Predictive Modeling with Loss Function

1

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

- Decision Tree (Cont'd)
- Loss Function & Linear Regression
- Logistic Regression & SVM (Intuition)
- Quiz

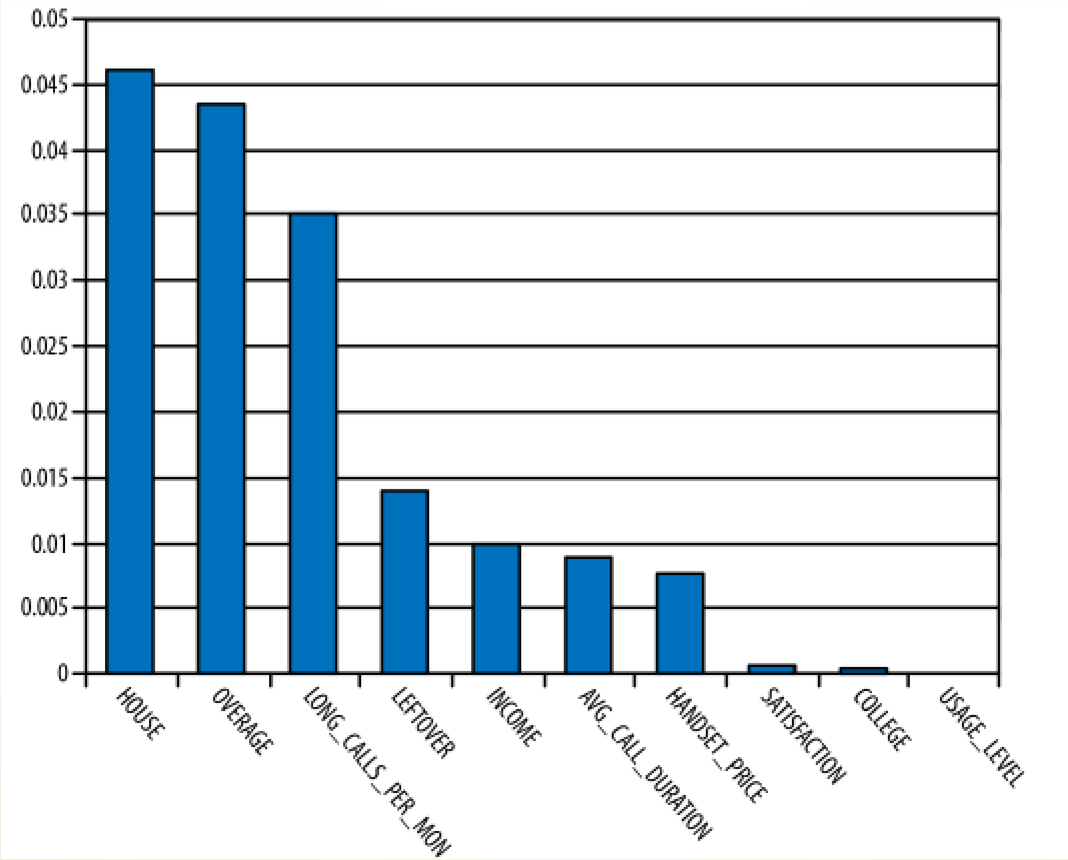
Example of Churn Prediction (1)

- **Given a historical data set of 20,000 customers**
  - ✓ Each customer either had stayed with the company or had left (churned)
  - ✓ Each customer is described by following attributes
  - ✓ How could we predict the churn probability of a new customer

Variable	Explanation
COLLEGE	Is the customer college educated?
INCOME	Annual income
OVERAGE	Average overcharges per month
LEFTOVER	Average number of leftover minutes per month
HOUSE	Estimated value of dwelling (from census tract)
HANDSET_PRICE	Cost of phone
LONG_CALLS_PER_MONTH	Average number of long calls (15 mins or over) per month
AVERAGE_CALL_DURATION	Average duration of a call
REPORTED_SATISFACTION	Reported level of satisfaction
REPORTED_USAGE_LEVEL	Self-reported usage level
LEAVE *(Target variable)*	Did the customer stay or leave (churn)?

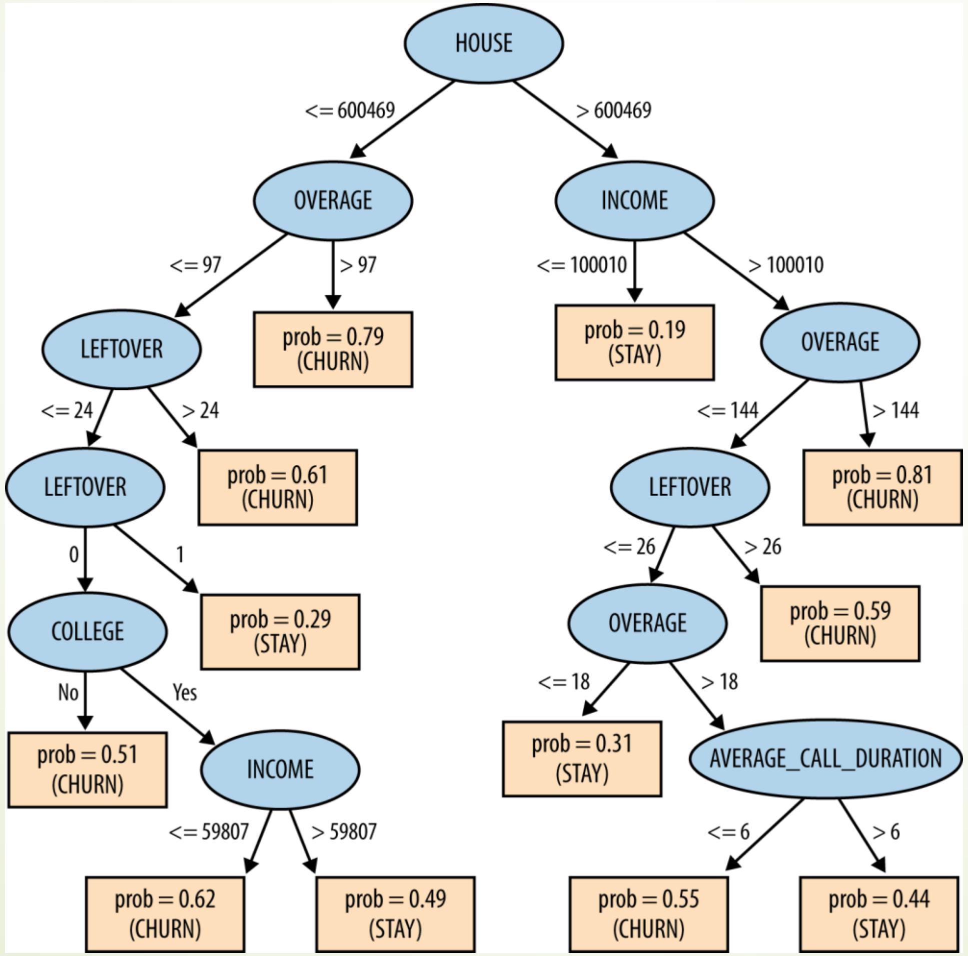
Example of Churn Prediction (2)

- **Ranking 10 informative attributes by information gain**



Example of Churn Prediction (3)

- **Recursively apply attribute selection and segmentation**



When to Stop Growing

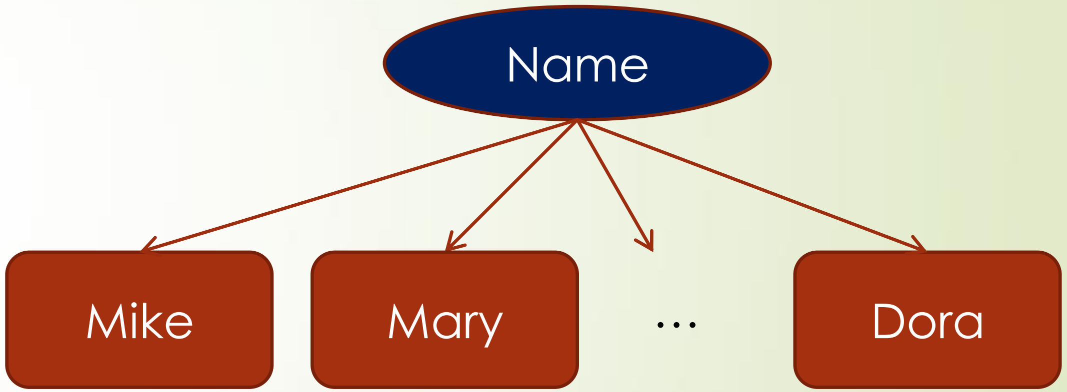
- **Grow as long as we have positive information gain?**

