Overfitting/Underfitting in Training

Dr. Yi Long (Neal)

Most contents (text or images) of course slides are from the following textbook
Provost, Foster, and Tom Fawcett. Data Science for Business: What you need to know about data mining and data-analytic thinking. " O'Reilly Media, Inc.", 2013

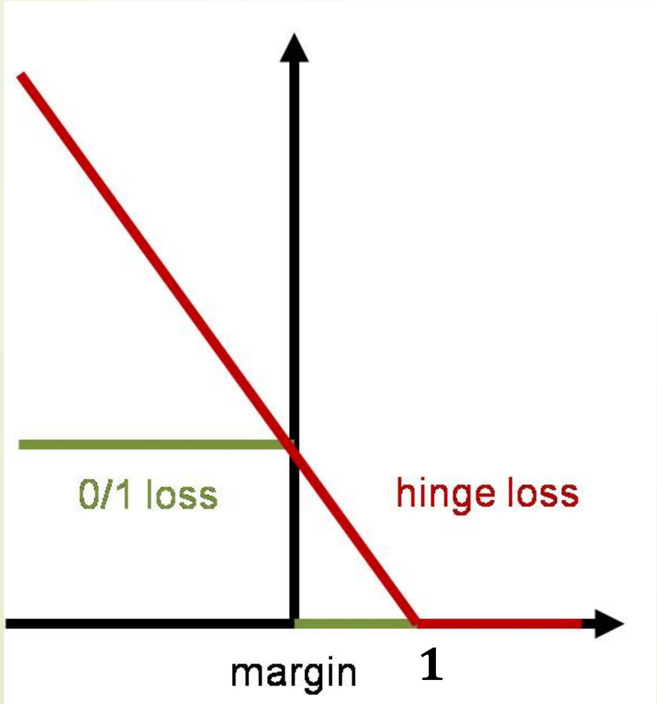
Outline

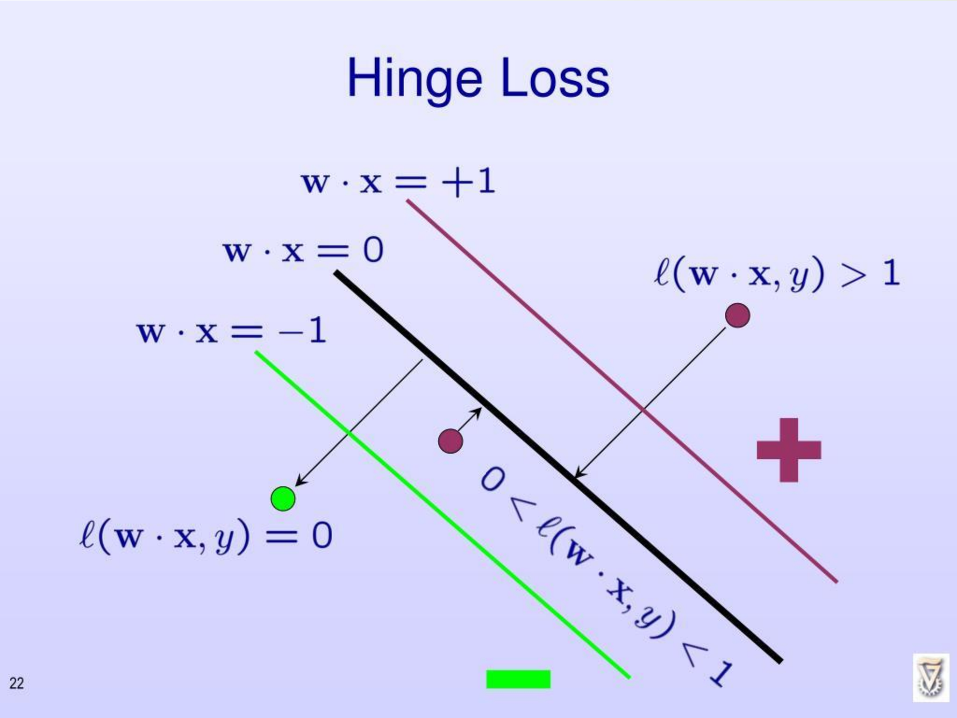
- Hinge Loss & SVM
- Nonlinear Models & Other Extensions
- Overfitting & Holdout Evaluation
- Quiz

Margin Defined by Hinge Loss

- Hinge Loss defines two hyperplanes are parallel to the decision boundary, and a margin between the two parallel hyperplanes

$$hinge\_loss(f, (\mathbf{x}, y)) = \max(0, 1 - yf(\mathbf{x}))$$

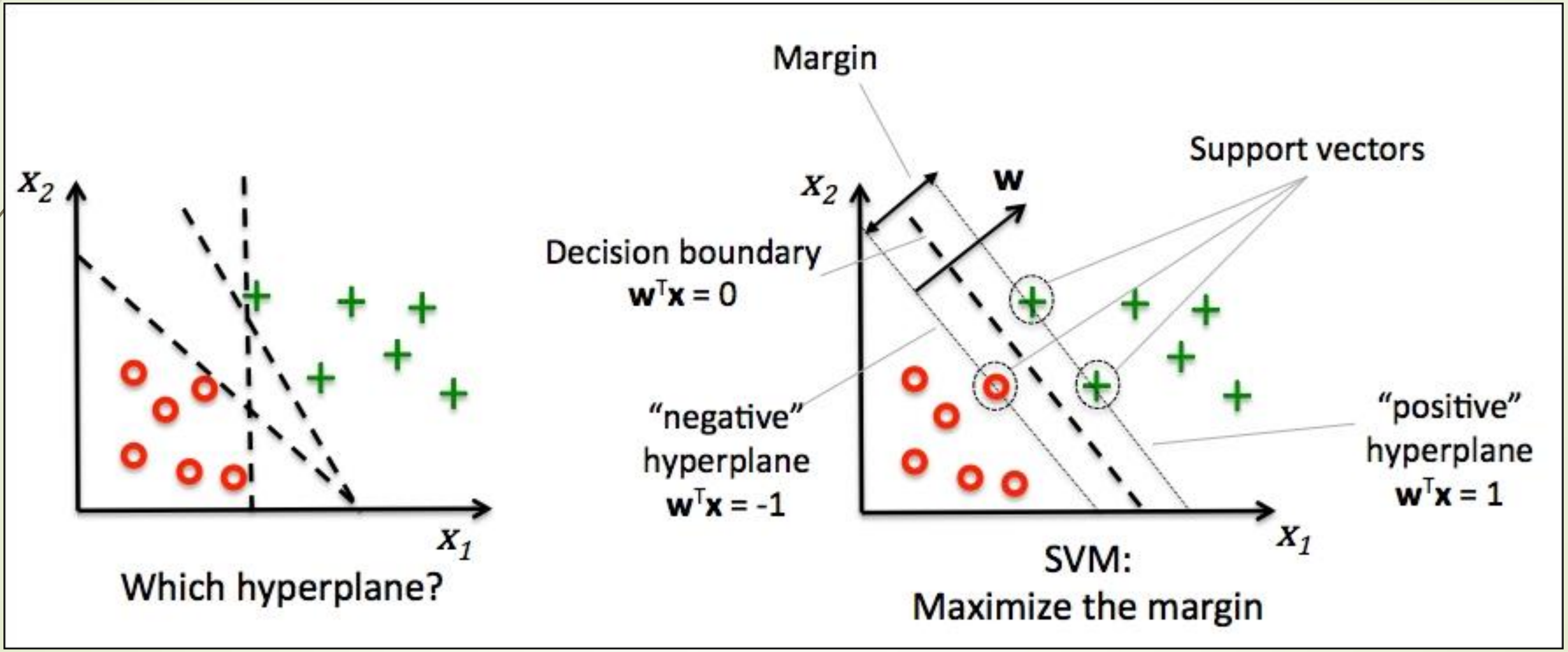




Source: Second Order Learning. Koby Crammer Department of Electrical Engineering. ECML PKDD 2013

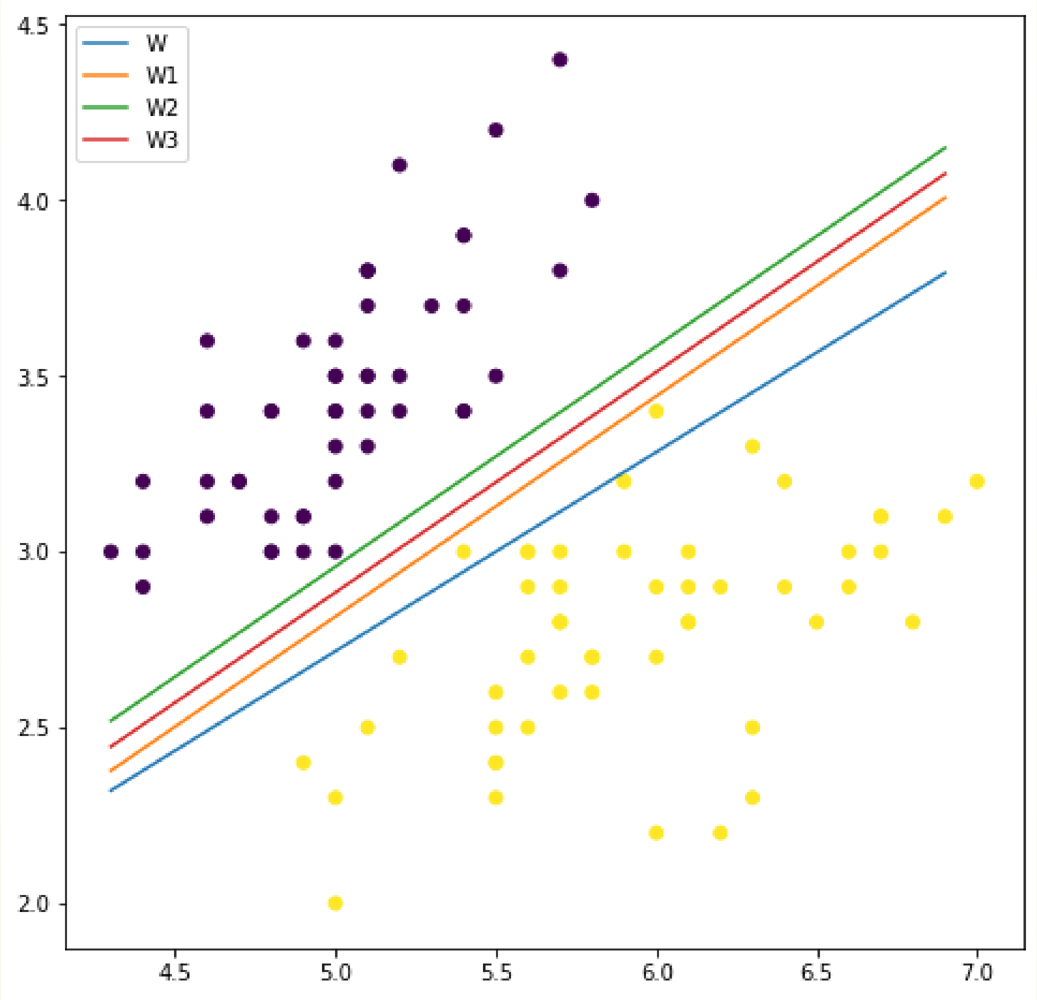
Hinge Loss and Clean Margin

- By minimizing hinge loss is actually trying to find decision boundary with a relative clean margin



