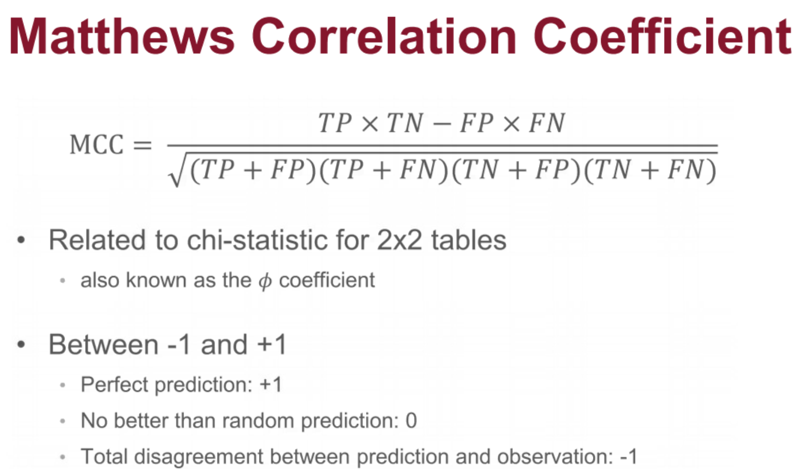
To avoid "wasting" too much training data in validation sets, a common technique is to use *cross-validation*: the training set is split into complementary subsets, and each model is trained against a different combination of these subsets and validated against the remaining parts. Once the model type and hyperparameters have been selected, a final model is trained using these hyperparameters on the full training set, and the generalized error is measured on the test set.

Hyperparameters need to tune



5. Limitations of Accuracy Measure
    a. *Unbalanced* class distributions
        i. Predicting bankrupt companies
            1. 1% will go bankrupt, 99% will not
            2. “Trusting” classifier will predict every company to be good (using majority rule): 99% accuracy
                a. No help avoiding bad companies
            3. “Cautious” classifier predicts every company will be bad: 1% accuracy
                a. Never invest in a good company
        ii. Per-class evaluation should be used instead of accuracy in these cases (Precision, Recall, F-score)
    b. *Unequal costs and benefits* for different classes
        i. Accuracy makes no distinction between false positive and false negative errors
        ii. Need to explicitly consider costs of decisions that classifiers make

6. Matthews Correlation Coefficient



Matthews Correlation Coefficient

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

- Related to chi-statistic for 2x2 tables
  - also known as the ϕ coefficient
- Between -1 and +1
  - Perfect prediction: +1
  - No better than random prediction: 0
  - Total disagreement between prediction and observation: -1

7. **ROC Curves**
   a. ROC curves are independent of class proportions as well as costs and benefits
   b. Data scientist can plot performance of classifiers on ROC graph as they get generated, knowing that positions and relative performance of classifiers will not change
   c. Not most intuitive visualization for business stakeholders

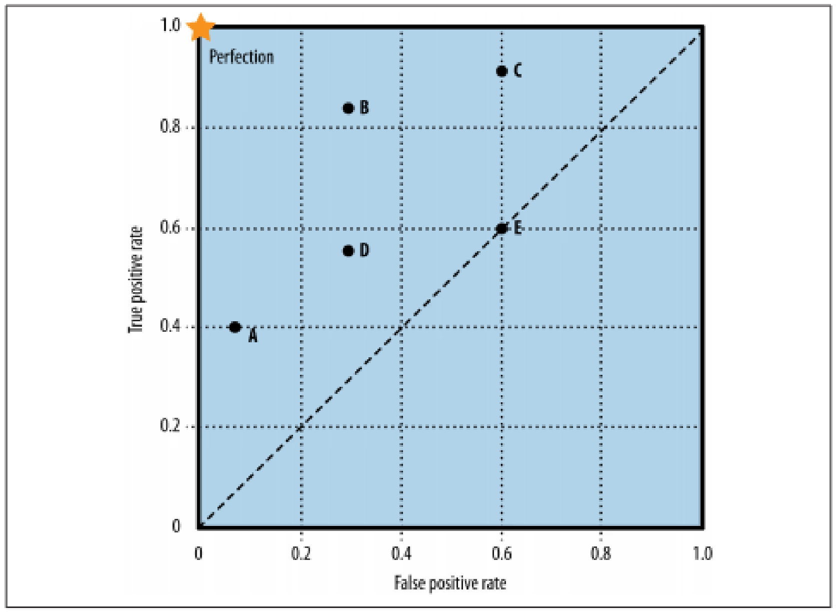


Figure 8-3. ROC space and five different classifiers (A-E) with their performance shown.