Attributes

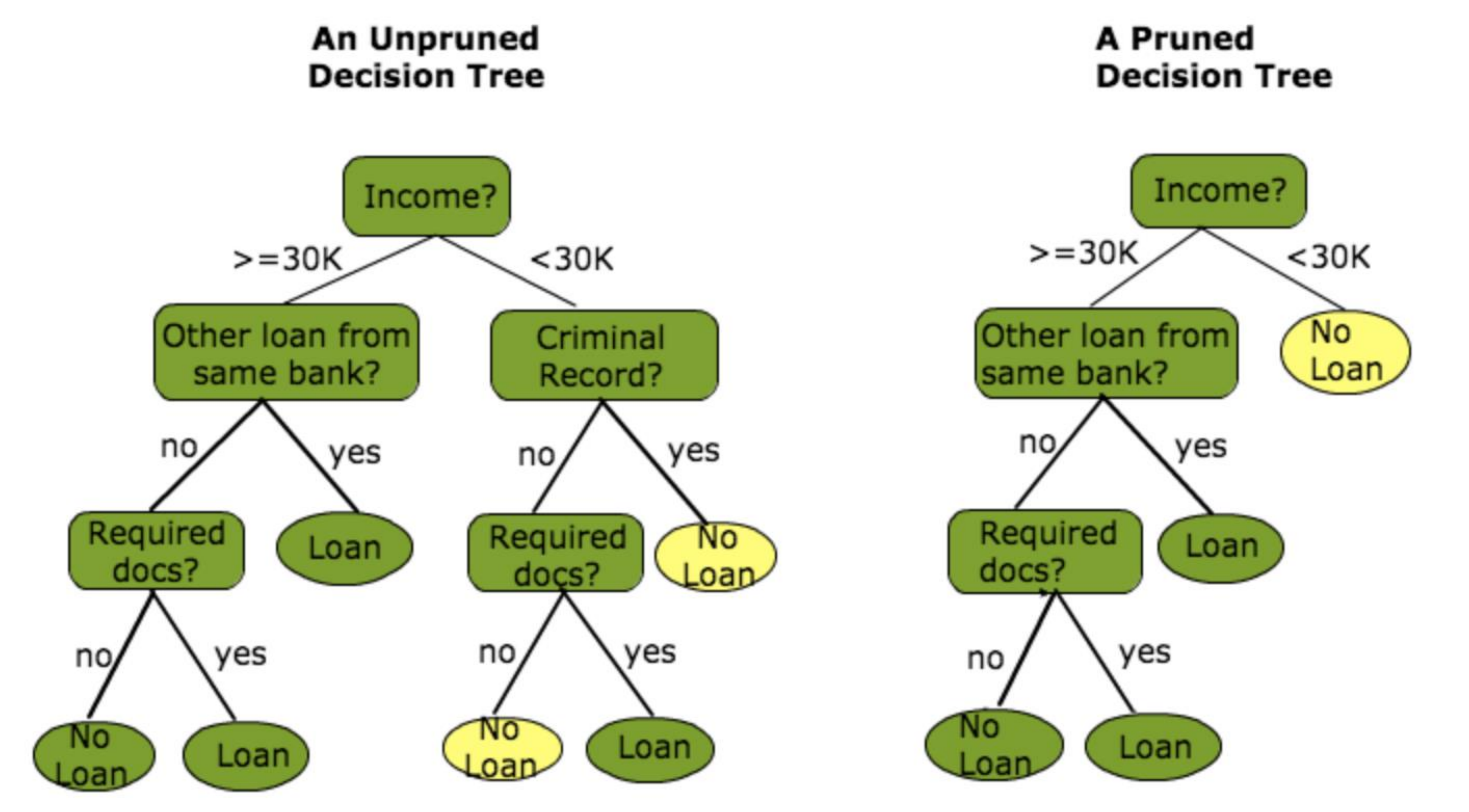
Target attribute

Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**



Decision Tree Pruning



Pruning Approaches

- **Pre-pruning (Early stopping):** stop growing the tree earlier
  - ✓ Minimum number(proportion) of samples required at a leaf node
  - ✓ The maximum depth
  - ✓ Minimum number of samples required to split an internal node
  - ✓ The number of features to consider when looking for the best split…
- **Post-pruning:** allows the tree to fully grow first, and then post prune it
  - ✓ Reduced error pruning
  - ✓ **Minimal** Cost-Complexity Pruning (Sklearn)

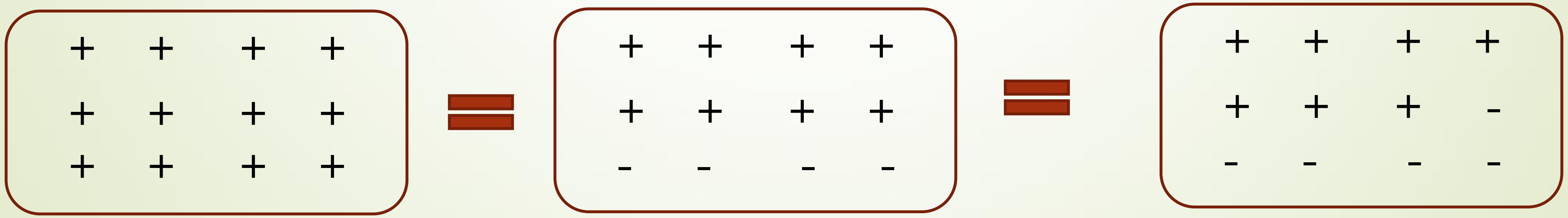
$$R_\alpha(T) = R(T) + \alpha|T|$$

where $|T|$ is the number of terminal nodes in T and

total sample weighted impurity of the terminal nodes for $R(T)$.

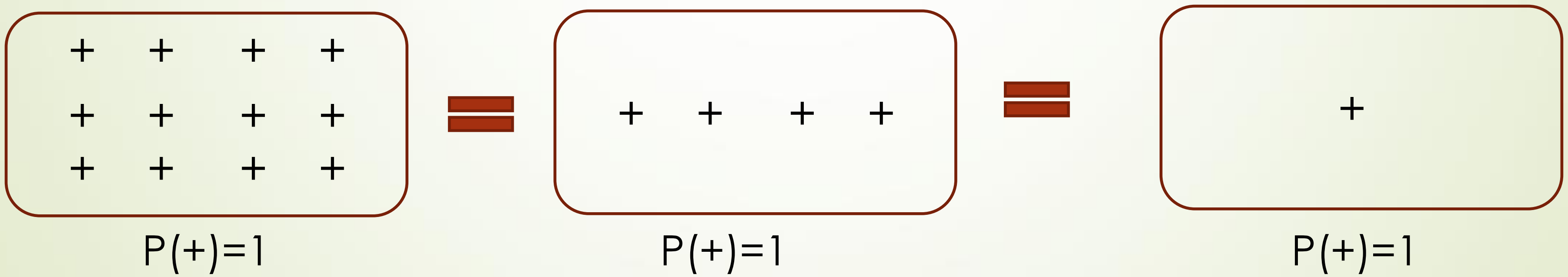
Probability Estimation

- **We often need a more informative prediction than just a classification**
  - ✓ E.g. allocate your budget to the instances with the highest expected loss
  - ✓ Each segment (leaf)to be assigned an estimate of the probability of membership in the different classes
- **Classification may oversimplify the problem**
  - ✓ E.g. if all segments have a probability of <0.5 for write-off,
  - ✓ All instances within one segment/leaf node will be labeled “not write-off“ once over 50% of all these instances are labeled “not write-off“



Probability Estimation by Frequency

- **Frequency-based estimate**
  - ✓ E.g. allocate your budget to the instances with the highest expected loss
  - ✓ If a leaf contains $\boldsymbol{n}$ positive instances and $\boldsymbol{m}$ negative instances (binary classification), the probability of any new instance being positive may be estimated as $\frac{n}{n+m}$
  - ✓ Vulnerable to noise



Laplace Correction

- **Laplace correction provide** "smoothed" version of the frequency-based estimate
  - ✓ If a leaf contains n positive instances and m negative instances (binary classification), the probability of any new instance being positive may be estimated as $\frac{n+1}{n+m+2}$

+ + + + + + + + + + + +	≥	+ + + +	≥	+
P(+)=13/14≈0.93		P(+)=5/6 ≈0.83		P(+)=2/3 ≈0.67

Decision Tree Algorithms

- **There are a lot of algorithms for building decision tree, ID3, C4.5, C5.0, CART**
  - ✓ Impurity measures: information entropy, Gini index, Variance…
  - ✓ Impurity improvement measures: information gain, information gain ratio
  - ✓ Ways of handling missing values, overfitting, outliers

	Splitting Criteria	Attribute type	Missing values	Pruning Strategy	Outlier Detection
ID3	Information Gain	Handles only Categorical value	Do not handle missing values.	No pruning is done	Susceptible to outliers
CART	Towing Criteria	Handles both Categorical & Numeric value	Handle missing values.	Cost-Complexity pruning is used	Can handle Outliers
C4.5	Gain Ratio	Handles both Categorical & Numeric value	Handle missing values.	Error Based pruning is used	Susceptible to outliers

Intuition of Gini Index

$$Gini = 1 - \sum_j p_j^2$$

3 red and 1 blue:

$Gini\ Index = 1\text{-}(probability\_red^2 + probability\_blue^2) = 1\text{-}(0.75^2 + 0.25^2) = 0.375$