Source: Python Machine Learning By Sebastian Raschka

Revisit of Quiz-6 Q2



Distance between Points/Lines

- Distance from point to straight line
  - https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

$$\operatorname{distance}(ax+by+c=0,(x_0,y_0)) = \frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}.$$

- Distance between straight lines
  - https://en.wikipedia.org/wiki/Distance_between_two_straight_lines

When the lines are given by

$$ax+by+c_1=0$$
$$ax+by+c_2=0,$$

the distance between them can be expressed as

$$d = \frac{|c_2-c_1|}{\sqrt{a^2+b^2}}.$$

- Distance from point to hyperplane

$$f(\mathbf{x}) = w_0 + w_1 * x_1 + w_2 * x_2 + \ldots = 0$$
$$f(\mathbf{x}) = w_0 + \mathbf{w}^T \cdot \mathbf{x} = 0 = 0$$

$$distance(f(\mathbf{x})=0, \mathbf{x}_0) = \frac{f(\mathbf{x}_0)}{\|\mathbf{w}\|_2}$$

- **Distance between hyperplanes**

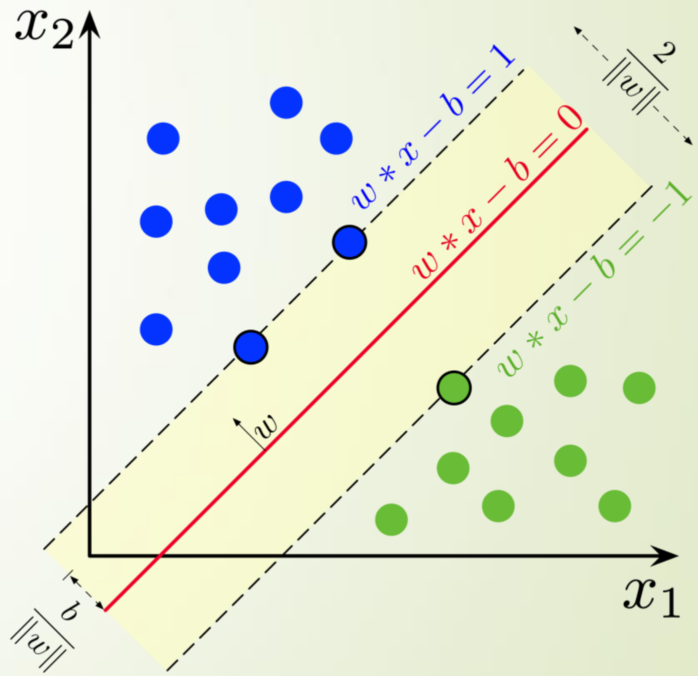
$$f_1(\mathbf{x}) = c_1 + \mathbf{w}^T \cdot \mathbf{x} = 0$$
$$f_2(\mathbf{x}) = c_2 + \mathbf{w}^T \cdot \mathbf{x} = 0$$

$$distance(f_1(\mathbf{x})=0, f_2(\mathbf{x})=0) = \frac{|c_2-c_1|}{\|\mathbf{w}\|_2}$$

Intuition of Support Vector Machine (SVM)

- SVM tries to find some trade-off between **a fat margin** and **a low total loss**
  - ✓ A fat margin is a penalty for complexity (restrict the searching of ***w***)
- Fat margin bar = Bar with large margin width between two parallel boundaries
  - ✓ Margin width between boundaries is $\frac{2}{\|\mathbf{w}\|_2}$
  - ✓ Can maximize margin by minimizing $\|\mathbf{w}\|_2$
  - ✓ Therefore, SVM tries to optimize

$$\min_{\mathbf{w}} HingeLoss(\mathbf{w}) + C \cdot \|\mathbf{w}\|_2$$

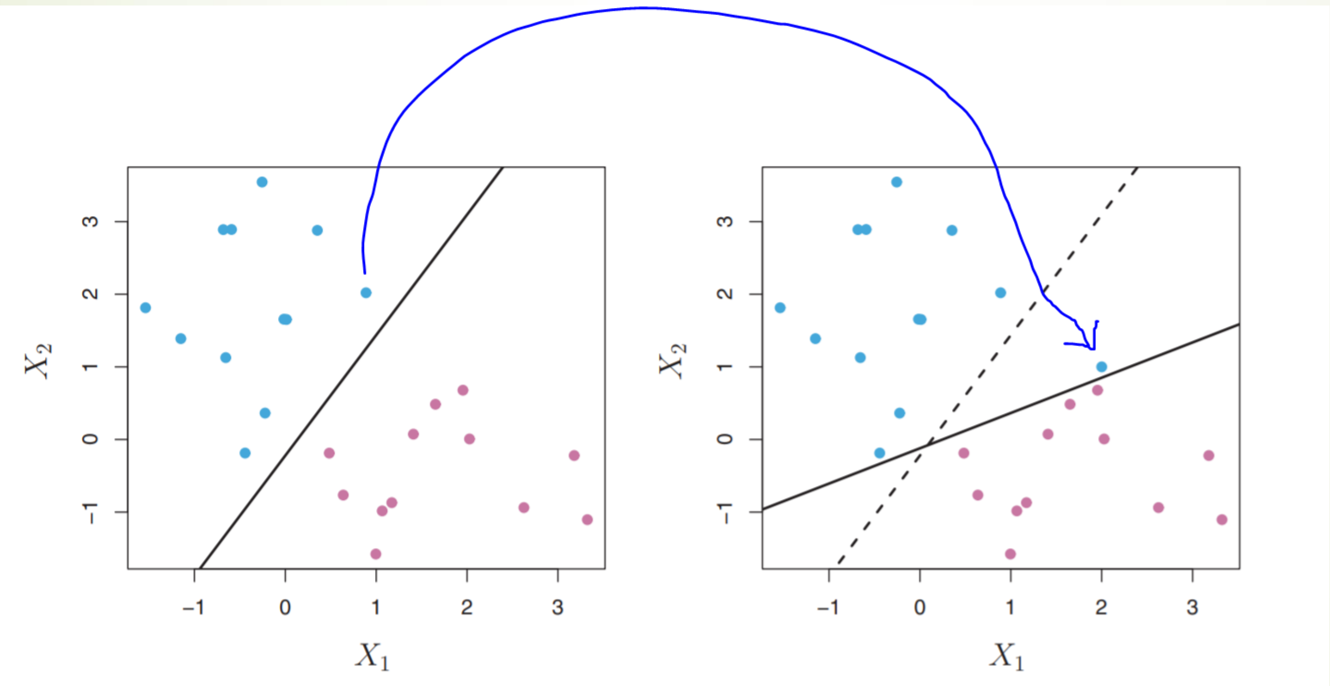


- **C** is the coefficient for the trade-off
  - Trade-off is everywhere

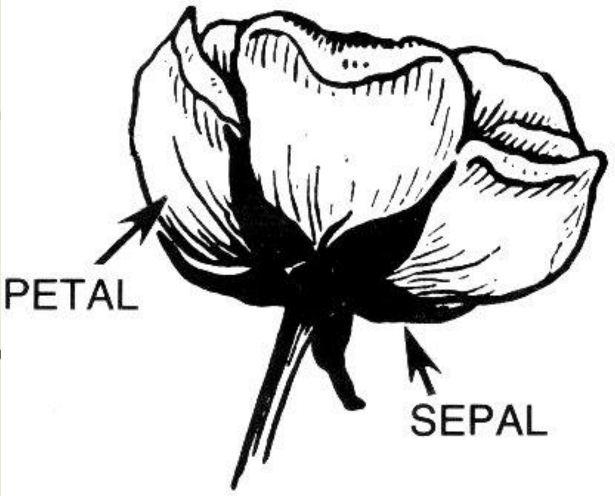
Minimal Cost-Complexity Pruning $R_\alpha(T) = R(T) + \alpha|T|$ where $|T|$ is the number of terminal nodes in T and total sample weighted impurity of the terminal nodes for $R(T)$.