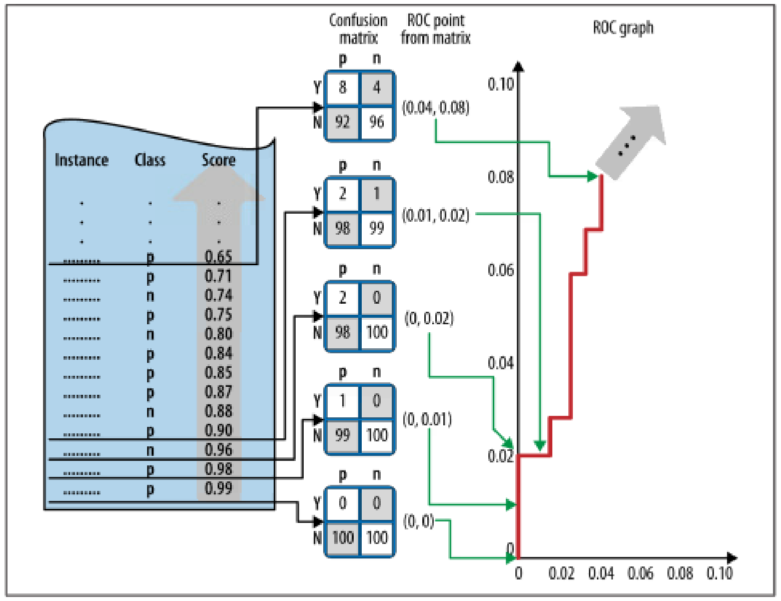


Figure 8-5. An illustration of how a ROC "curve" (really, a stepwise graph) is constructed from a test set. The example set, at left, consists of 100 positives and 100 negatives. The model assigns a score to each instance and the instances are ordered decreasing from bottom to top. To construct the curve, start at the bottom with an initial confusion matrix where everything is classified as **N**. *Moving upward, every instance moves a count of 1 from the* **N** *row to the* **Y** *row, resulting in a new confusion matrix. Each confusion matrix maps to a (fp rate, tp rate) pair in ROC space.*

8. **Generating ROC Curve**
   a. Sort by model predictions
   b. Start with max(prediction) as cutoff
   c. Decrease cutoff, each step count number of TP (positive with prediction above cutoff) and FP (negative above cutoff)
   d. Calculate TP rate (TP/P) and FP (FP/N) rate
   e. Plot current number of TP/P as function of FP/N

9. Area Under the ROC Curve (AUC)
   a. Area under classifiers' curve expressed as fraction (0 to 1)
   b. AUC useful when single number is needed for summarizing performance

c. Equivalent to **Mann-Whitney-Wilcoxon** measure

d. Probability that randomly chosen positive instance will be ranked ahead of randomly chosen negative instance

10. Cumulative Response curves and Lift Curves

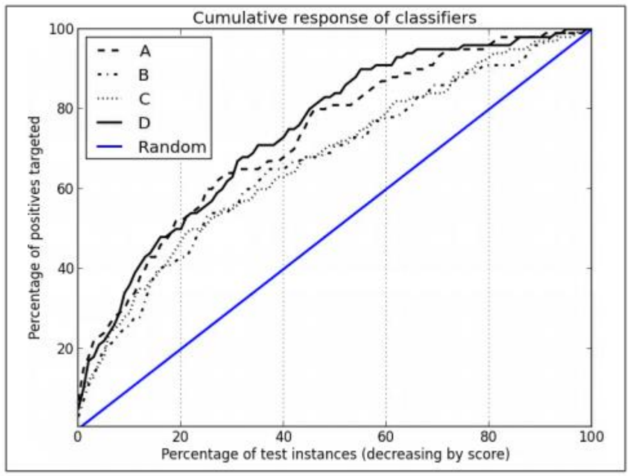


Figure 8-6. Four example classifiers (A–D) and their cumulative response curves.

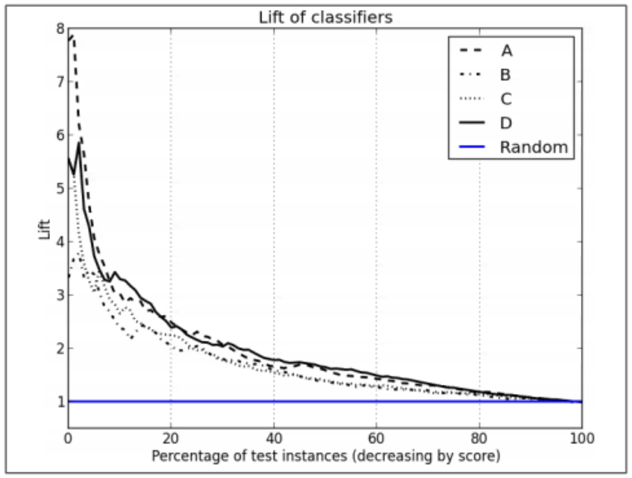
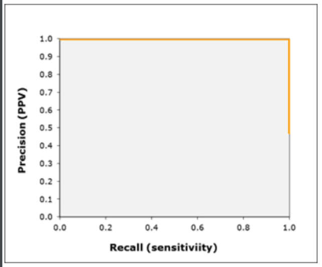
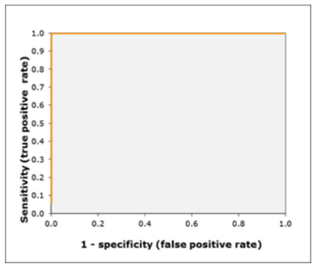


Figure 8-7. The four classifiers (A–D) of Figure 8-6 and their lift curves.