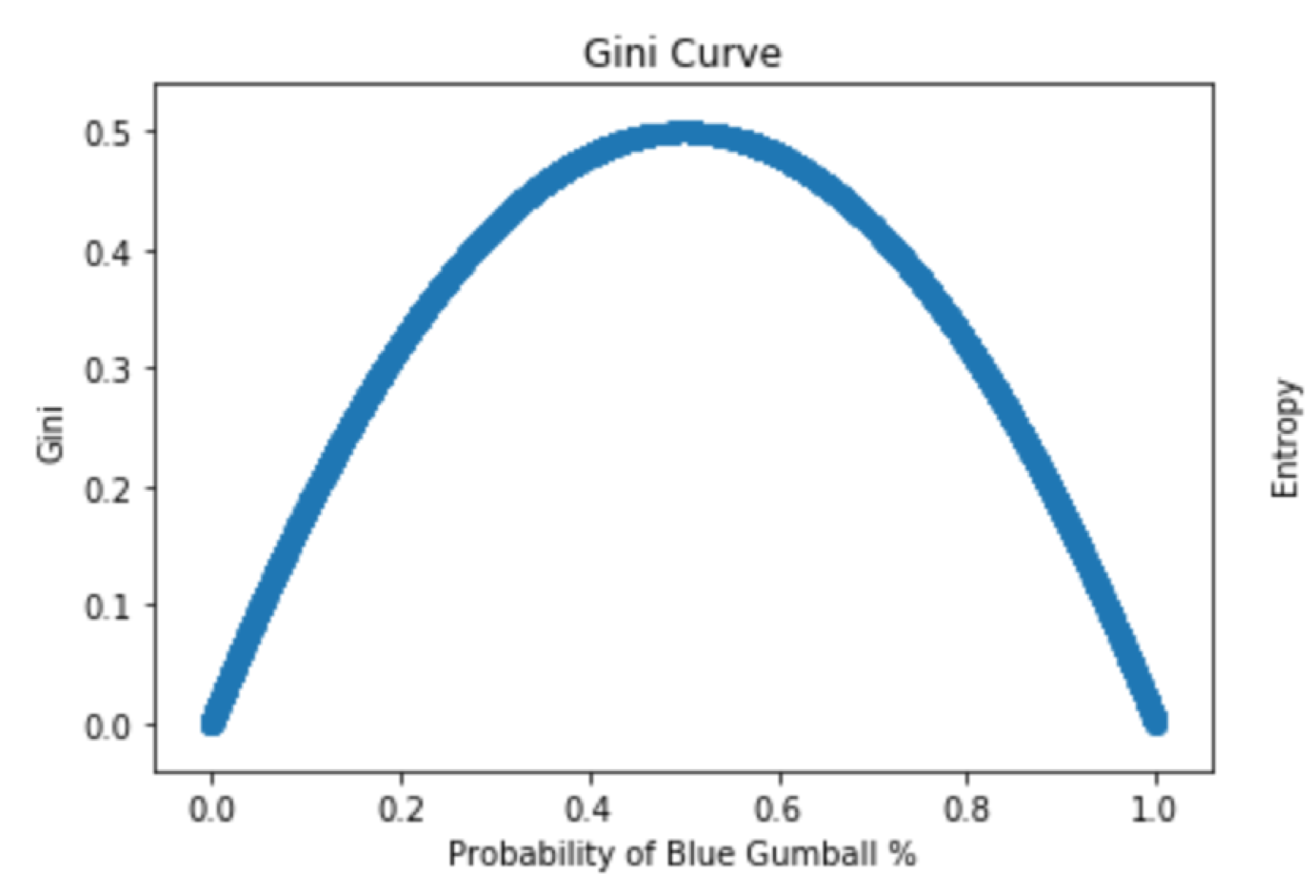
2 red and 2 blue:

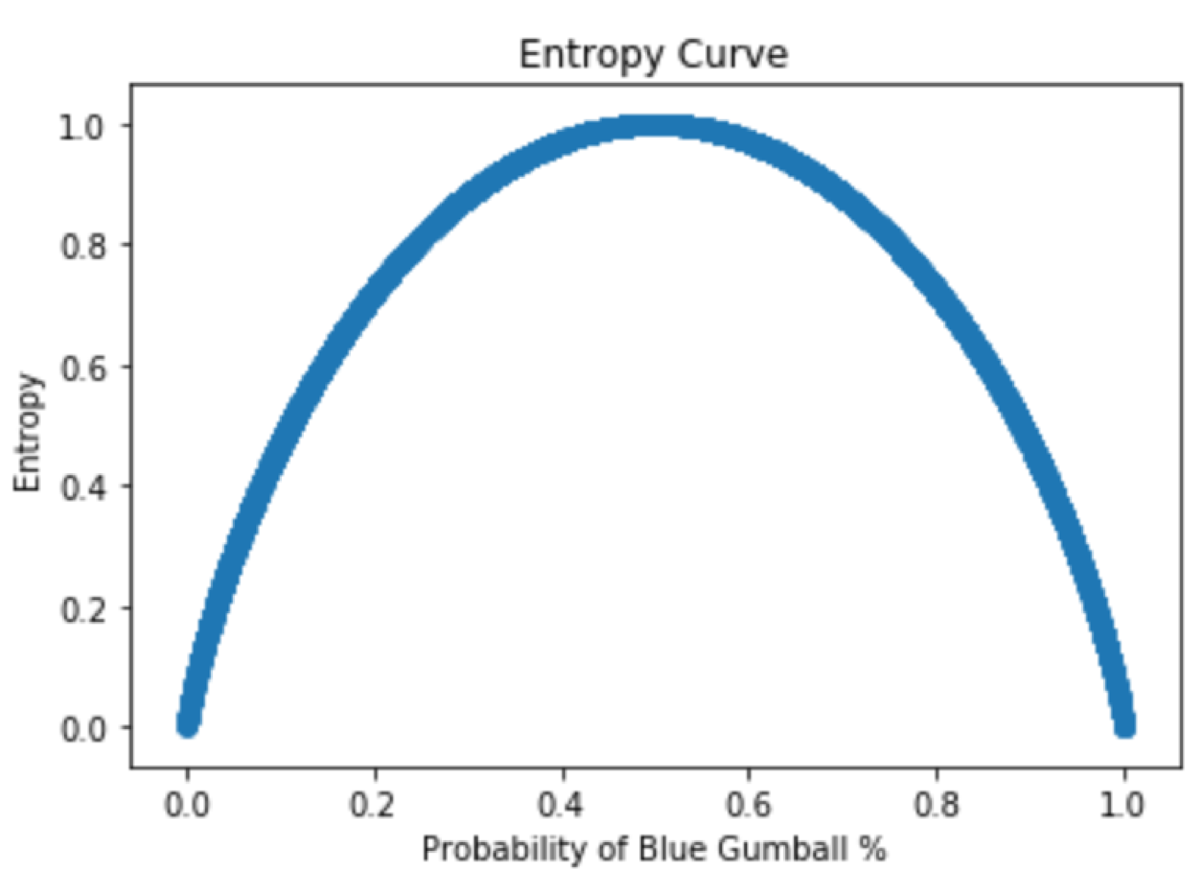
$Gini\ Index = 1\text{-}(probability\_red^2 + probability\_blue^2) = 1\text{-}(0.5^2 + 0.5^2) = 0.5$

4 red and 0 blue:

$Gini\ Index = 1\text{-}(probability\_red^2 + probability\_blue^2) = 1\text{-}(1^2 + 0^2) = 0$

Gini Index vs. Entropy





Decision Tree in Sklearn

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entro
for the information gain.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best sp
and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all lea
contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minin number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. T may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimu number of samples for each node.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Pruning Approaches

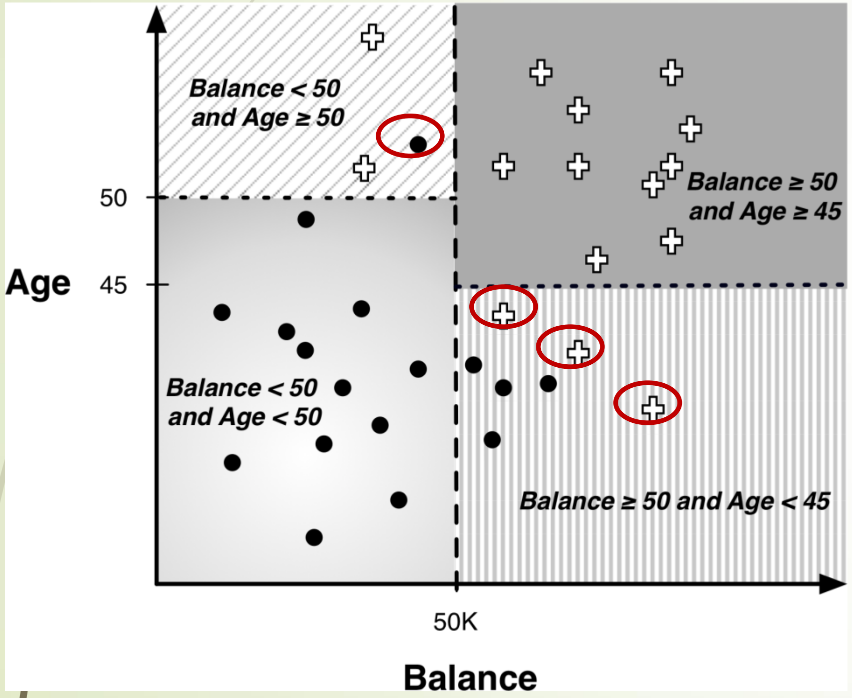
- **Pre-pruning (Early stopping):** stop growing the tree earlier
  - ✓ Minimum number(proportion) of samples required at a leaf node
  - ✓ The maximum depth
  - ✓ Minimum number of samples required to split an internal node
  - ✓ The number of features to consider when looking for the best split…
- **Post-pruning:** allows the tree to fully grow first, and then post prune it
  - ✓ **Reduced error pruning**
  - ✓ **Minimal** Cost-Complexity Pruning (Sklearn)