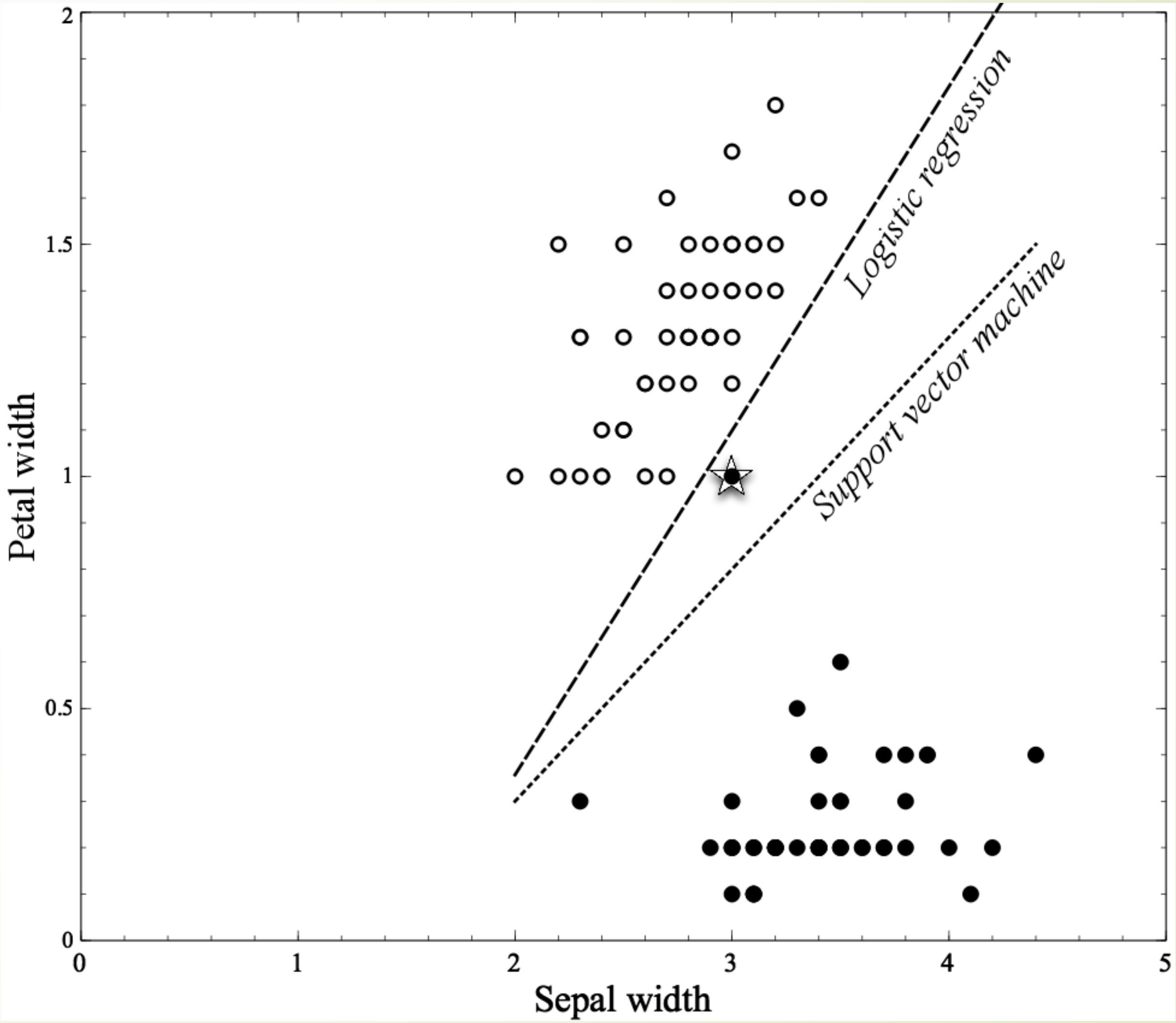
Intuition of Fat Margin

- We may want a classifier that may **misclassify a few** observations but with **fat margin** for:
  - Robustness for individual observation.
  - Better classification of **most** of the training observation **(rather than all)**



SVM vs. Logistic Regression

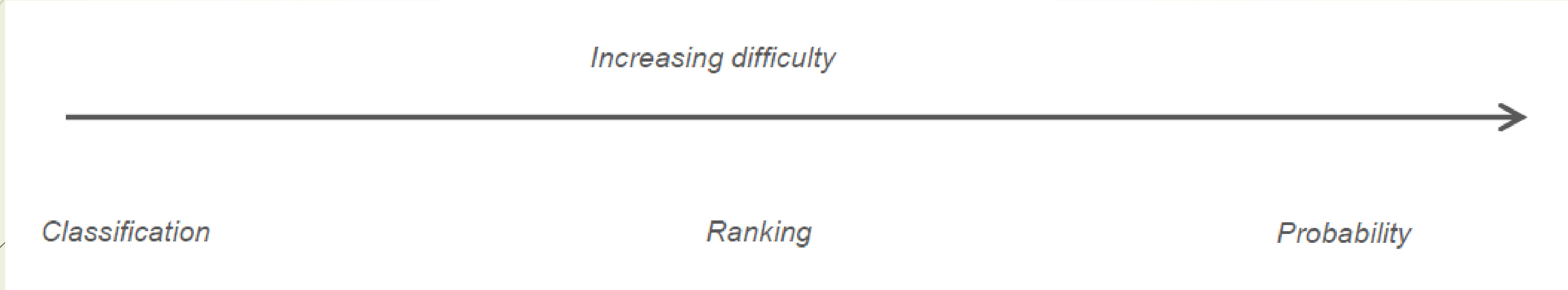




Outline

- Hinge Loss & SVM
- Nonlinear Models & Other Extensions
- Overfitting & Holdout Evaluation
- Quiz

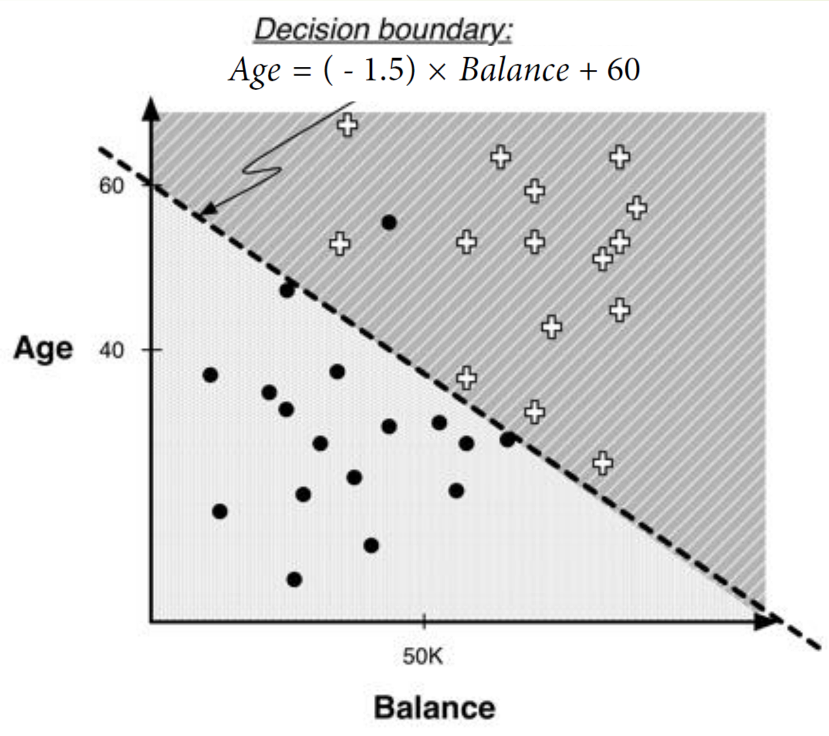
Classification/Ranking/Probability Estimation



- With estimated probability, we can computed the expected loss of fraud detection/churn prediction
  - ✓ We can further classify or rank samples easily
- Ranking can make classification results more useful
  - ✓ Locate limited budget to award/retain those users to be churned

Distance of Linear Discriminant Functions

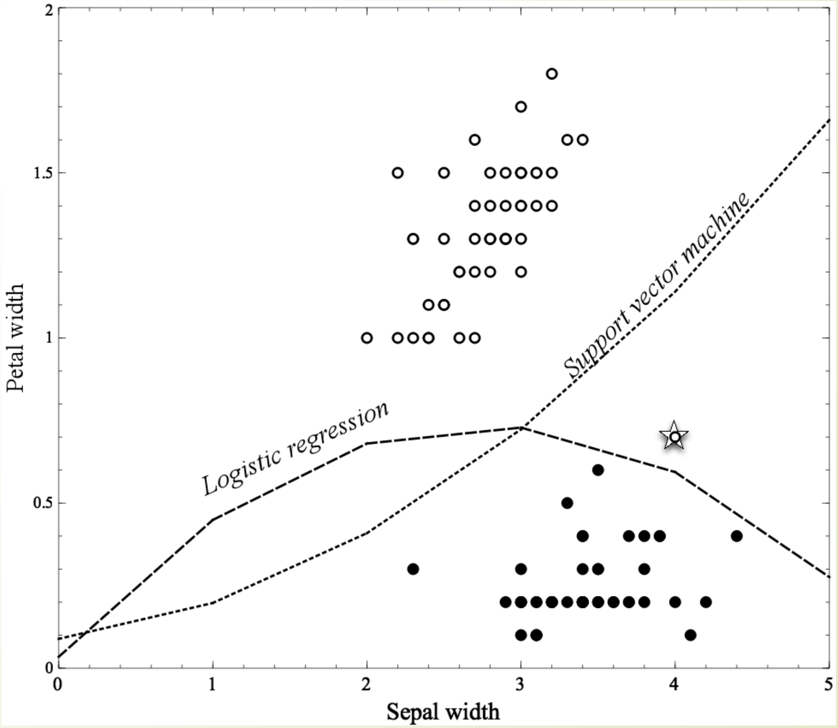
- When a sample near the decision boundary, we would be most uncertain about its class
- When it goes further away from the decision boundary, we would expect the higher likelihood of predicted class based on sign of linear discriminant functions
- **Distance** provides us free **ranking measure** of likelihood of correct estimation
- Logistic regression **directly** interpret the distance as class probability



Nonlinear Decision Boundaries

- Linear functions can actually represent nonlinear decision boundaries in higher dimension
  - if we extend the original feature by more complex features, and then we actually extend linear functions to nonlinear decision boundaries accordingly

Add Sepal width^2 to the original feature



Nonlinear Models-Example

- We can extend 2-d feature vector of each sample (both training and new data) by adding quadratic combinations to a 5-d feature vector

$$(x_1, x_2) \rightarrow (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \rightarrow (x'_1, x'_2, x'_3, x'_4, x'_5)$$