Best and Worse Cases



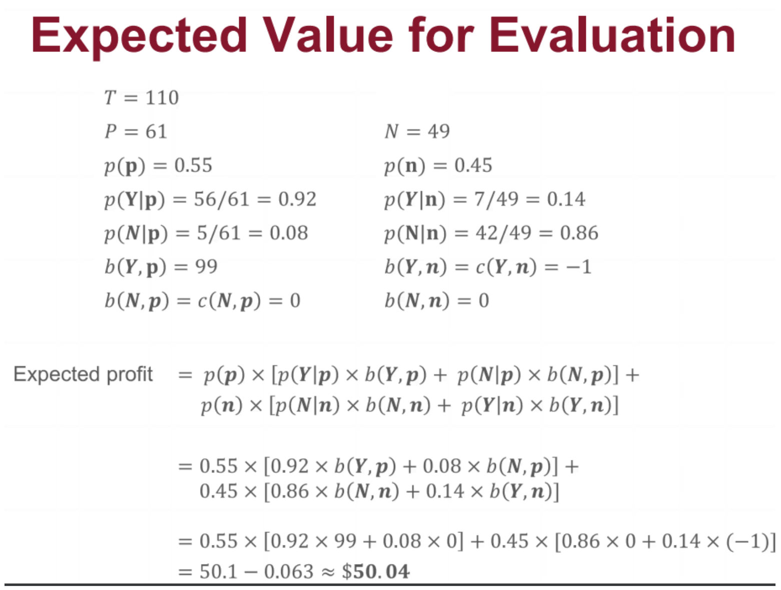


11. Best/Worse Case

12. Expected Value

a. Expected value computation provides framework to organize thinking about data-analytic problems

b. Transforms thinking into
   i. Structure of problem
   ii. Elements of analysis that can be derived from data
   iii. Elements of analysis that need to be derived from outside sources
   iv. EV = $p(o_1) * v(o_1) + p(o_2) * v(o_2) + p(o_3) * v(o_3)$



Expected Value for Evaluation

$T = 110$	
$P = 61$	$N = 49$
$p(\mathbf{p}) = 0.55$	$p(\mathbf{n}) = 0.45$
$p(\mathbf{Y}\mid\mathbf{p}) = 56/61 = 0.92$	$p(\mathbf{Y}\mid\mathbf{n}) = 7/49 = 0.14$
$p(\mathbf{N}\mid\mathbf{p}) = 5/61 = 0.08$	$p(\mathbf{N}\mid\mathbf{n}) = 42/49 = 0.86$
$b(\mathbf{Y},\mathbf{p}) = 99$	$b(\mathbf{Y},\mathbf{n}) = c(\mathbf{Y},\mathbf{n}) = -1$
$b(\mathbf{N},\mathbf{p}) = c(\mathbf{N},\mathbf{p}) = 0$	$b(\mathbf{N},\mathbf{n}) = 0$

Expected profit $= p(\mathbf{p}) \times [p(\mathbf{Y}\mid\mathbf{p}) \times b(\mathbf{Y},\mathbf{p}) + p(\mathbf{N}\mid\mathbf{p}) \times b(\mathbf{N},\mathbf{p})] + p(\mathbf{n}) \times [p(\mathbf{N}\mid\mathbf{n}) \times b(\mathbf{N},\mathbf{n}) + p(\mathbf{Y}\mid\mathbf{n}) \times b(\mathbf{Y},\mathbf{n})]$

$= 0.55 \times [0.92 \times b(\mathbf{Y},\mathbf{p}) + 0.08 \times b(\mathbf{N},\mathbf{p})] + 0.45 \times [0.86 \times b(\mathbf{N},\mathbf{n}) + 0.14 \times b(\mathbf{Y},\mathbf{n})]$

$= 0.55 \times [0.92 \times 99 + 0.08 \times 0] + 0.45 \times [0.86 \times 0 + 0.14 \times (-1)]$
$= 50.1 - 0.063 \approx \$\mathbf{50.04}$

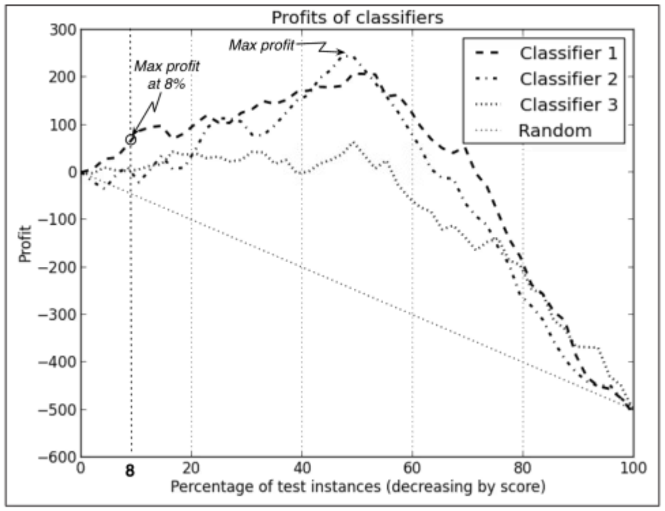
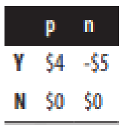


Figure 8-2. Profit curves of three classifiers. Each curve shows the expected cumulative profit for that classifier as progressively larger proportions of the consumer base are targeted.



	p	n
Y	\$4	-\$5
N	\$0	\$0

Week 4:
Fundamentals of numeric prediction

- Techniques: Linear regression, k-NN, regression trees

1. Classification vs Regression

   a. Predicting numeric outcome var instead of categorical var
   b. Some techniques naturally get extended to numeric prediction
      i. DT
      ii. k-NN
      iii. SVM
      iv. NN