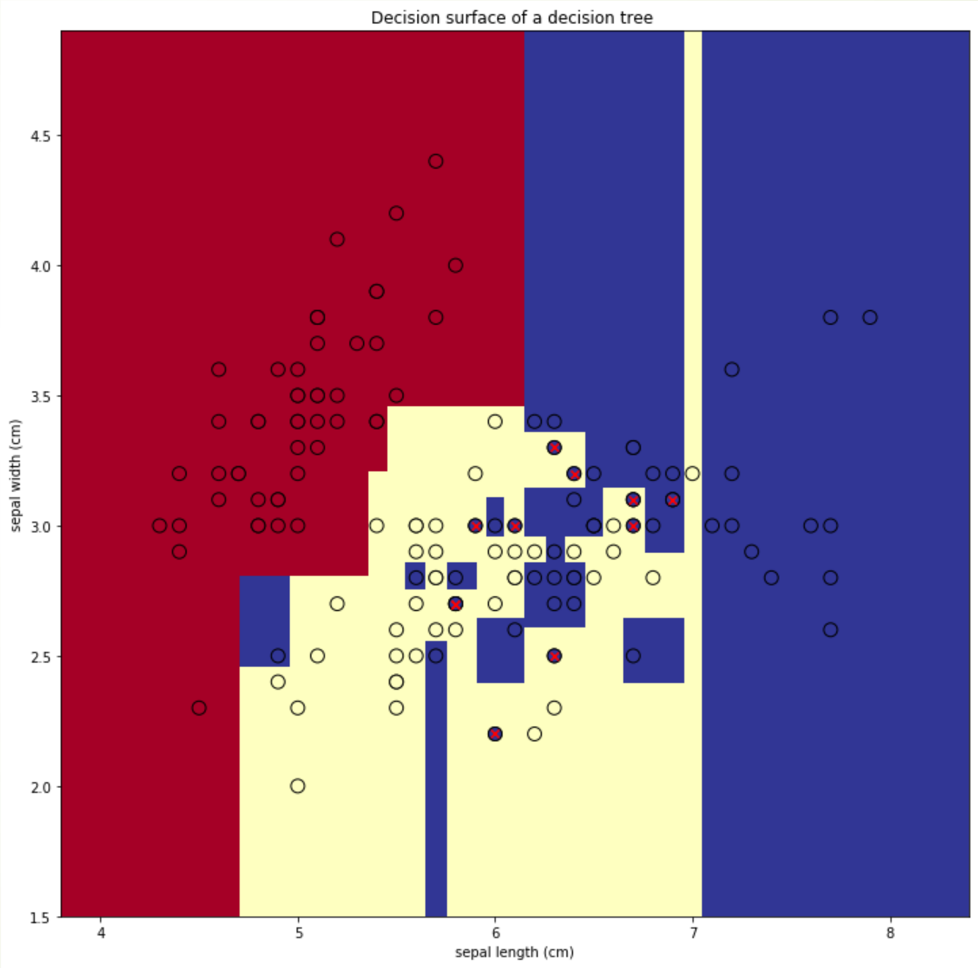
$$R_\alpha(T) = R(T) + \alpha|T|$$

where $|T|$ is the number of terminal nodes in T and

total sample weighted impurity of the terminal nodes for $R(T)$.

Errors in Decision Trees





Outline

- Decision Tree (Cont'd)
- Loss Function & Linear Regression
- Logistic Regression & SVM (Intuition)
- Quiz

Predictive Model as Formula and Parameters

- A predictive model can be viewed as a formula f to **estimate** the value of target y given the feature vector $\mathbf{x}=(x_1, x_2, \ldots, x_n)$, i.e., $\hat{y} = f(\mathbf{x})$, which can be
  - ✓ A set of rules (decision tree)
  - ✓ A mathematical function (what we learn today)
  - ✓ Neural networks …
- Ideally, the estimated target $\hat{y}$ should be close to/match the true target y
  - For classification, we usually refer as **predicted label** and **true label**
- Predictive models can be controlled by model "parameters"
  - ✓ **Tree structure, split points** for decision tree
  - ✓ **Coefficients** (w_0, w_1, w_2…) for linear functions, such as $f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots$

Fitting A Model to Data

- Model deduction/training is to find the best predictive model on the training data, i.e., fitting the labeled data
  - ✓ Training data is a collection of [($\mathbf{x}_1$,y_1), ($\mathbf{x}_2$,y_2),…], and y_1, y_2 are true labels for $\mathbf{x}_1$, $\mathbf{x}_2$ ...
  - ✓ Given a model f, we can then estimate $\hat{y}_1 = f(\mathbf{x}_1)$, $\hat{y}_2 = f(\mathbf{x}_2)$ … for $\mathbf{x}_1$, $\mathbf{x}_2$ …
  - ✓ The objective of fitting a **classification** model to the above training data is to **match** the estimated $\hat{y}_1$ to y_1, $\hat{y}_2$ to y_2 as many as possible
  - ✓ The objective of fitting a **regression** model to the above training data is to **minimize** the difference between $\hat{y}_1$ and y_1 , $\hat{y}_2$ and y_2 …
  - ✓ Achieve above objectives by choosing optimal model "parameters"

Model fitting is to find "best/optimal" model "parameters" to achieve a given objective (minimize mismatch/difference of predictions) on training data

Loss/error/cost/penalty Function

- A loss/error/cost/penalty function determines how much penalty should be assigned to an instance based on the error in the model's predicted value
- Loss function for regression model *f()* on one training instance ($\mathbf{x}_i$, y_i) :
  - ✓ Squared loss: $\text{loss}(f, (\mathbf{x}_i, y_i)) = (f(\mathbf{x}) - y)^2 = (\hat{y} - y)^2$
  - ✓ Absolute loss: $\text{loss}(f, (\mathbf{x}_i, y_i)) = |f(\mathbf{x}) - y| = |\hat{y} - y|$
- Loss function for classification model *f()* on one training instance ($\mathbf{x}_i$, y_i) :
  - ✓ Zero-one loss : if $f(\mathbf{x}_i) == y_i$, *then* $\text{loss}(f, (\mathbf{x}_i, y_i)) = 0$; otherwise $\text{loss}(f, (\mathbf{x}_i, y_i)) = 1$
  - ✓ Logistic loss (Logistic Regression)
  - ✓ Hinge loss (Support Vector Machine)
- The objective of model fitting is to find the "**best/optimal**" model "parameters" **minimize the total loss** on the whole training data (usually the sum of individual loss)

Decision Tree for Zero-one Loss

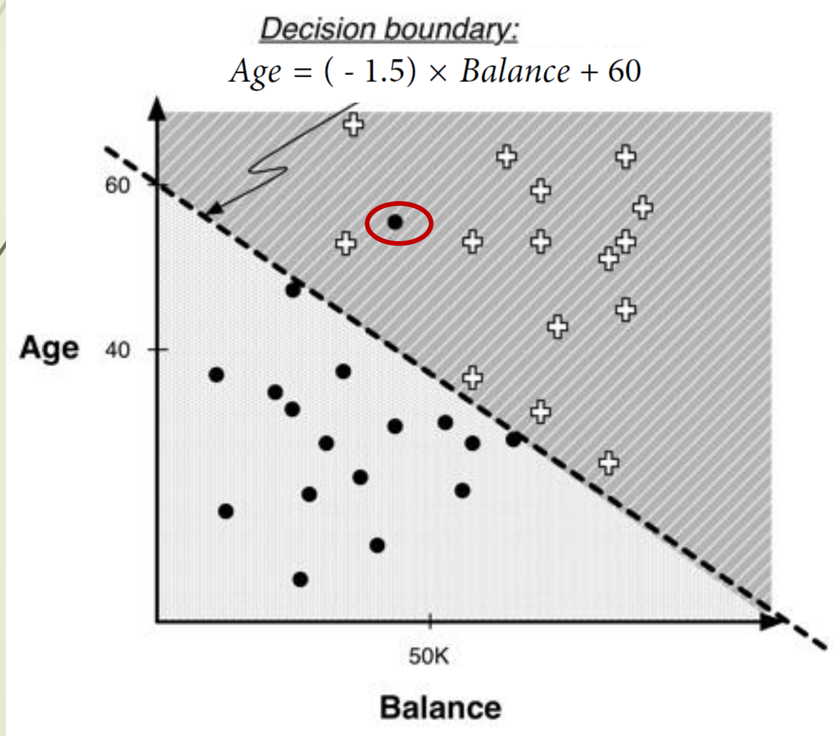
- Decision tree optimize the zero-one loss passively and locally
  - ✓ Optimize the information gain at each segmentation step by step
  - ✓ Top kindergarten-> top primary school -> top middle school -> …???

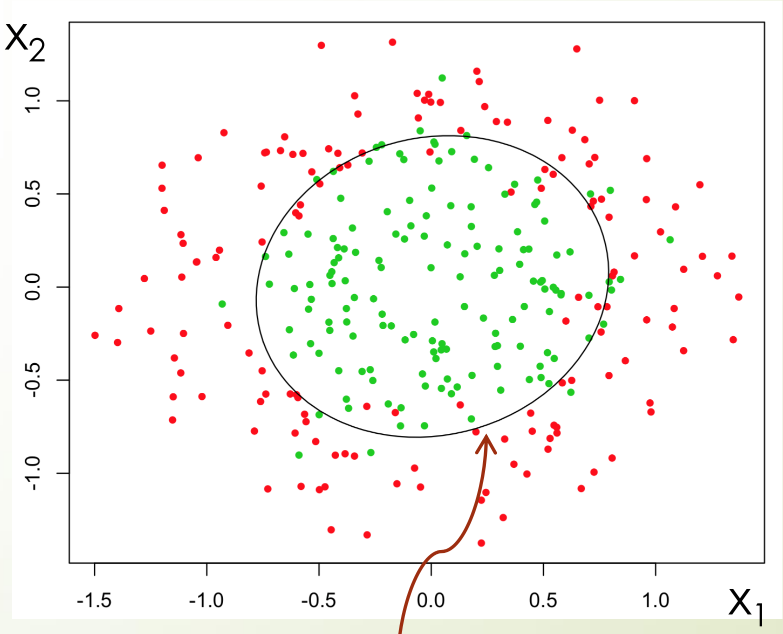
Total loss = $\text{loss}(f,(\mathbf{x}_1,y_1))+\text{loss}(f,(\mathbf{x}_2,y_2))+\ \ldots$

$= 4$

Decision Boundary for Zero-one Loss

- Find the best decision boundary w.r.t zero-one loss directly and globally
  - ✓ Decision boundary can be a curve/straight line, curved surface/ plane





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

Total loss = $\text{loss}(f,(\mathbf{x_1},y_1))+\text{loss}(f,(\mathbf{x_2},y_2))+\ \ldots\ = 1$