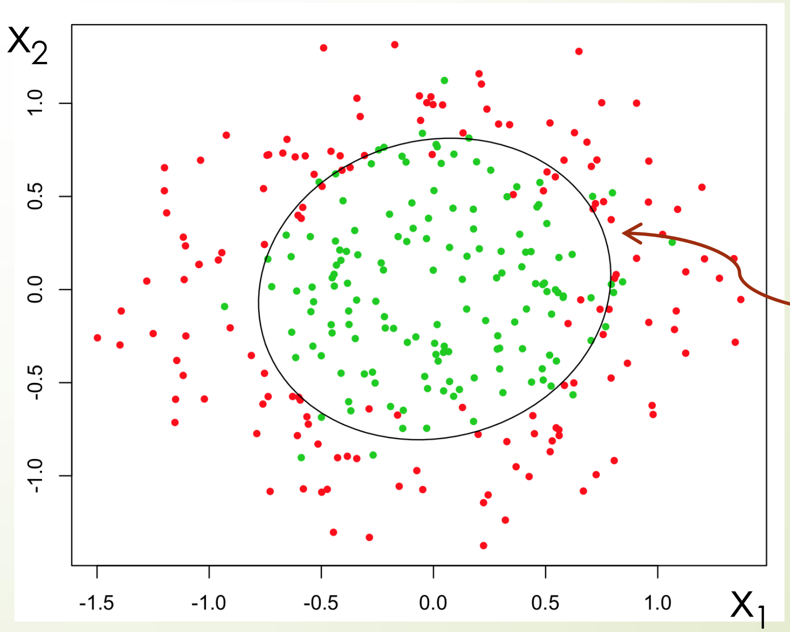
- Accordingly, the linear discriminant function in 5-d space changes to

$$f(\mathbf{x}) = w_0 + w_1 x'_1 + w_2 x'_2 + w_3 x'_3 + w_4 x'_4 + w_5 x'_5$$

- Then we can create a hyperplane in the 5-d space as follows to separate the data as good as the nonlinear function in 2-d space with

w_0=-0.54, w_1=-0.8, w_2=0

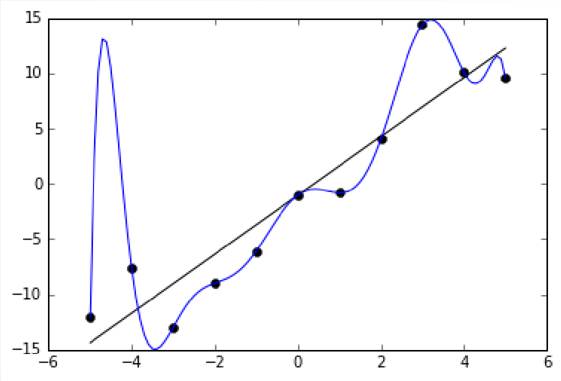
w_3=1, w_4=1, w_5=0



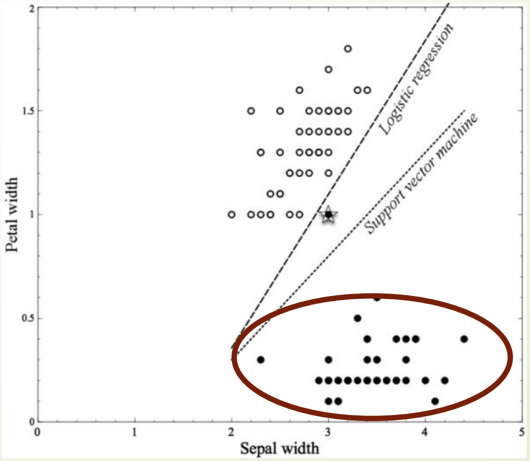
$(x_1-0.4)^2+x_2^2=0.7$

Other Nonlinear Models

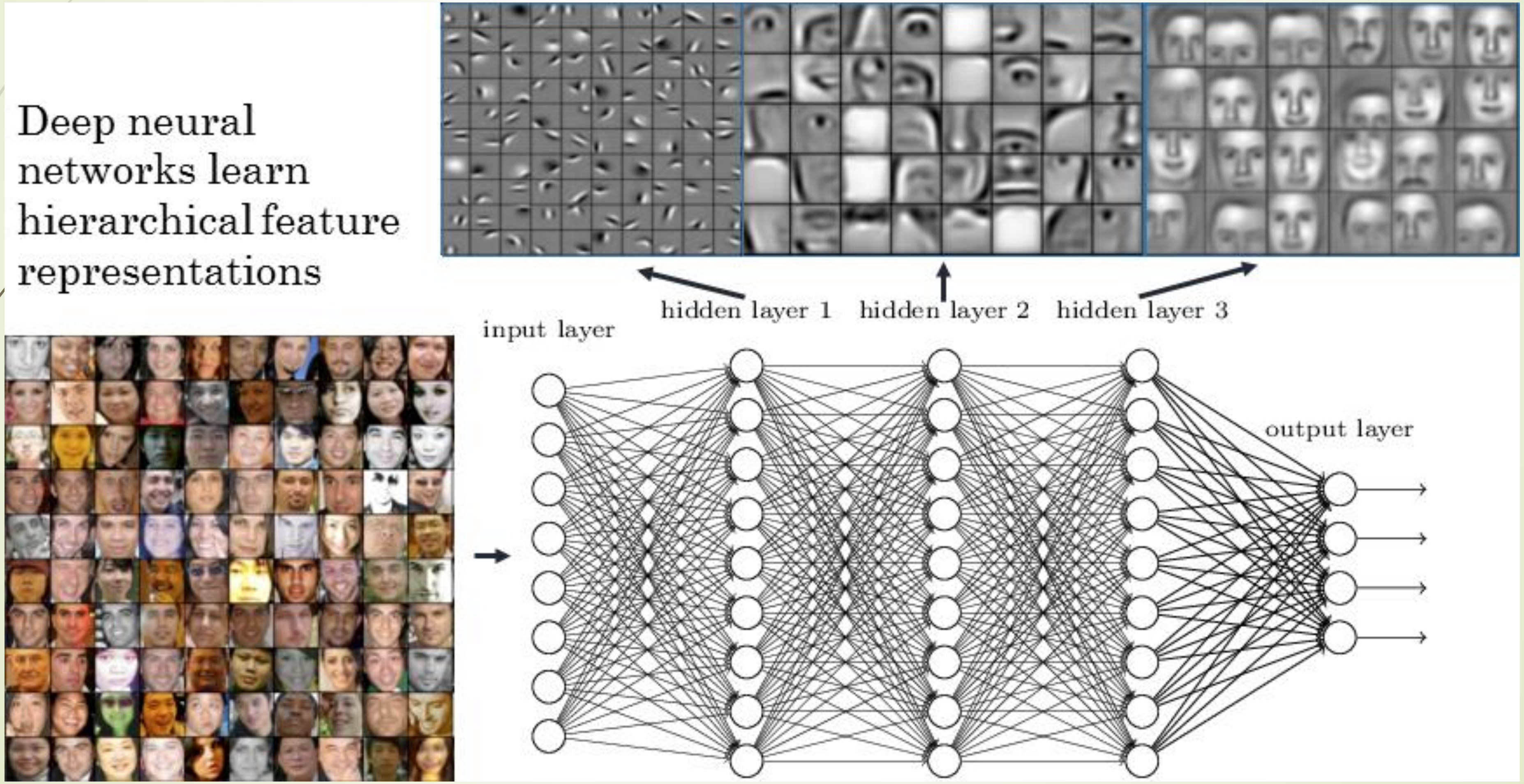
- Nonlinear support vector machine with a "polynomial kernel" consider "higher-order" combinations of the original features
  - 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' kernel in SVM
- Neural network as a "stack" of models
  - ✓ On the bottom of the stack are the original features
  - ✓ Each layer in the stack applies a simple model to the outputs of the previous layer
  - ✓ Middle layer tries to learn meaningful representation of original feature
- Might fit data too well (overfitting)