2. Measuring Performance
   a. Key component of most measures is **difference** between *actual outcome* and *predicted outcome* or **likelihood**

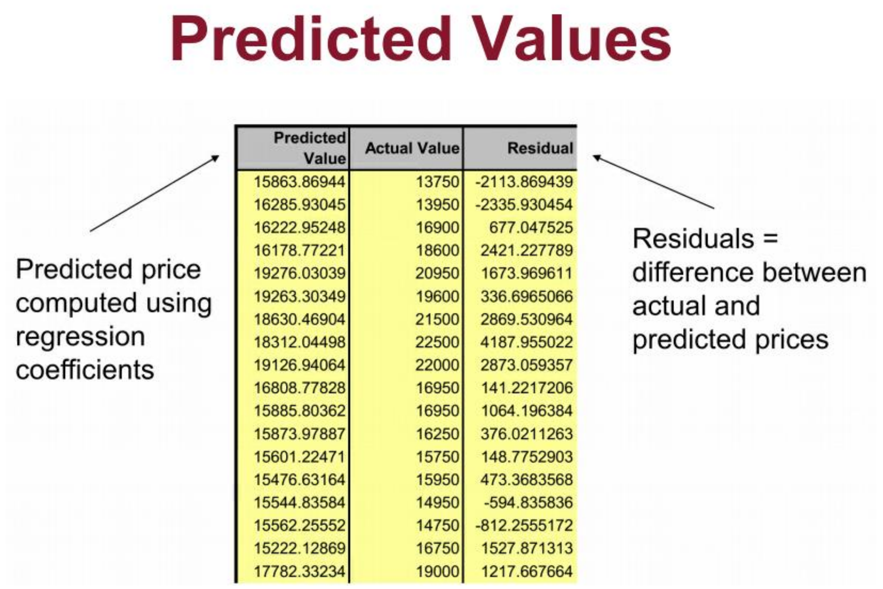
3. **Measures of Error (program these)**
   a. Mean Absolute Error (**MAE**): | y_1 - x_1 | + | y_2 - x_2 | + | y_k - x_k | / *k* (K is the number of predictions)
   b. Mean Absolute Percentage Error (**MAPE**): | y_1 - x_1 / x_1 | + | y_2 - x_2 / x_2 | + | y_k - x_k / x_k |
   c. Root Mean Squared Error (**RMSE**): [(y_1 - x_1) ^ 2 + (y_2 - x_2) ^ 2 + (y_k - x_k) ^ 2] ^ .5
   d. Sum of Squared Errors (SSE): want this to be small
   e. Average Error: (y_1 - x_1) + (y_2 - x_2) + (y_k - x_k) / *k* (K is number of preds)

4. Comparisons
   a. MAPE involves percentage error, use this when this is important for your goal
   b. RMSE vs MAE
      i. RMSE penalizes very large errors
      ii. RMSE may favor model that occasionally makes small errors over model that makes large errors

5. **k-NN** for Numeric Prediction
   a. Instead of using majority vote to determine class label, uses **average** of nearest neighbor outcome values
      i. Weighted average with weight decreasing with distance

6. Regression Trees for Numeric Prediction
   a. Used with continous outcome variable
   b. Recursive partitioning, pruning
   c. Splits are choosen based on minimized impurity
   d. Same advantages/limitations of classification

        i. Easy to intrepret, built-in variable selection
        ii. May not caputre more complex interactions between several vars or structure of data that cannot be split orthogonal
    e. **Differences**:
        i. Prediction is computed as **average** of numerical target variable (in classification is *majority vote*)
        ii. Impurity measured by **sum of squared deviations** from leaf mean
        iii. Performance measured not by classification accuracy (precision, recall) but by RMSE, etc.



Predicted Values

Predicted Value	Actual Value	Residual
15863.86944	13750	-2113.869439
16285.93045	13950	-2335.930454
16222.95248	16900	677.047525
16178.77221	18600	2421.227789
19276.03039	20950	1673.969611
19263.30349	19600	336.6965066
18630.46904	21500	2869.530964
18312.04498	22500	4187.955022
19126.94064	22000	2873.059357
16808.77828	16950	141.2217206
15885.80362	16950	1064.196384
15873.97887	16250	376.0211263
15601.22471	15750	148.7752903
15476.63164	15950	473.3683568
15544.83584	14950	-594.835836
15562.25552	14750	-812.2555172
15222.12869	16750	1527.871313
17782.33234	19000	1217.667664

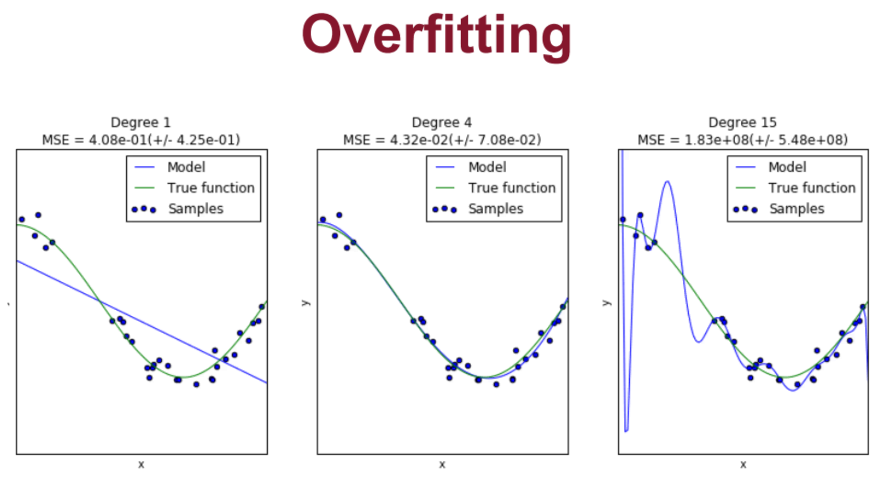
Predicted price computed using regression coefficients