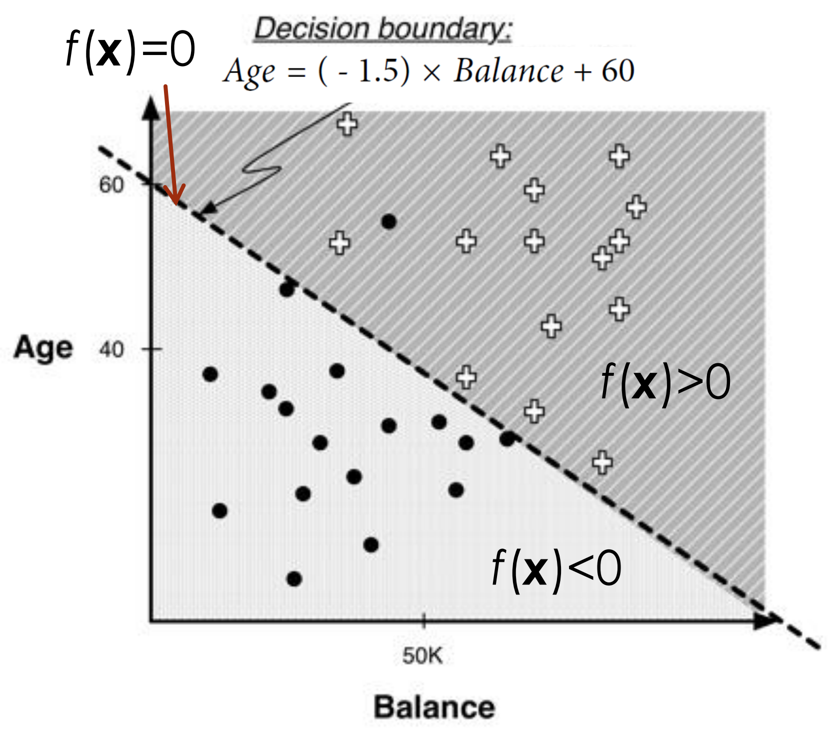
Linear Discrimination Functions

- We currently focus on those decision boundaries which can be represented as linear functions (linear discrimination functions), as $f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \cdots$
  - ✓ Line in two dimensions is $y = mx + b$, e.g., $Age = (-1.5) \times Balance + 60$
  - ✓ We can easily get the discrimination function by mathematical transformation

    $$f(x) = 1.0 \times Age + 1.5 \times Balance - 60$$

  - ✓ We would classify an instance x as **+** if it is above (top-right, $f(\mathbf{x})>0$) the line, and as a • if it is below (bottom –left, $f(\mathbf{x})<0$) the line.

$$class(\mathbf{x}) = \begin{cases} + \text{ if } 1.0 \times Age - 1.5 \times Balance + 60 > 0 \\ \bullet \text{ if } 1.0 \times Age - 1.5 \times Balance + 60 \leq 0 \end{cases}$$



Distance from Decision Boundary

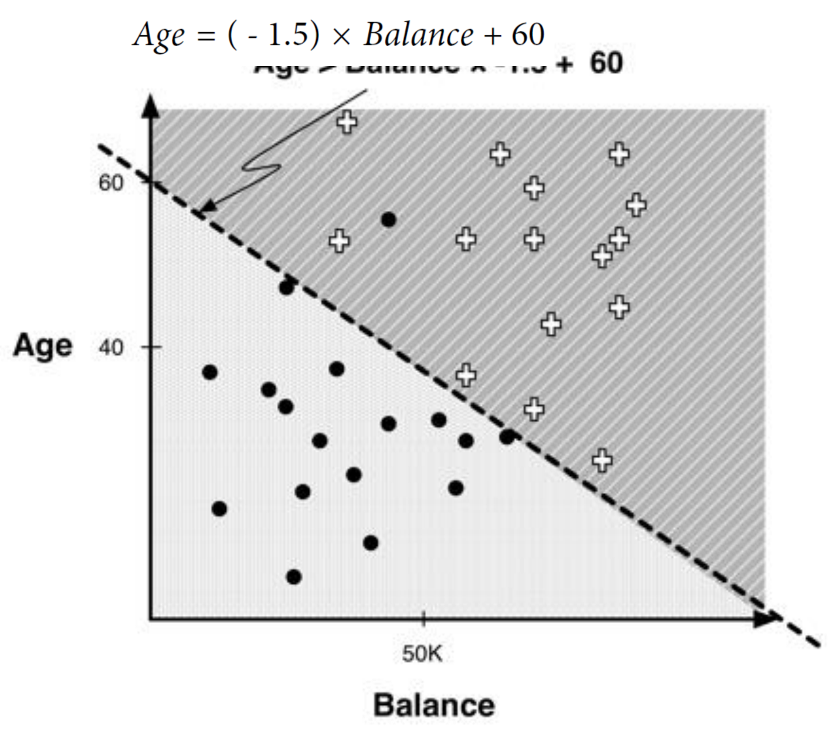
- For a boundary/line defined equation $ax + by + c = 0$, where a, b and c are constants (a and b not both zero), the distance from the line to a point (x_0, y_0) is

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

$f(\mathbf{x_0}) = \hat{y}_0$

Fixed for the line

$$f(x) = 1.0 \times Age + 1.5 \times Balance - 60$$



- For a boundary/line defined by $f(\mathbf{x}) = 0$, then the distance of training sample $\mathbf{x}_i$ to the boundary is proportional to $|f(\mathbf{x}_i)|$ or $|\hat{y}_i|$
- When a sample near the decision boundary we would be most uncertain about its class
- When it is far away from the decision (with larger $|f(\mathbf{x}_i)|$), we would expect the highest likelihood of predicted class based on sign of $f(\mathbf{x}_i)$

Simplifying Assumptions

- To keep the discussion focused, and to avoid excessive footnotes, we make following simplifying assumptions:
  - ✓ For classification and class probability estimation, we will consider only **binary** classes. In particular the label should be **either 1 or -1**
  - ✓ We assume that all attributes are **numerical and well normalized**.
  - ✓ We can rewrite $f(x) = w_0 + w_1 x_1^i + w_2 x_2^i + \ldots = \mathbf{w}^T \cdot \mathbf{x}$, where ***w***=*(w$_0$, w$_1$, w$_2$ …), and* **x** = =*(x$_0$, x$_1$, x$_2$ …)*

Zero-one loss :

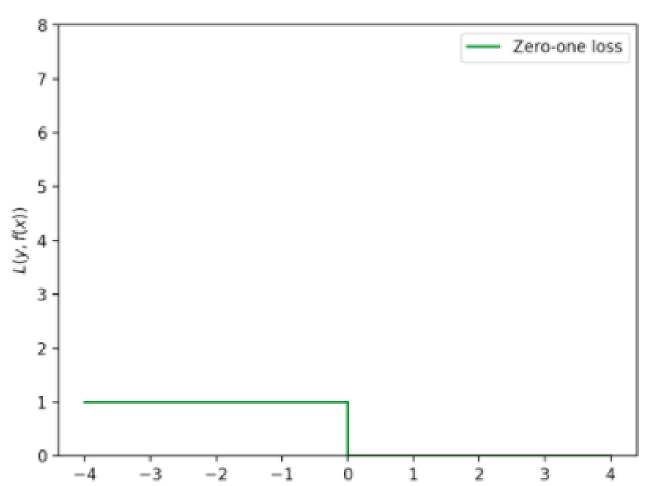
if $f(\mathbf{x}_i) == y_i$ *then* loss$(f,(\mathbf{x}_i, y_i)) = 0$,

otherwise, loss$(f,(\mathbf{x}_i, y_i)) = 1$

⇕

if $f(\mathbf{x}_i) \cdot y_i > 0$, then loss$(f,(\mathbf{x}_i, y_i)) = 0$,

if $f(\mathbf{x}_i) \cdot y_i < 0$ then loss$(f,(\mathbf{x}_i, y_i)) = 1$



f(x) ·*y*

Optimizing an Objective Function

- Creating an objective function that matches the true goal of the data mining is usually impossible but ultimately essential
- Given the predefined loss function in terms of model parameters, we are trying find the optimal weights that achieve the minimum total loss on training data
  - ✓ For objective function of minimizing total loss for linear models $f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \cdots$

    total loss = loss(f,($\mathbf{x_1}$,y_1))+loss(f,($\mathbf{x_2}$,y_2))+ …

    = loss($\mathbf{w}$,($\mathbf{x_1}$,y_1))+loss($\mathbf{w}$,($\mathbf{x_2}$,y_2))+ … , where $\mathbf{w}$=(w_0, w_1 , w_2 …)

    = *TrainLoss*($\mathbf{w}$)

Fitting linear model to training data is to find the best ***w*** that can have least TrainLoss(w) by solving following optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} \text{TrainLoss}(\mathbf{w})$$

2-D Linear Regression (Example)

w$=(b_0, b_1)$,
loss$(f,(\mathbf{x_1},y_1))$ = loss$(\mathbf{w},(\mathbf{x_1},y_1))$ = $(y_1-\mathbf{w}^T\mathbf{x_1})^2$

- We are now trying to predict students' weight (y) according to their height (x) by linear model

$$\hat{y}_i = b_0 + b_1 x_i$$

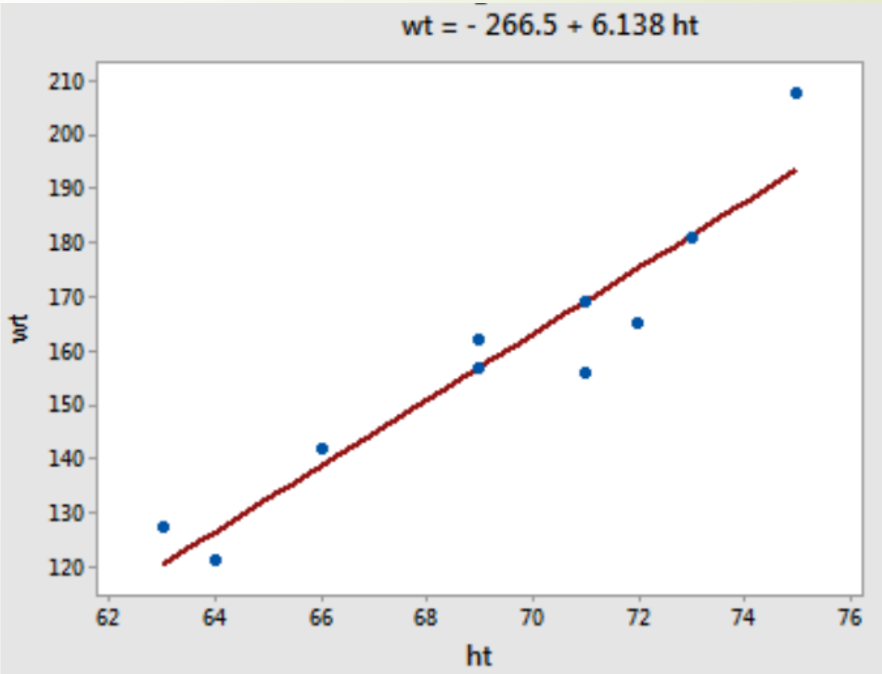
- Then the total training loss on a training data of n instances is