Regression

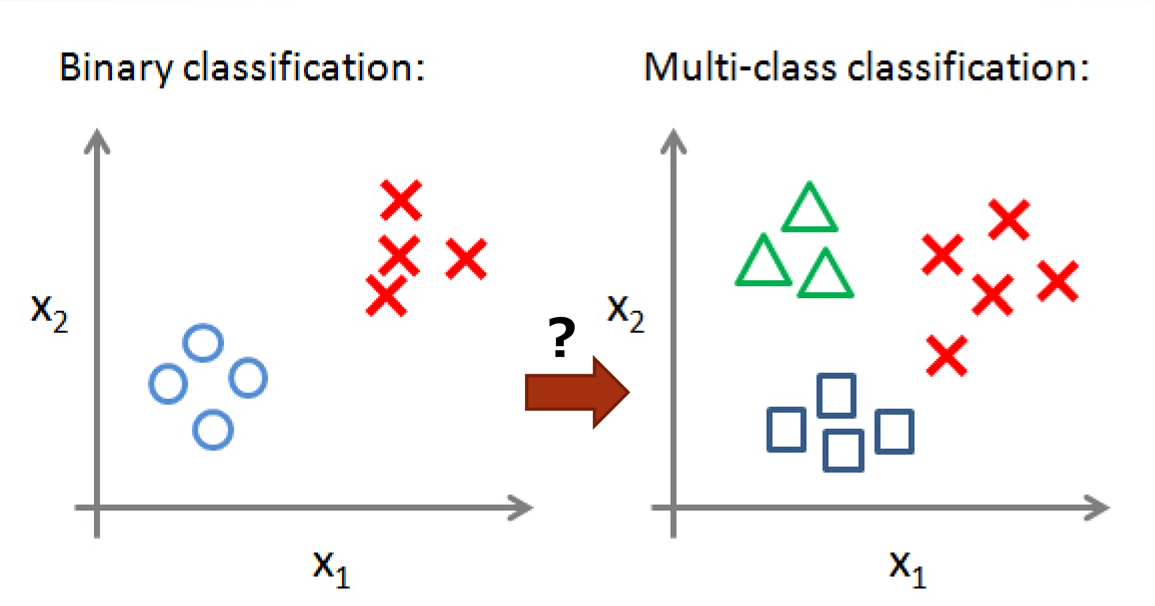


Representation Learning



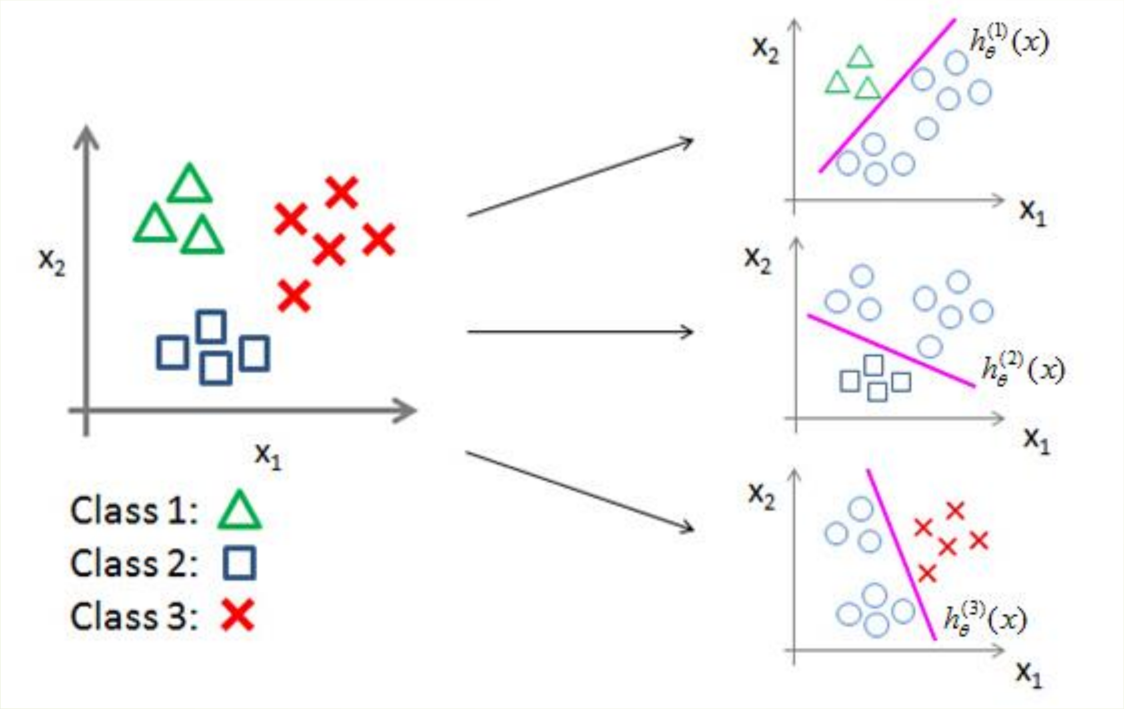
Binary to Multi-class classification

- We have studied binary classification extensively, but multi-class classification



One-versus-rest

- We can utilize binary classification model to do multi-class classification by training multiple binary classifier, and choose the predicted label with highest votes/probability



Recommended Resources

- Visualization: https://www.w3resource.com/graphics/matplotlib/
- Data Collection:
  - Selenium: https://selenium-python.readthedocs.io/
  - Chrome: F12 is your friend
  - Fiddler: https://www.telerik.com/blogs/how-to-capture-android-traffic-with-fiddler
  - Python Scrapy: https://docs.scrapy.org/en/latest/intro/tutorial.html

Outline

- Hinge Loss & SVM
- Nonlinear Models & Other Extensions
- Overfitting & Holdout Evaluation
- Quiz

Logistic Regression vs. Decision Tree

- Predict whether is cell (with image) are breast cancer or not

Attribute name	Description
RADIUS	*Mean of distances from center to points on the perimeter*
TEXTURE	*Standard deviation of grayscale values*
PERIMETER	*Perimeter of the mass*
AREA	*Area of the mass*
SMOOTHNESS	*Local variation in radius lengths*
COMPACTNESS	*Computed as: perimeter^2/area – 1.0*
CONCAVITY	*Severity of concave portions of the contour*
CONCAVE POINTS	*Number of concave portions of the contour*
SYMMETRY	*A measure of the symmetry of the nucleii*
FRACTAL DIMENSION	*'Coastline approximation'* – 1.0
DIAGNOSIS (Target)	*Diagnosis of cell sample: malignant or benign*

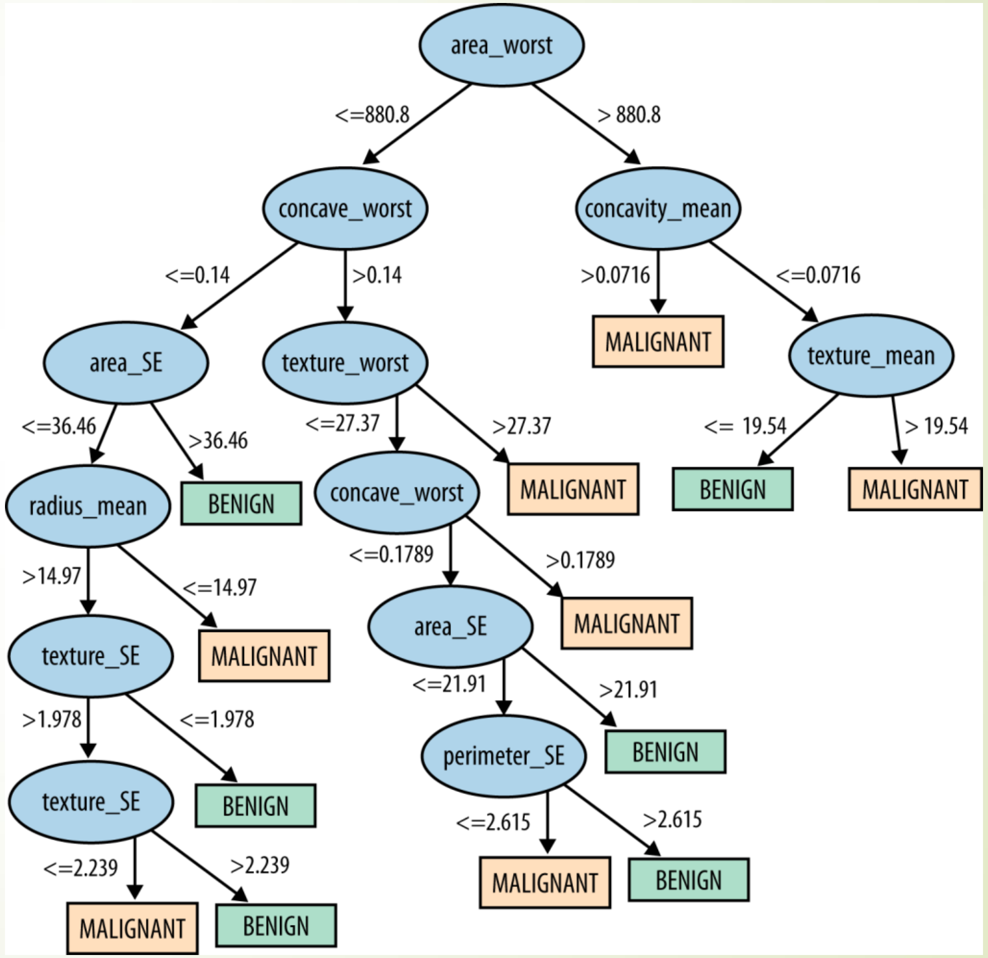
From each of these basic characteristics, three values were computed:

- the mean (_mean),
- standard error (_SE),
- “worst” or largest

Logistic Regression vs. Decision Tree

Attribute	Weight (learned parameter)
SMOOTHNESS_worst	22.3
CONCAVE_mean	19.47
CONCAVE_worst	11.68
SYMMETRY_worst	4.99
CONCAVITY_worst	2.86
CONCAVITY_mean	2.34
RADIUS_worst	0.25
TEXTURE_worst	0.13
AREA_SE	0.06
TEXTURE_mean	0.03
TEXTURE_SE	−0.29
COMPACTNESS_mean	−7.1
COMPACTNESS_SE	−27.87
w_0 *(intercept)*	−17.7

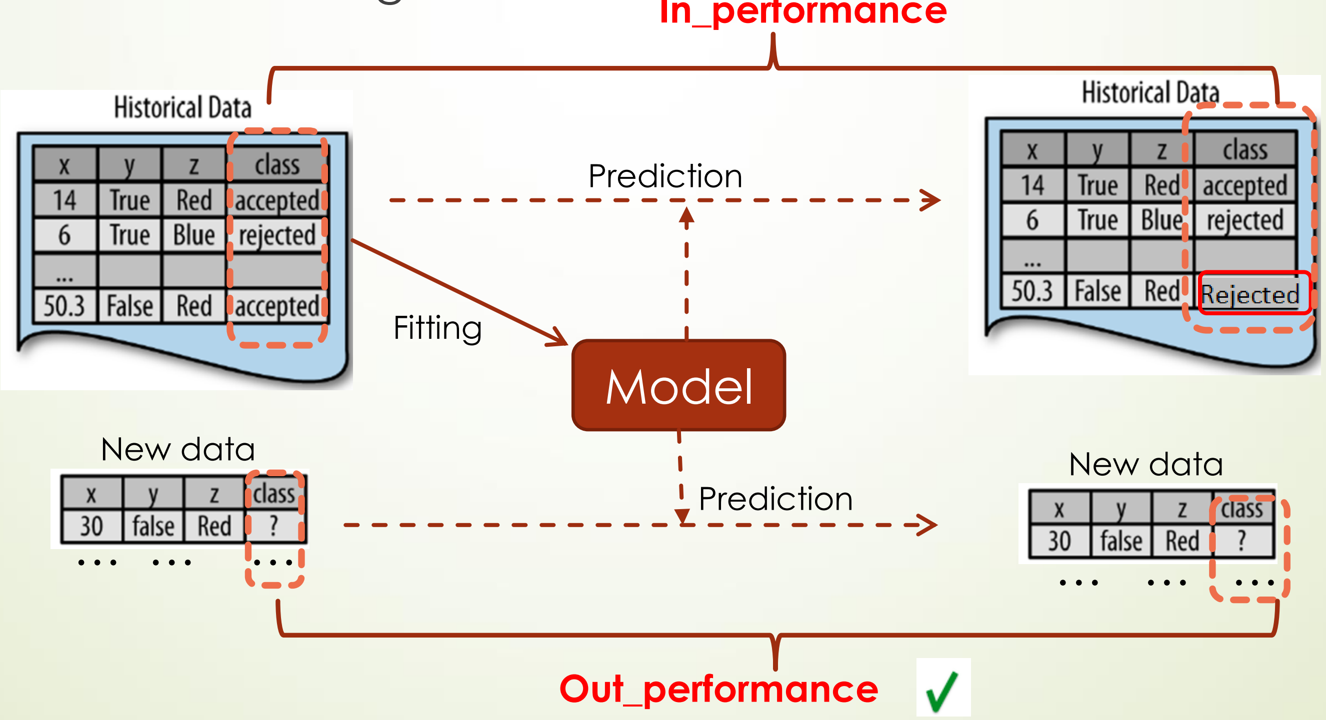
Linear equation learned by logistic regression
acc. = 98.9%