Residuals = difference between actual and predicted prices

7. Linear Regression for Predictive
    a. Goal: predict target values in other data where predictor values, but not target values
        i. Optimize predictive accuracy
        ii. Train model on training data
        iii. ASsess performance on test (hold-out) data

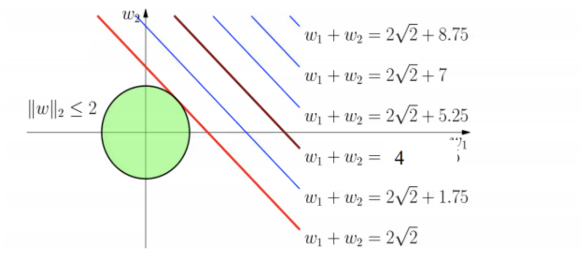
            iv. Evaluation: RMSE, MAE
            v. Explaining role of predictors is not primary purpose but can be useful
    b. Popular for explanatory modeling and prediction
    c. Good predictive model has high predictive accuracy
    d. Removing redundant predictors is key to achieving predictive accuracy and robustness

8. Polynomial



9. Elastic Net
    a. Middle ground between Ridge Regression (L1) and Lasso Regression (L2)
        i. Lasso can behave erratically
            1. # of features > # of training instances
            2. Features are highly correlated -> high variance
    b. Repeated Feature Case

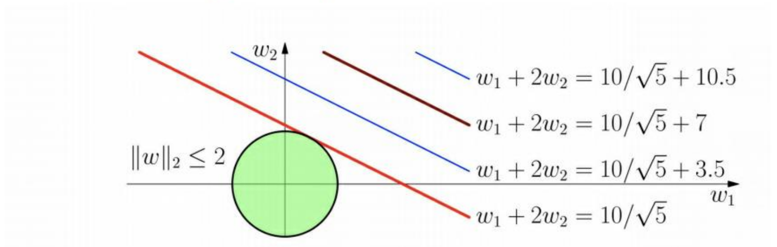
Repeated Features Case


w_2
$\|w\|_2 \le 2$
$w_1 + w_2 = 2\sqrt{2} + 8.75$
$w_1 + w_2 = 2\sqrt{2} + 7$
$w_1 + w_2 = 2\sqrt{2} + 5.25$
$w_1 + w_2 = 4$
w_1
$w_1 + w_2 = 2\sqrt{2} + 1.75$
$w_1 + w_2 = 2\sqrt{2}$

- Error increase as we move away from these parameter settings
- Intersection of w1+w2 = 2 and the norm ball is ridge solution.
- And w1=w2

c. **Linearly Dependent Features Case**

Linearly Dependent Case

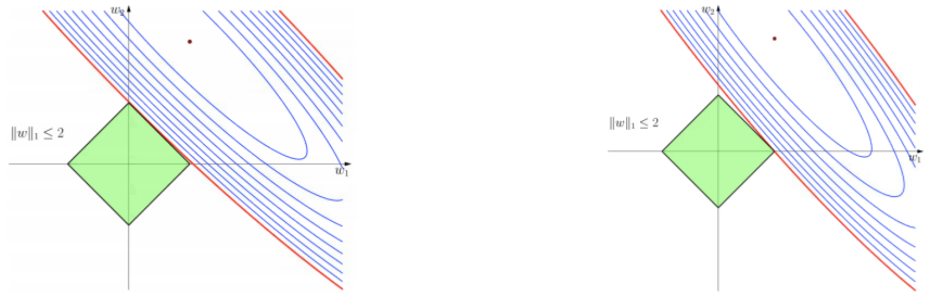

w_2
$\|w\|_2 \le 2$
$w_1 + 2w_2 = 10/\sqrt{5} + 10.5$
$w_1 + 2w_2 = 10/\sqrt{5} + 7$
$w_1 + 2w_2 = 10/\sqrt{5} + 3.5$
w_1
$w_1 + 2w_2 = 10/\sqrt{5}$

- $w_1 + 2w_2 = 10/\sqrt{5} + 7$ corresponds to the empirical risk minimizers.
- Intersection of $w_1 + 2w_2 = 10\sqrt{5}$ and the norm ball $\|w\|_2 \leqslant 2$ is ridge solution.
- At solution, $w_2 = 2w_1$.

d. **Correlated Features Case**

   i. X1 and x2 are highly correlated on same scale (highly typical after normalization)
   ii. Higher the correlation, closer to deneration we get, until we get almost two parallel lines

Correlated Features Case

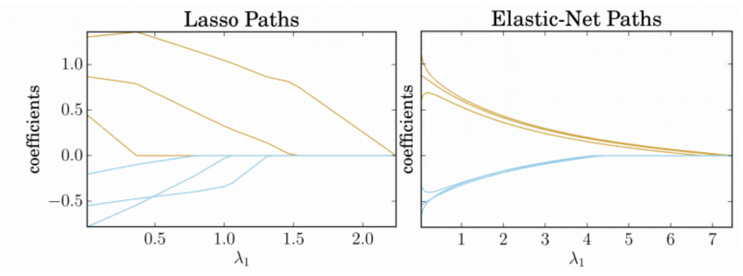


e. **Elastic Net** to the rescue!