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2$$

- We can easily get a closed-form solution of b_0 *and* b_1 *as follows:*

$$b_0 = \bar{y} - b_1\bar{x}$$

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$



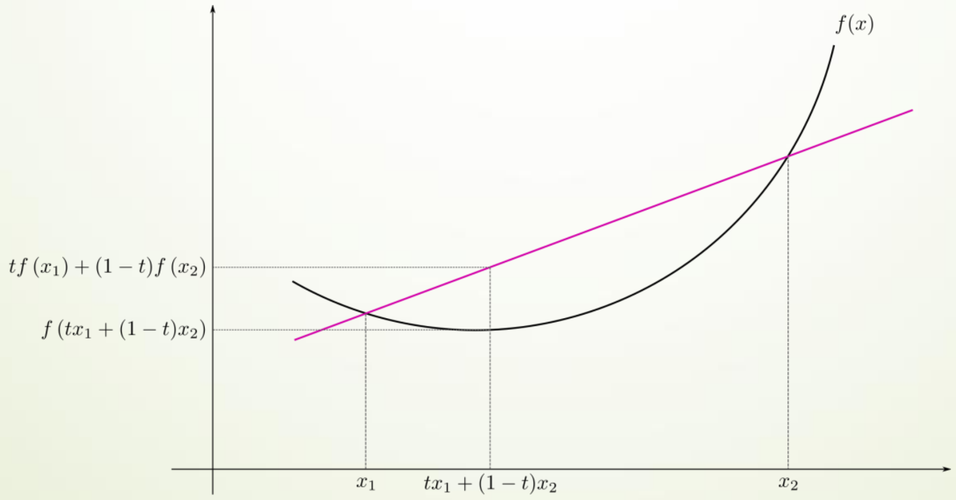
Convex Optimization*

- If *TrainLoss*(x) is a convex function, then the following problem is also convex

$$\min_{\mathbf{w}\in\mathbb{R}^d} \text{TrainLoss}(\mathbf{w})$$

- A function is convex if

$$\forall x_1, x_2 \in X, \forall t \in [0,1]: \qquad f(tx_1 + (1-t)x_2) \leq t f(x_1) + (1-t) f(x_2)$$

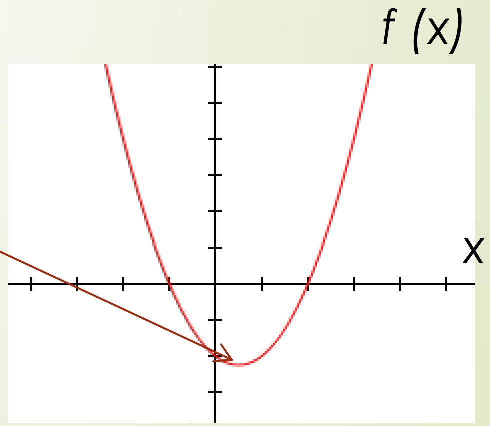


Optimization of Squared Loss*

- Squared loss function is convex

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2$$

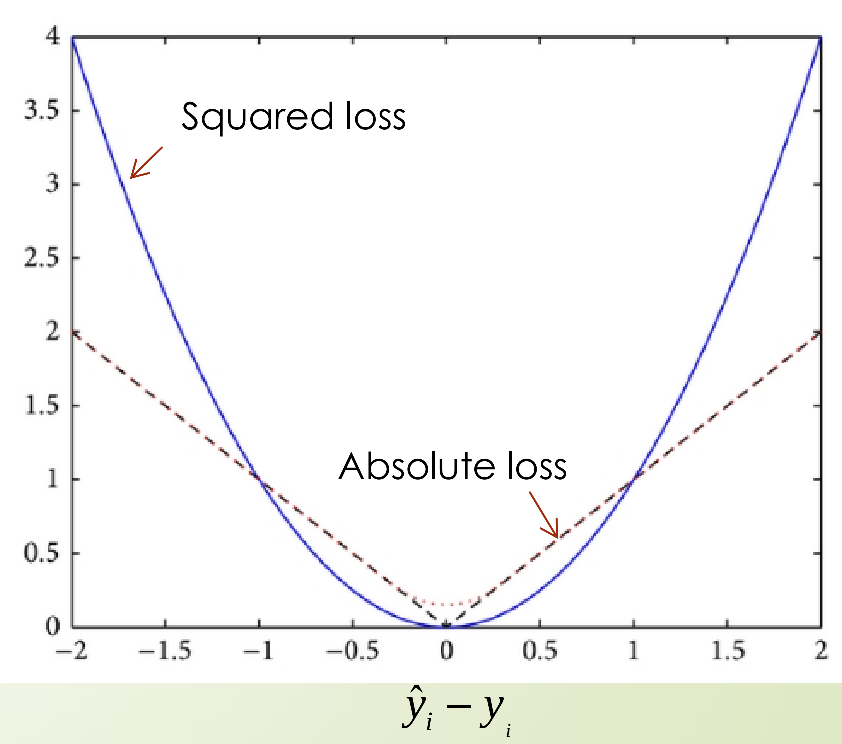
- If *TrainLoss'*(x) = 0, then c is a <u>global minimum</u> of *TrainLoss*(x).
  - ✓ *f(x)=ax^2+bx+c (a>0) is convex, and f'(x) = 2ax+b*
  - ✓ *So by solve f'(x) = 2ax+b = 0,* we can get $x = \frac{-b}{2a}$
  - ✓ *TrainLoss*(x) will be lowest when $x = \frac{-b}{2a}$
  - ✓ When fitting model, **w** is the **x** to tune



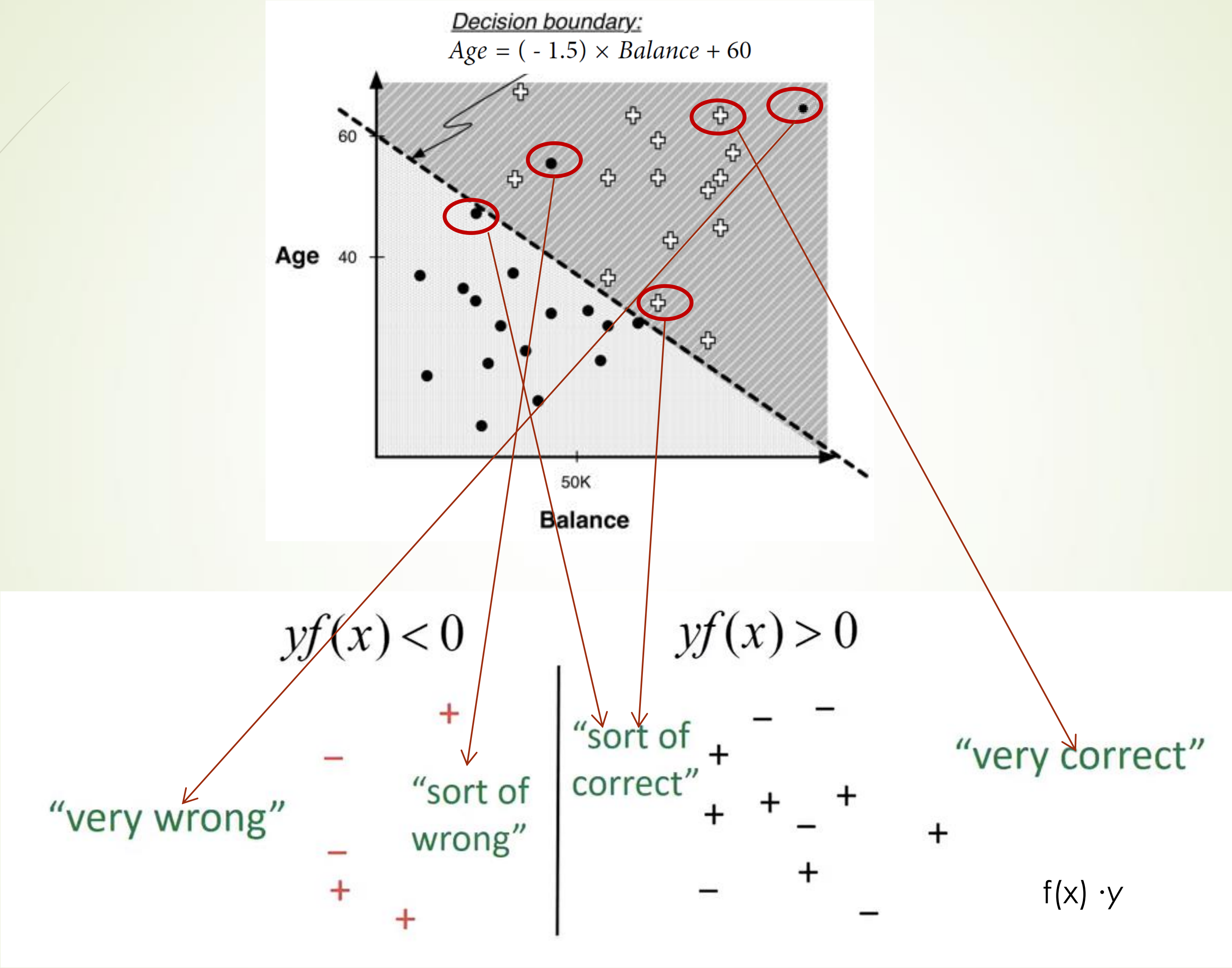
Absolute Loss vs. Squared Loss

- Loss function for regression model *f()* on one training instance ($\boldsymbol{X}_i, y_i$) :
  - ✓ Squared loss: $\text{loss}(f,(\mathbf{x}_i, y_i)) = (f(\mathbf{x}) - y)^2 = (\hat{y} - y)^2$
  - ✓ Absolute loss: $\text{loss}(f,(\mathbf{x}_i, y_i)) = |f(\mathbf{x}) - y| = |\hat{y} - y|$