Deducted decision tree(J48)
acc. = 99.1%

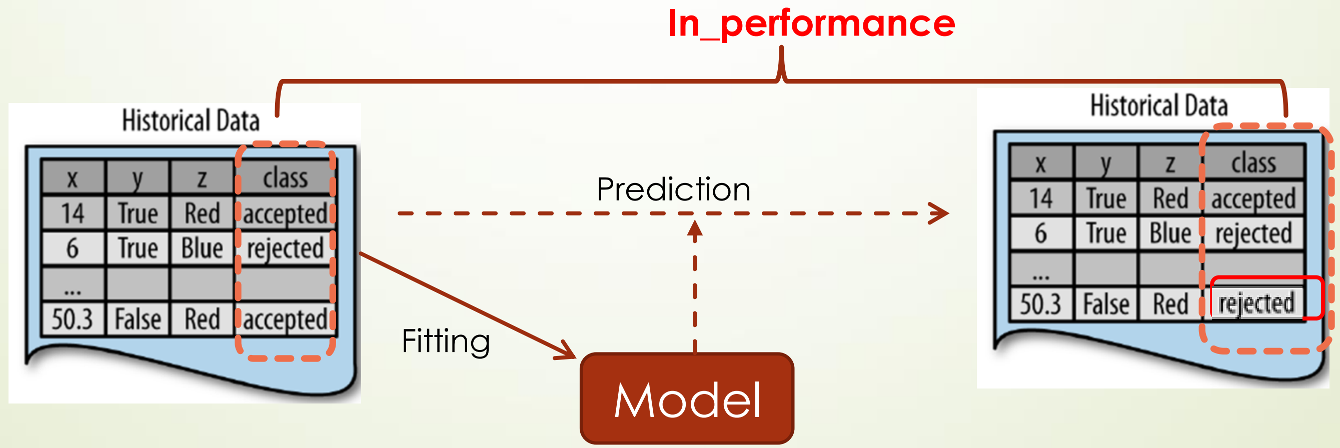
Generalization

- We build predictive models to identify **general patterns** that predict the target attribute well for instances that we have not observed yet (Unknown new data)
  - ✓ Good model: **generalize** well on the unknown data
  - ✓ Model performance on the unknown data (***out_performance***) is more important but impossible to measure in general



Overfitting

- We try to optimize **in_performance** when fitting predictive models to training data
  - ✓ We measure the in_performance by error(/loss) or accuracy
  - ✓ But good in_performance **DO NOT** guarantee good out_performance
- Finding chance occurrences in data that look like interesting patterns, but which do not generalize, is called ***overfitting the (training) data***
  - ✓ Overfitted model has ***good* in_performance**, but **poor out_performance**
  - ✓ We are easily fooled by overfitted model



"Table" Model

- "Table" Model has perfect in_performance (100% accuracy)
  - It does not make a single mistake, identifying correctly all the churners as well as the nonchurners.
  - Fitting : store the feature vector for each customer who has churned in a database table T_c
  - Prediction: it takes the customer's feature vector, looks his/her up in T_c, and reports "100% likelihood of churning" if she is in T_c and "0% likelihood of churning" if he/she is not in
- "Table" Model but terrible out_performance
  - Will predict "0% likelihood of churning" for every customer (not in the training data)
- A model that **looked perfect** would be completely **useless** in practice!

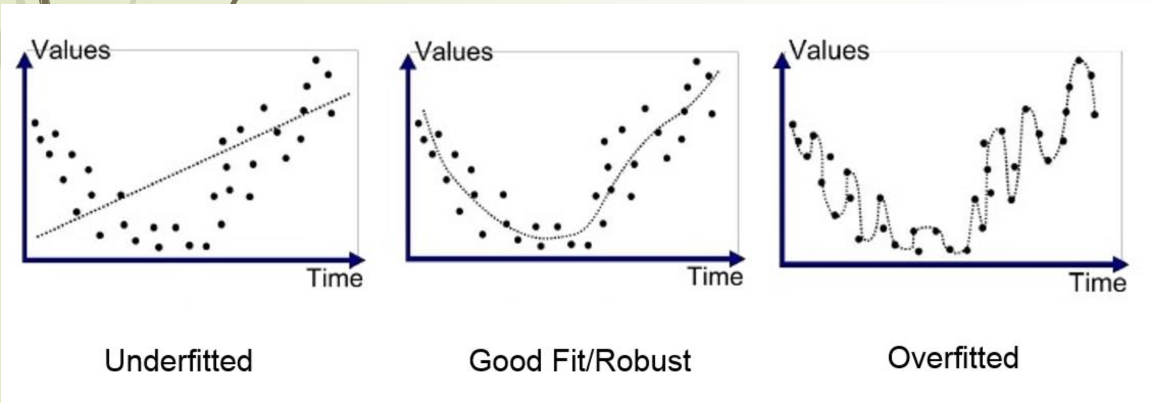
Overfitting !

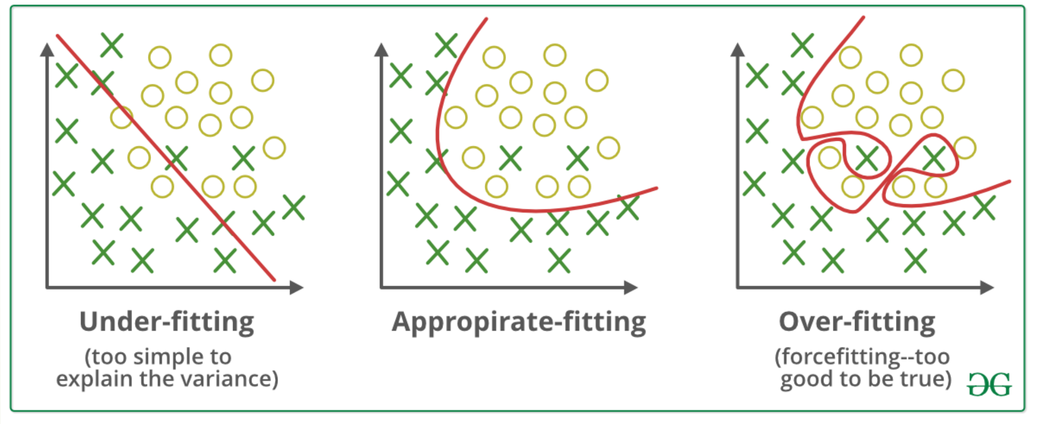
Overfitting and Model Complexity

- "If you torture the data long enough, it will confess" -- Ronald Coase
  - ✓ If we are allowed to use more complex models, we will have higher flexibility to pick up complex but effective patterns(good in_performance)
  - ✓ More complex patterns are more likely to be just chance occurrences in the training data and then cannot generalize
  - ✓ Dubious patterns: second character of living city name for credit scoring …
- Increase model complexity can improve in_performance, but may not improve out_performance (that's why overfitting)
  - ✓ Complexity of decision tree: nodes in the tree
  - ✓ Complexity of discriminant functions: dimension of feature vector, nonlinear or linear

Trade-off of Model Complexity

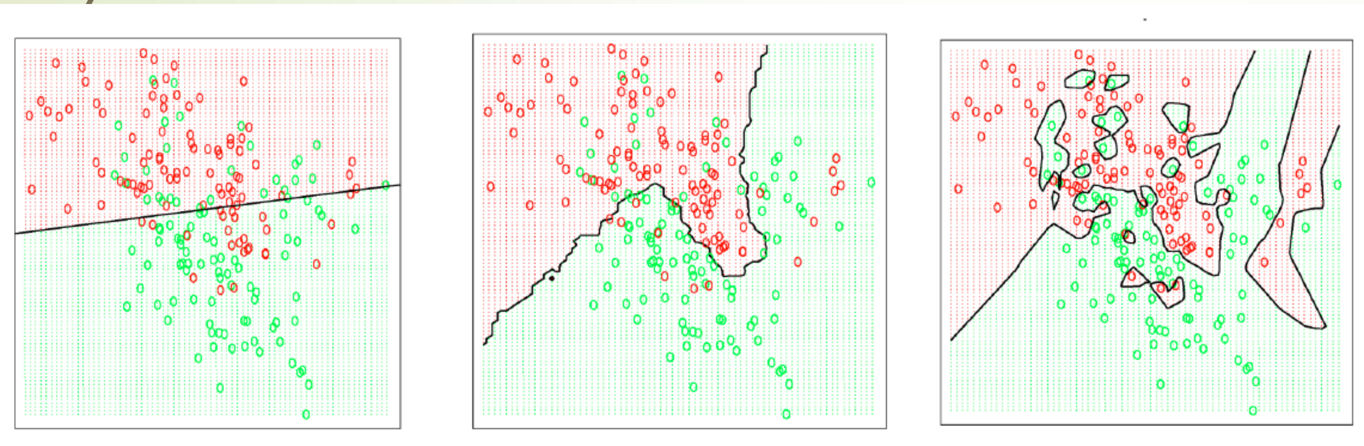
- Trade-off between model complexity and the possibility of overfitting is fundamental in data mining
  - ✓ There is no single choice or silver bullet that will eliminate over-fitting, we need to recognize over-fitting and manage complexity in a principled way
  - ✓ All data mining procedures have **the tendency to over-fit** to some extent since they usually tailor models to the training data (some ones are better)



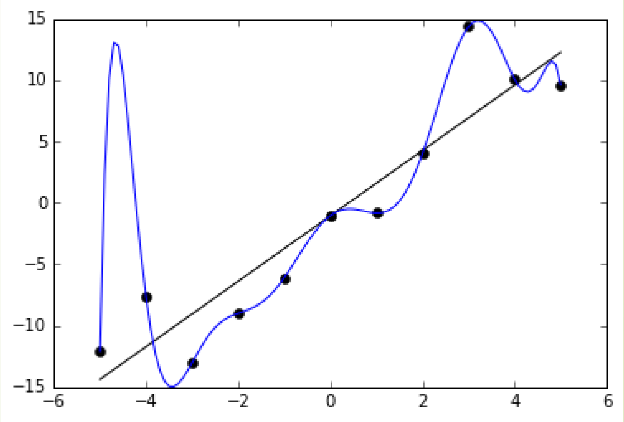


Overfitting Mathematical Functions

- Increase the complexity of mathematical functions $f(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$
  - ✓ Add more variables (more attributes): $f(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \boxed{w_4x_4 + w_5x_5}$
  - ✓ Extend existing variables: $f(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + \boxed{w_6x_1^2 + w_7 * x_2/x_3}$
- As you increase the dimensionality, you can perfectly fit larger and larger sets of arbitrary points
  - ✓ Carefully prune the attributes in order to avoid overfitting -> manual/auto selection