Example

$f(x) = w1x1 + x2x2 + w3x3 + w4x4 + w5x5 + w6x6$

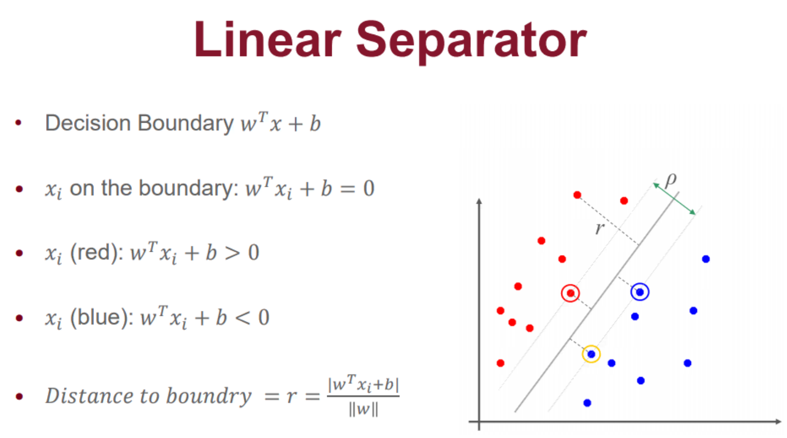
- $x1, x2, x3$ are highly correlated
- $x4, x5, x6$ are highly correlated



Week 5:
Support Vector Machines

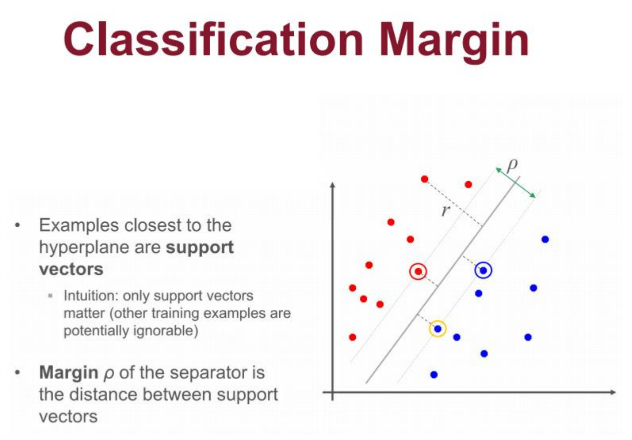
- Classification
- Regression

1. Linear Separator



Linear Separator

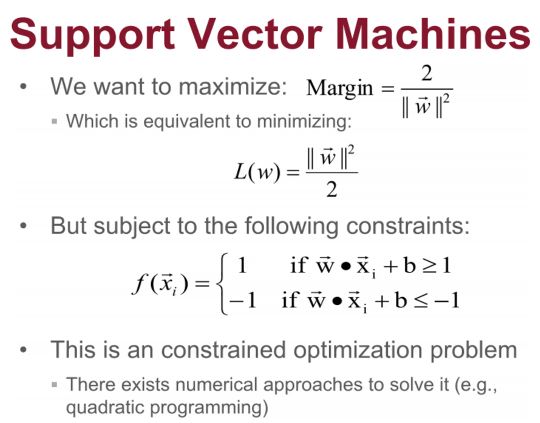
- Decision Boundary $w^T x + b$
- x_i on the boundary: $w^T x_i + b = 0$
- x_i (red): $w^T x_i + b > 0$
- x_i (blue): $w^T x_i + b < 0$
- $Distance\ to\ boundry\ = r = \frac{|w^T x_i + b|}{\|w\|}$



Classification Margin

- Examples closest to the hyperplane are **support vectors**
  - Intuition: only support vectors matter (other training examples are potentially ignorable)
- **Margin** ρ of the separator is the distance between support vectors

2. Support Vector Machines (optimization problem)



Support Vector Machines

- We want to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing:

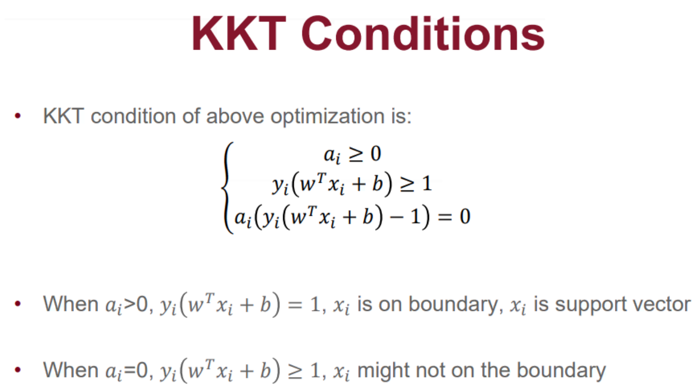
$$L(w) = \frac{\|\vec{w}\|^2}{2}$$

- But subject to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

- This is an constrained optimization problem
  - There exists numerical approaches to solve it (e.g., quadratic programming)