- Squared loss is preferable because:
  - ✓ It penalize small error less, and strongly penalizes very large errors(but sensitive to outliers)
  - ✓ Easy to decompose and analyze error
  - ✓ It is <u>convex</u> and can provide close-form solution

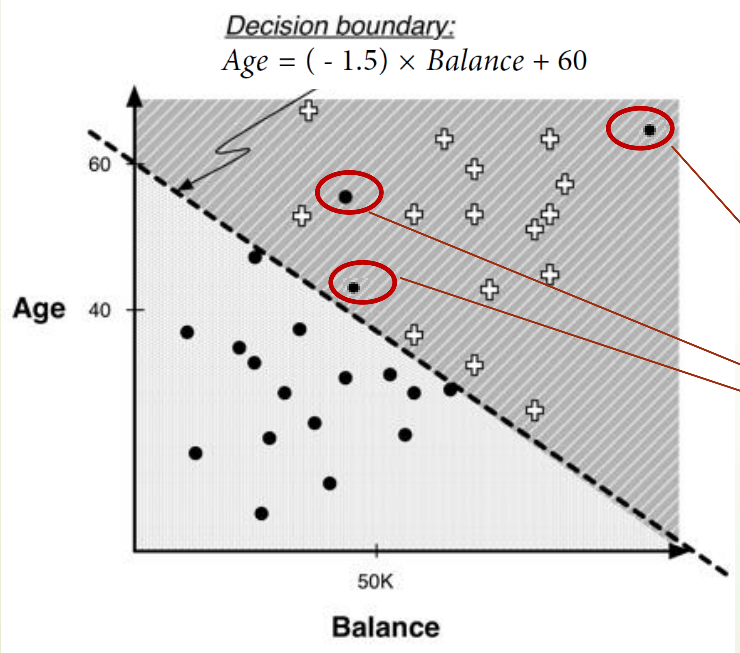


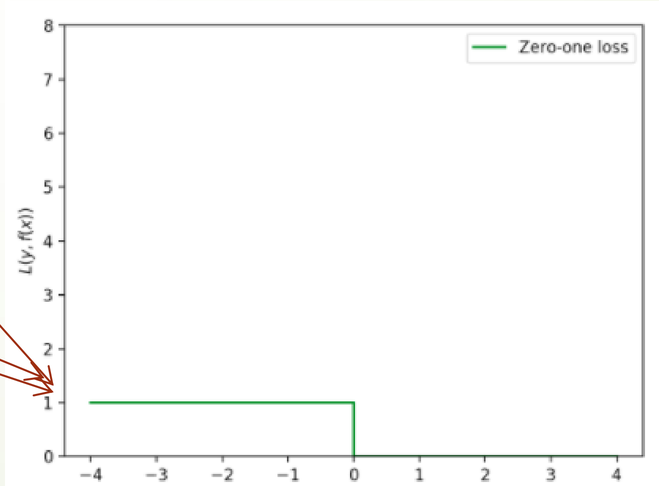
Loss function intuition



Drawbacks of Zero-one Loss

- Zero-one loss is intuitive, but
  - ✓ Hard to optimize: not convex and not continuous(/differentiable)
  - ✓ Do not penalize those severe errors greatly
- If linear separable, we can use Perceptron Learning Algorithm to optimize





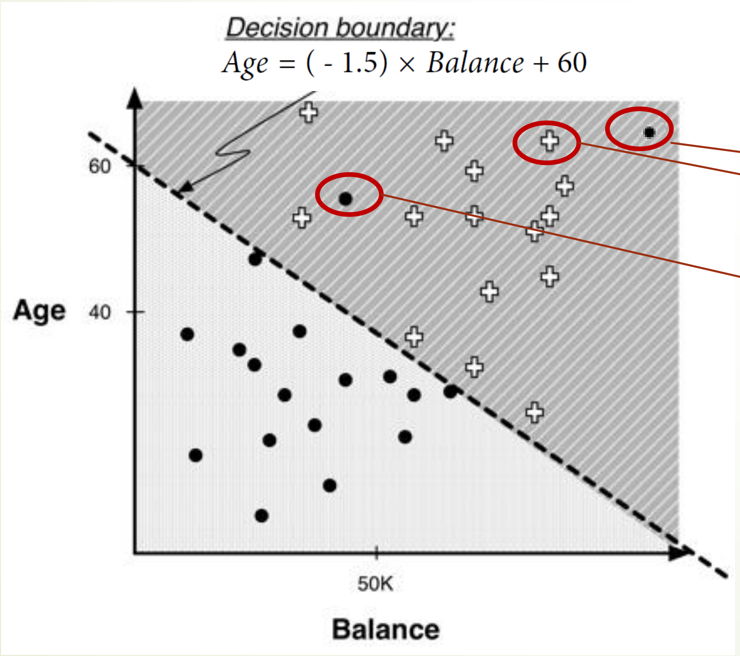
f(x) ·y

Outline

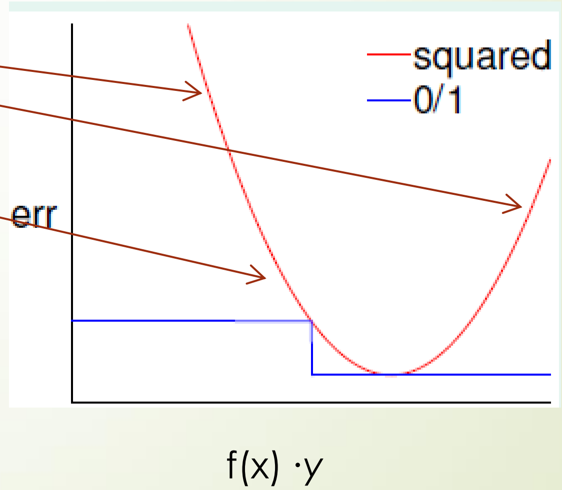
- Decision Tree (Cont'd)
- Loss Function & Linear Regression
- Logistic Regression & SVM (Intuition)
- Quiz

Squared Error for Classification ?

- Squared error is easy to optimize, but it will punish far-way data severely, no matter correct or wrong



$$V(f(\vec{x}), y) = (1 - yf(\vec{x}))^2$$



Logistic "Regression"

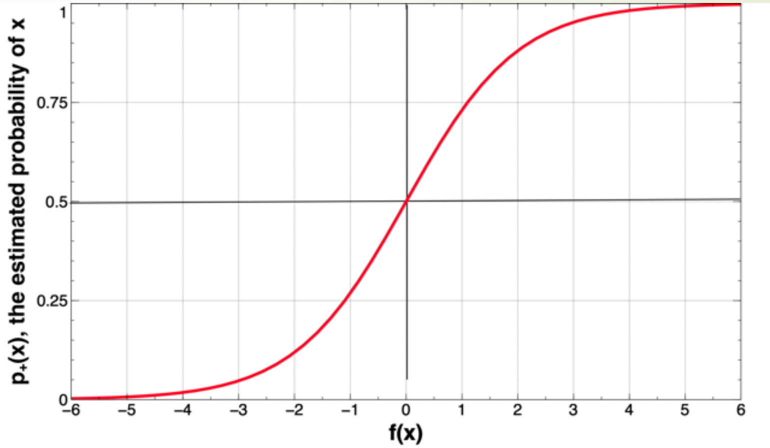
- Name of Logistic "Regression" is a misnomer since it is for classification, and
  - ✓ It addresses the class probability assignment directly, predict $p_+(\mathbf{x})$
  - ✓ It adopts linear function to predict the probability with input feature vector

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots = w_0 \cdot 1 + w_1 x_1 + w_2 x_2 + \ldots = \mathbf{w}^T \cdot \mathbf{x}$$

  - ✓ Probability $p_+(\mathbf{x})$ falls in [0,1], but *f*(**x**) ranges from $-\infty$ to $+\infty$
  - ✓ We can use Logistic function as follow to squeezes the *f*(**x**) into correct range of probabilities $p_+(\mathbf{x})$

$$p_+(\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}}$$

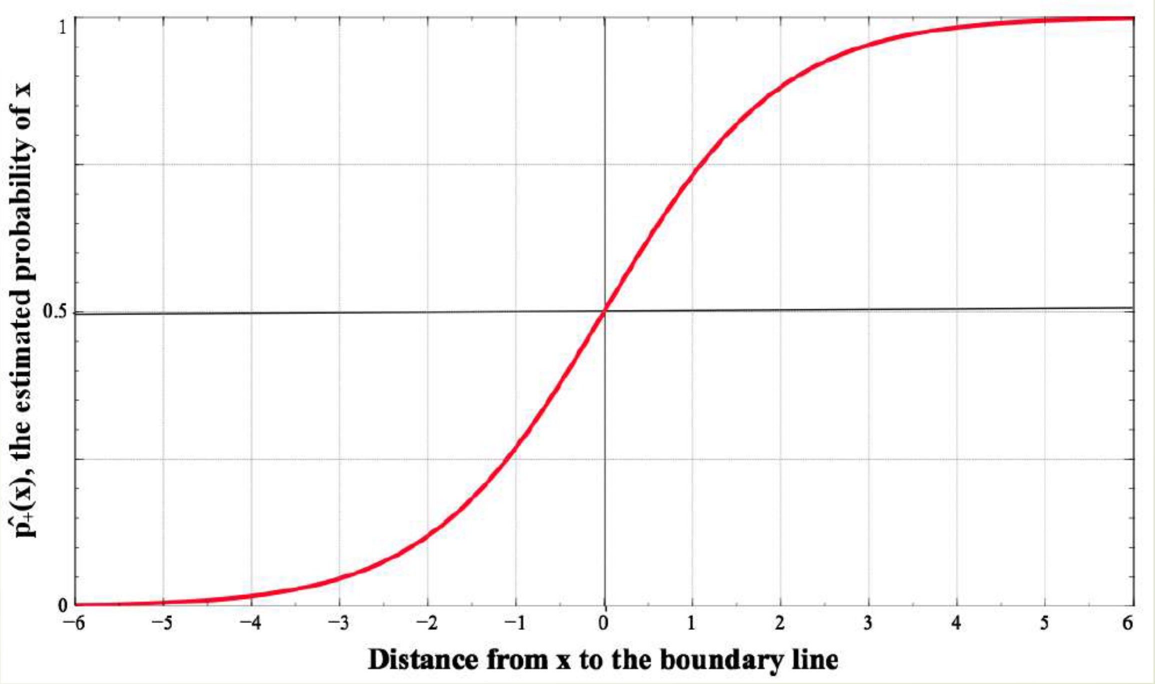
$$p_+(f, \mathbf{x}) = \frac{1}{1 + exp(-f(\mathbf{x}))} \quad \Leftrightarrow \quad p_+(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T \cdot \mathbf{x})}$$