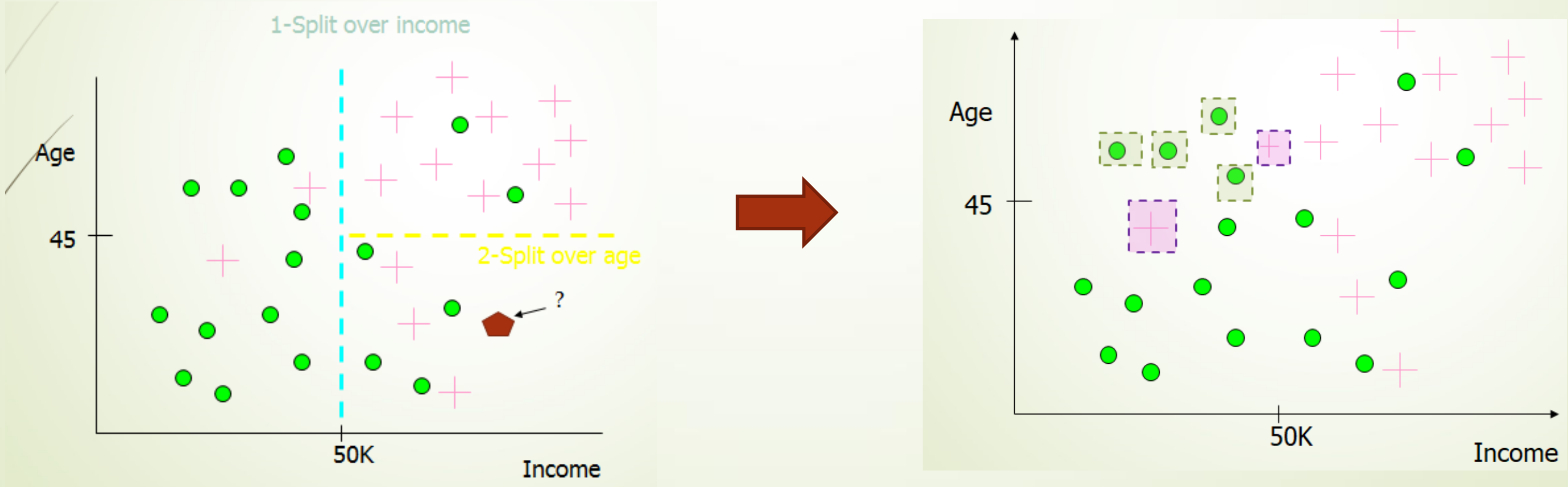
Classification



Regression

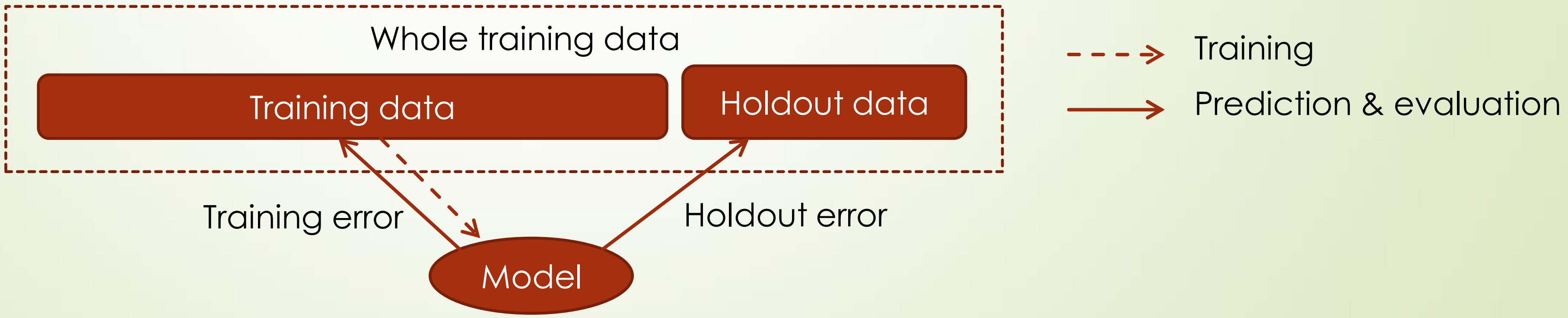
Overfitting Decision Tree

- **Tree induction**: find important, predictive individual attributes recursively to smaller and smaller data subsets
  - ✓ We can build a perfect model if we are allowed to split the data/data subsets as many times as we can such that each leaf node is pure
  - ✓ The extreme case would be that each leaf node just contains one training example
  - ✓ Similar to look-up table, have perfect in_performance but poor out_performance (slightly better than look-up table)



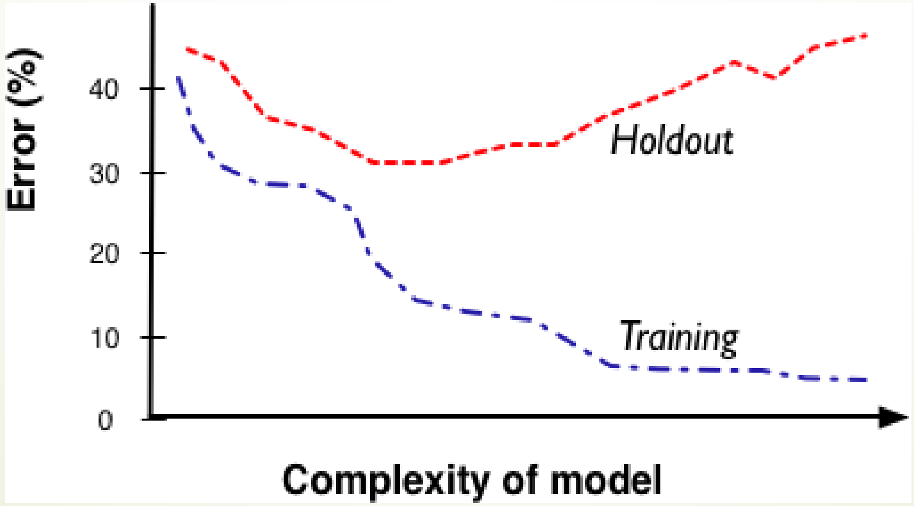
Examine Overfitting

- Overfitted model has good **in_performance**, but poor **out_performance**
- *Generalization performance* is measured by **out_performance**
- It is hard to measure **out_performance** on unknown data in general way
  - ✓ Approximating the out_performance by performance evaluation conducted on <u>data that have not been used for training (holdout data)</u>
  - ✓ Split the whole training data into two parts: subset of training data and holdout data
  - ✓ Train model on training data, and then predict & evaluate performance on training data(training error) and holdout data (holdout error) respectively.



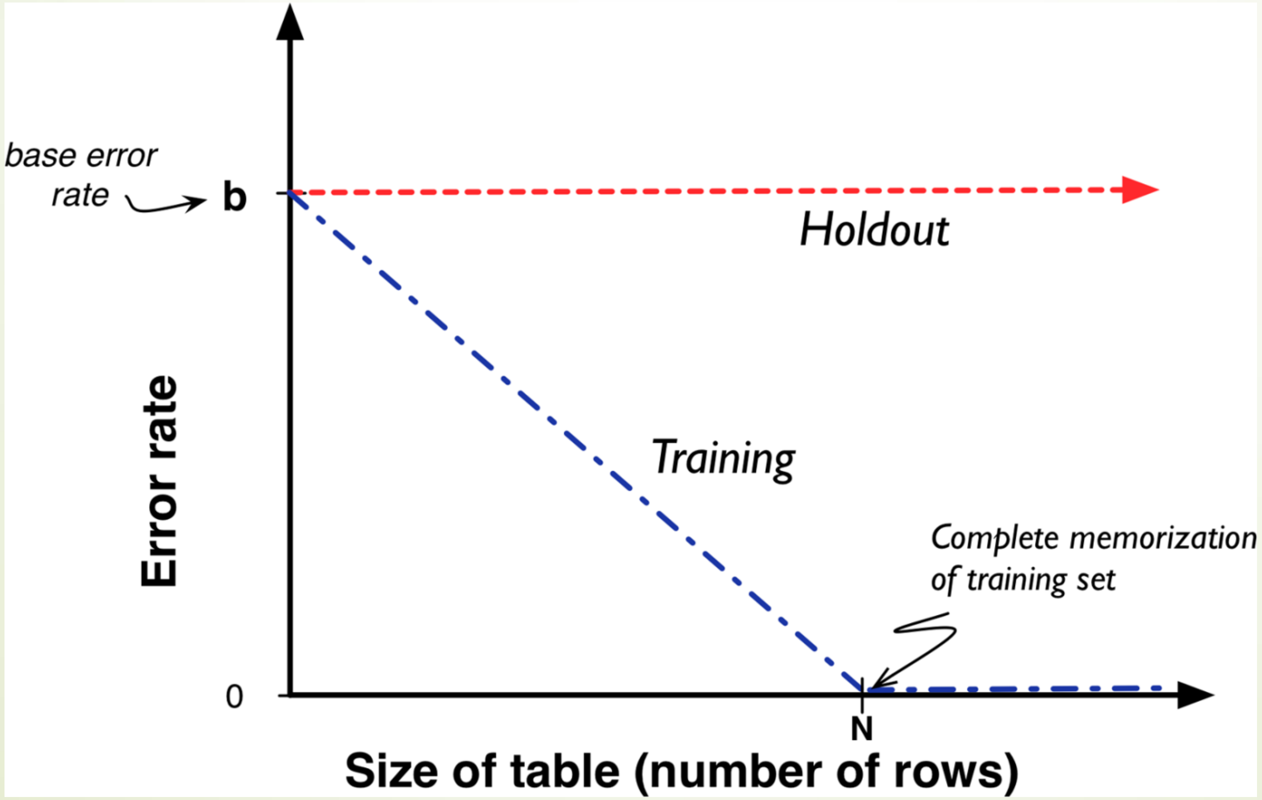
Fitting Graph

- A fitting graph shows the performance(error/accuracy) of a model on training data and holdout data as a function of complexity
  - ✓ Focusing on the performance on the holdout data