3. KKT Conditions



KKT Conditions

- KKT condition of above optimization is:

$$\begin{cases} a_i \geq 0 \\ y_i(w^T x_i + b) \geq 1 \\ a_i(y_i(w^T x_i + b) - 1) = 0 \end{cases}$$

- When a_i>0, $y_i(w^T x_i + b) = 1$, x_i is on boundary, x_i is support vector
- When a_i=0, $y_i(w^T x_i + b) \geq 1$, x_i might not on the boundary

4. **Why use Dual?**
    a. Primal problem: solvable (quadratic optimization problems)
    b. Dual problem
        i. Only few support vectors matter
        ii. Inner join with few support vectors -> Fast
        iii. Easy to transfer to non-linear SVMs

5. **SVMs explained using the BOOK (WAY MORE INTUITIVE)**
    a. SVMs are linear discriminants
    b. SVMs choose based on simple elegenant idea: instead of separating with a line, first fit the fattest bar betwen the classes
    c. The objective function incorporates the idea that wider bar = better
    d. Once widest bar is found, linear discriminant will center line through the bar, and distance between dashed parallel lines is called *margin*, and objective is to maximize margin

6. **SVMs explanation, continued**
    a. Maximizing margin is intuitively satisfying because training data is sample from population; in predictive modeling we want to predict instances we have not seen yet, and they will be scattered about
    b. Some points will be positive and will land closer to discriminant boundary than any we have seen, likewise with negative points

c. Margin maximizing boundary gives maximal leeway for classifying this marginal points

7. **SVMs, last call**
    a. Second important idea of SVMs lies in how they handle points falling on the wrong side of the discrimination boundary. some data points will inevitably be misclassified by the model.
    b. There may be no such perfect separating line!
    c. Once again, the support-vector machine's solution is intuitively satisfying; the objective function measures how well a particular model fits the training points and we simply penalize a training point for being on the wrong side of the decision boundary.
        i. In cases where data are linearly separable, we incur no penalty and simply maximize the margin.
        ii. If the data are not linearly separable, the best fit is some balance between a fat margin and a low total error penalty and penalty for a misclassified point is proportional to the distance from the decision boundary,
        iii. If possible the SVM will make only "small" errors (known as *hinge loss*)

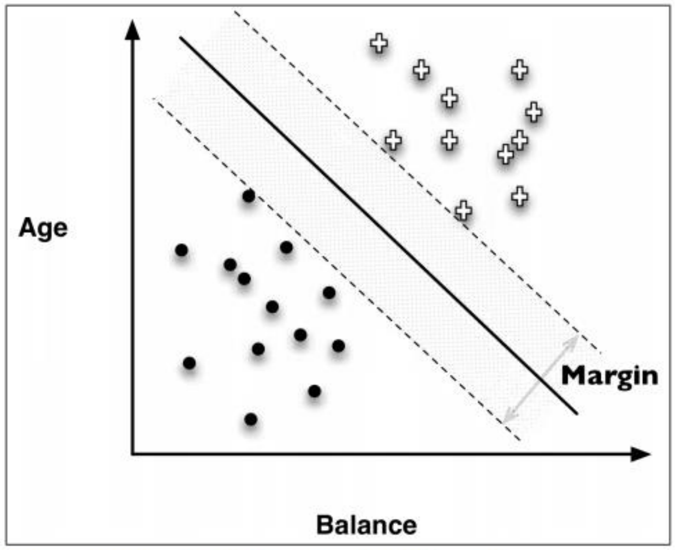


Figure 4-8. The points of Figure 4-2 and the maximal margin classifier.