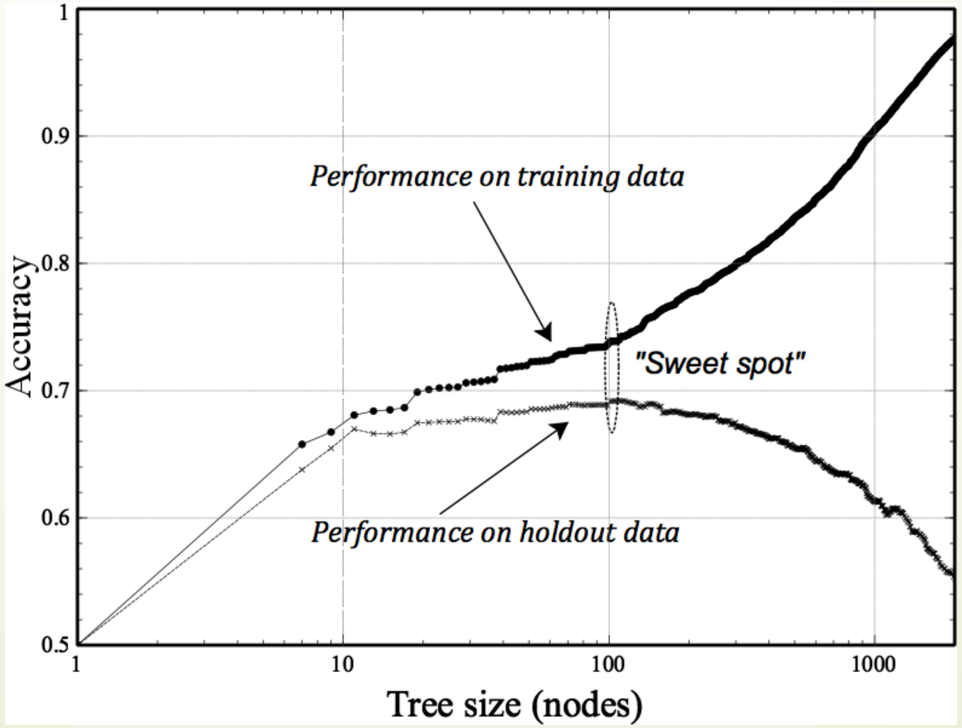
Fitting Graph of Table Model

- When new data comes to a table, make the prediction to be not churned
  - N is the number of churned examples in training set
  - M is total number of examples in training set, b=N/M



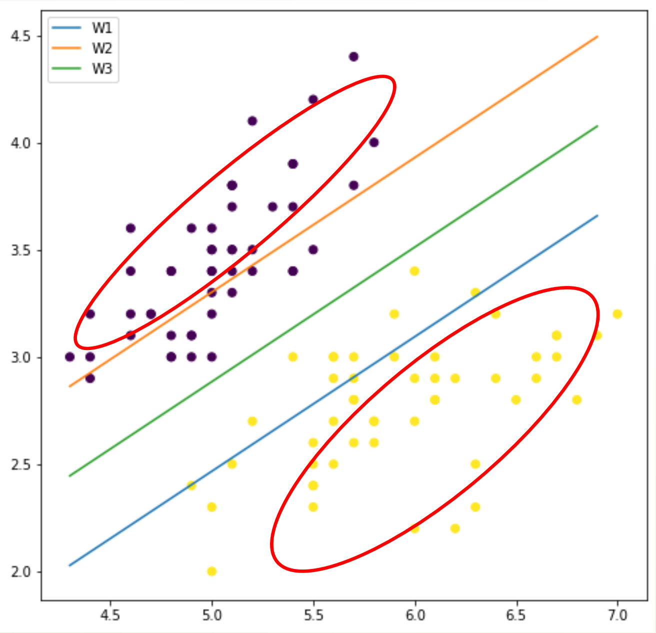
Fitting Graph of Decision Tree

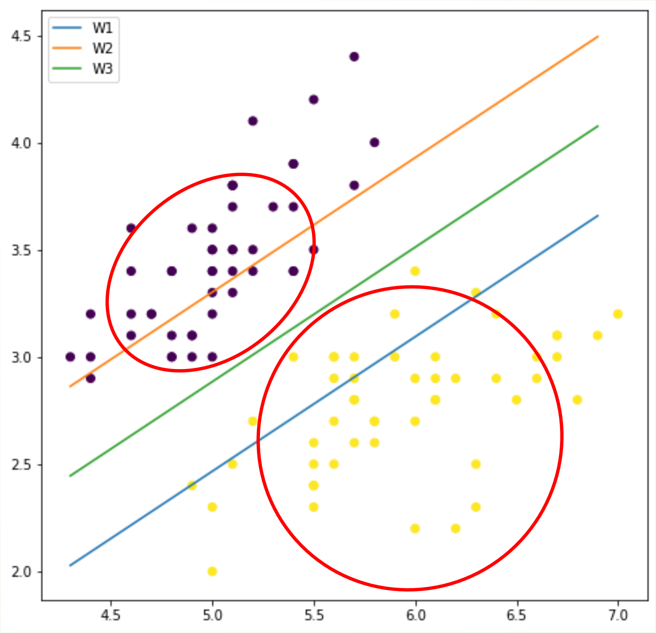
- Measure model complexity by number of nodes in decision tree
  - ✓ Sweet spot represents the best trade-off between the extremes of (i) **not splitting** the data at all and simply using the average target value in the entire dataset, and (ii) building a complete tree out until **all** the leaves are pure.



Drawbacks of Holdout Evaluation

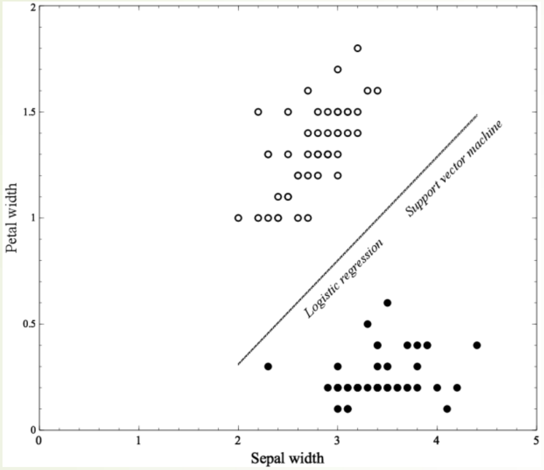
- While a holdout set will indeed give us an estimate of generalization performance, it is just a single estimate.
  - ✓ A single estimate of model accuracy might have just been a single particularly lucky (or unlucky) choice of training and test data
  - ✓ Should we have any confidence in our estimation?



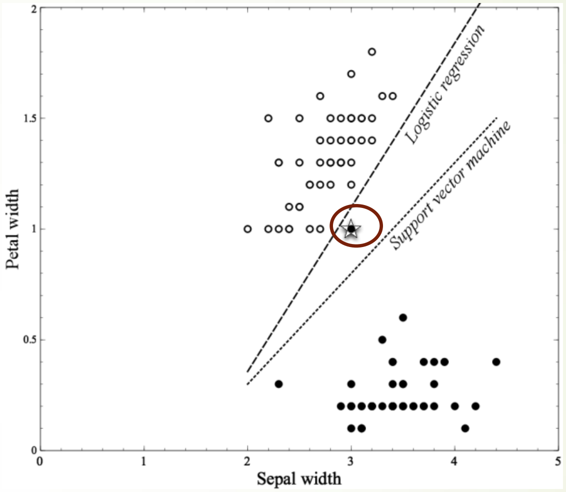


How to Avoid Overfitting?

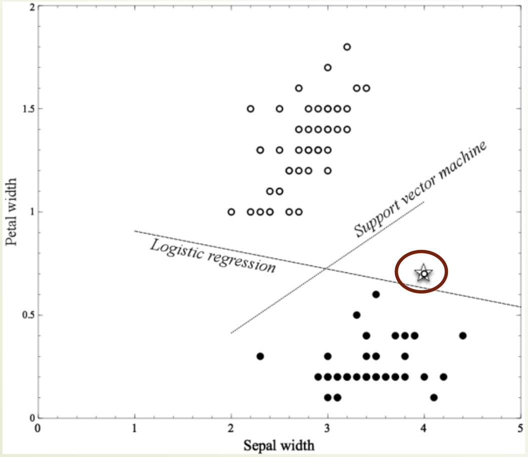
- Logistic regression are more likely to be overfitted than SVM



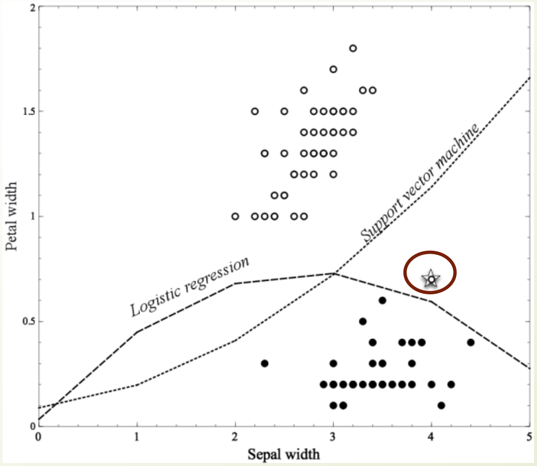
Original data



Adding one outlier



Adding one outlier



Adding one outlier with nonlinear feature

Classification Models

- Linear classifier
  - ✓ Logistic Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
  - ✓ LinearSVC https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
  - ✓ Perceptron https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron
- Non-linear classifier
  - ✓ SVC: SVM for classification : https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
  - ✓ Decision Tree: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
- Comparison
  - ✓ https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

Outline

- Hinge Loss & SVM
- Nonlinear Models & Other Extensions
- Overfitting & Holdout Evaluation
- Quiz

Lab Quiz

- **Deadline**: 17:59 p.m., Mar. 20, 2020
- Two questions accounting for **5%** of overall score
- **Upload** the **answer worksheet** and the accomplished **Python files** to the **Blackboard**
- You may submit **unlimited times** but only the **LAST** submission will be considered
- **Only the answers in answer sheet** will be referred for grading
- Note： **MUST attach ALL** the required files in every submission/resubmission, otherwise other files will be missing.