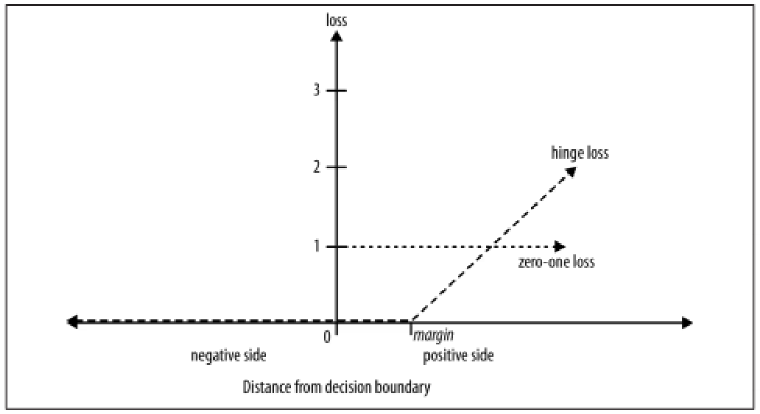


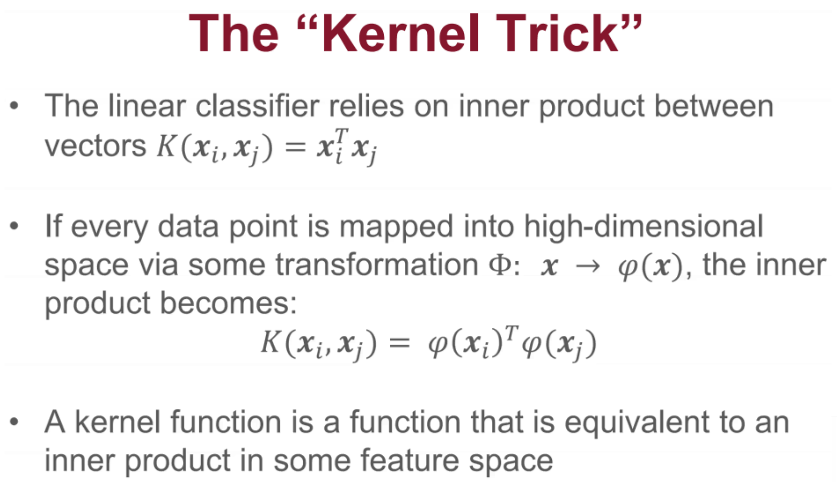
Figure 4-9. Two loss functions illustrated. The x axis shows the distance from the decision boundary. The y axis shows the loss incurred by a negative instance as a function of its distance from the decision boundary. (The case of a positive instance is symmetric.) If the negative instance is on the negative side of the boundary, there is no loss. If it is on the positive (wrong) side of the boundary, the different loss functions penalize it differently. (See "Sidebar: Loss functions" on page 94.)

8. Not Linearly Separable
    a. Soft Margin Classification

b. Non-Linear SVMs

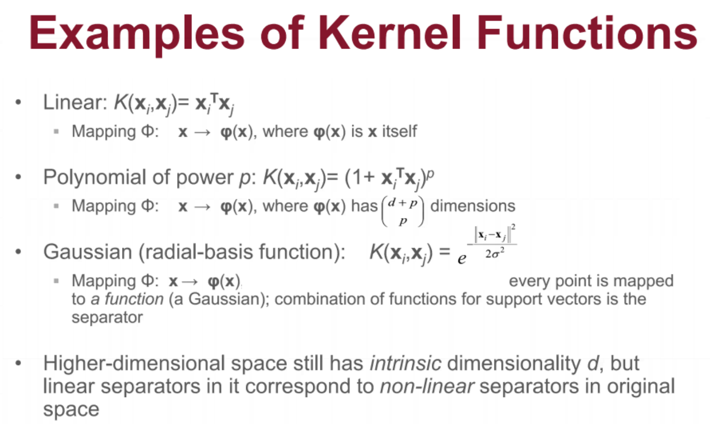
9. Soft Margin
   a. Add slack variables to allow misclassification of difficult/noisy example, results in margin that is *soft*

10. Kernel Trick



The “Kernel Trick”

- The linear classifier relies on inner product between vectors $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$, the inner product becomes:
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$$
- A kernel function is a function that is equivalent to an inner product in some feature space

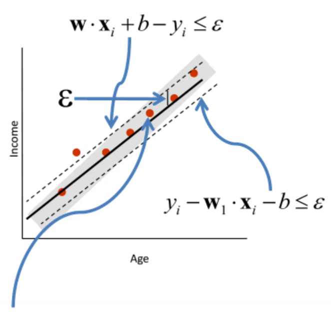


Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself
- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions
- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$ every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator
- Higher-dimensional space still has *intrinsic* dimensionality d, but linear separators in it correspond to *non-linear* separators in original space

11. SVM Regression
   a. Borrows idea from SVM to get hyperplane, find nodes around hyperplane, and margin around hyperplane
   b. Differences:
      i. Prediction error
      ii. Outside margin
         1. Classification: more nodes = better
         2. Regression: less nodes = better
   c. Minimize error, part of error is tolerated

SVM Regression: Another Look


$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon$
ε
Income
$y_i - \mathbf{w}_1\cdot\mathbf{x}_i - b \le \varepsilon$
Age

We do not care about errors as long as they are less than ε

Find a function, $f(x)$, with at most deviation from the target y

$$\min \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t.\ y_i - \mathbf{w}_1\cdot\mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1\cdot\mathbf{x}_i + b - y_i \le \varepsilon;$$

Hint: Compare this with regression regularization problem

What if the problem is not feasible?

SVM Regression

- Introduce slake variable: Soft margin loss

Given training data

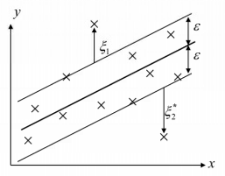
$(\mathbf{x}_i, y_i) \quad i = 1,\dots,m$

Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

Under constraints

$$\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1,\dots,m \end{cases}$$



12. SVM Regression vs Linear Regression
    a. Error
        i. Linear Reg: difference of prediction with real value
        ii. SVM: not error within margin
    b. Data
        i. LR: All training data
        ii. SVM: only support vectors
    c. High Dim
        i. LR: Polynomial Regression
        ii. SVM: Kernel Trick

13. Summary
    a. Typically uses nonlinear mapping to transform original training data into higher dim space
    b. With new dims, searches for linear optimal separating **hyperplane** (decision boundary)
    c. With appropriate nonlinear mapping to high dim, data from two classes can always be separate by hyperplane
    d. SVM finds hyperplane using **support vectors** (essential training tuples) and **margins** (defined by support vectors)