Insights into Logistic Function

- It matches the intuition that we relative certainty in the estimation of class label far from the decision boundary, and uncertainty near the boundary
- Changes of slope also matches our intuition that changes on points near the boundary will get higher changes in the probability
- Logistic function is symmetric

$$\begin{aligned} 1 - p_+(\mathbf{w}, \mathbf{x}) &= 1 - \frac{1}{1 + exp(-\mathbf{w}^T \cdot \mathbf{x}))} \\ &= \frac{exp(-\mathbf{w}^T \cdot \mathbf{x})}{1 + exp(-\mathbf{w}^T \cdot \mathbf{x})} \\ &= \frac{1}{1 + exp(\mathbf{w}^T \cdot \mathbf{x})} = p_+(\mathbf{w}, -\mathbf{x}) \end{aligned}$$



Likelihood of Logistic Regression*

- Following function computing the "likelihood" that a particular labeled example $\mathbf{x_i}$ belongs to the correct class

$$likelihood(\mathbf{w}, \mathbf{x_i}) \begin{cases} p_+(\mathbf{w}, \mathbf{x_i}) & if \quad class_i = +1 \\ 1 - p_+(\mathbf{w}, \mathbf{x_i}) & if \quad class_i = -1 \end{cases}$$

,with $p_+(\mathbf{w}, \mathbf{x}) = \dfrac{1}{1 + exp(-\mathbf{w}^T \cdot \mathbf{x})}$

- Recall that $1 - p_+(\mathbf{w}, \mathbf{x}) = p_+(\mathbf{w}, -\mathbf{x})$, then $likelihood(\mathbf{w}, \mathbf{x_i}) = p_+(\mathbf{w}, y_i x_i)$
- Therefore, the likelihood on all training data would be

$$\begin{aligned} total\_likelihood(\mathbf{w}) &= likelihood(\mathbf{w}, \mathbf{x_1}) \times likelihood(\mathbf{w}, \mathbf{x_2}) \times \ldots \\ &= p_+(\mathbf{w}, y_1 \mathbf{x_1}) \times p_+(\mathbf{w}, y_2 \mathbf{x_2}) \times \ldots \end{aligned}$$

- Then the objective of fitting Logistic model to data is to find ***w*** to maximize the *total_likelihood*, but commonly we maximize its log-likelihood, which is monotone increasing with original likelihood

$$\max_{\mathbf{w}} log(total\_likelihood(\mathbf{w})) = log(p_+(\mathbf{w}, y_1 \mathbf{x_1})) + log(p_+(\mathbf{w}, y_2 \mathbf{x_2})) +$$

Loss Function of Logistic Regression*

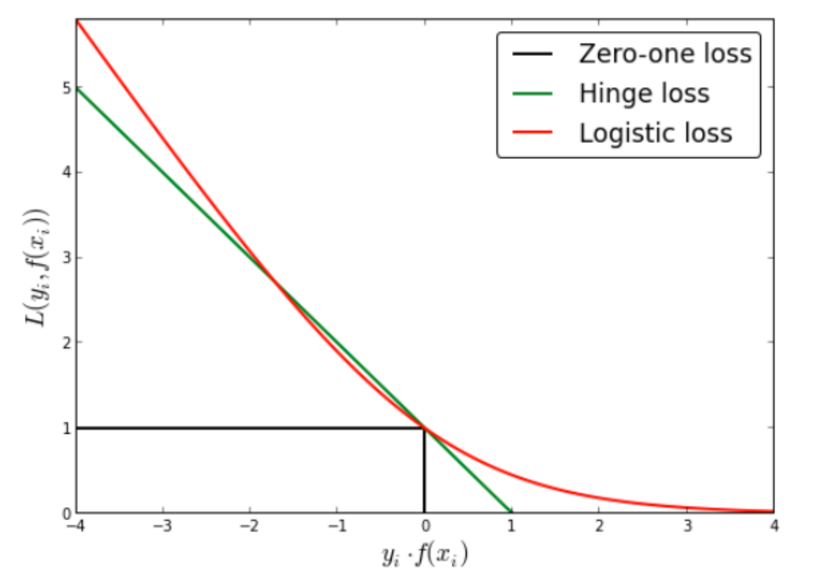
- We could get same optimal **w** by solving $\max_{\mathbf{w}} f(\mathbf{w}) = \min_{\mathbf{w}} -f(\mathbf{w})$
- So by maximize log(total_likelihood), we then minimize -log(total_likelihood)

$$\min_{\mathbf{w}} log(total\_likelihood(\mathbf{w})) = -log(p_+(\mathbf{w}, y_1\mathbf{x_1})) - log(p_+(\mathbf{w}, y_2\mathbf{x_2})) - ..$$
$$= \boxed{log(\frac{1}{p_+(\mathbf{w}, y_1\mathbf{x_1})})} + log(\frac{1}{p_+(\mathbf{w}, y_2\mathbf{x_2})}) + \ldots$$

Loss on sample $(\mathbf{x_1}, y_1)$

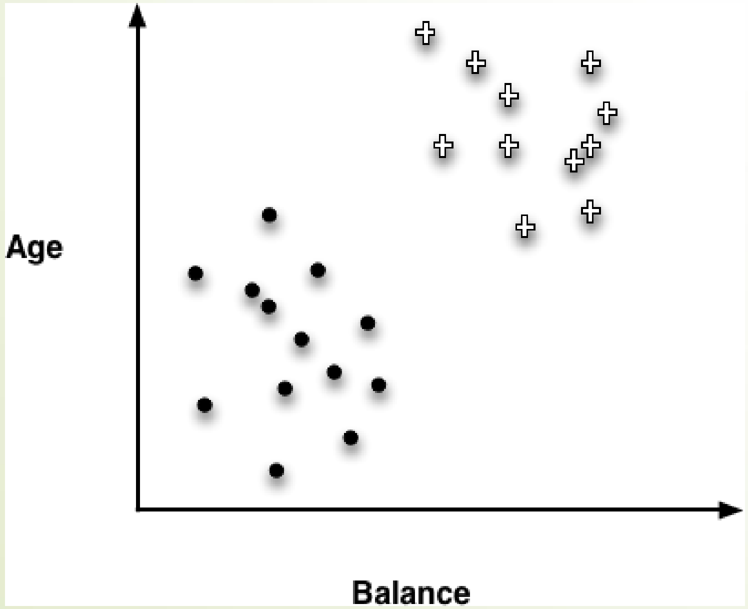
- Recall that $p_+(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T \cdot \mathbf{x})}$ we could know the loss function of Logistic regression is

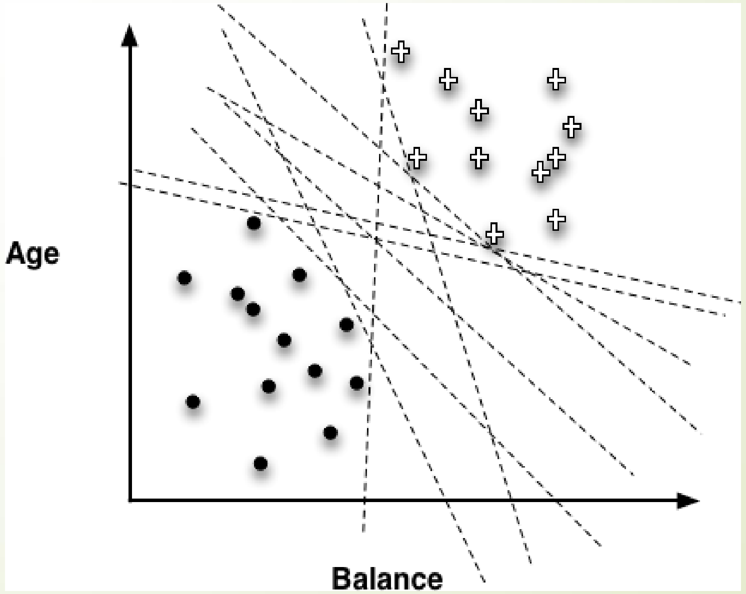
$$loss(f, (\mathbf{x}, y)) = loss(\mathbf{w}, (\mathbf{x}, y))$$
$$= log(1 + exp(-y\mathbf{w}^T \cdot \mathbf{x}))$$
$$= log(1 + exp(-yf(\mathbf{x})))$$



Summary of Logistic Regression

- Logistic regression is easy to optimize (convex and differentiable)
- Logistic regression address the class probabilities directly
- Logistic regression loss penalize those extreme error strongly but hence is relatively sensitive to outliers
- Logistic regression cannot search (or clearly define) the "best boundary"



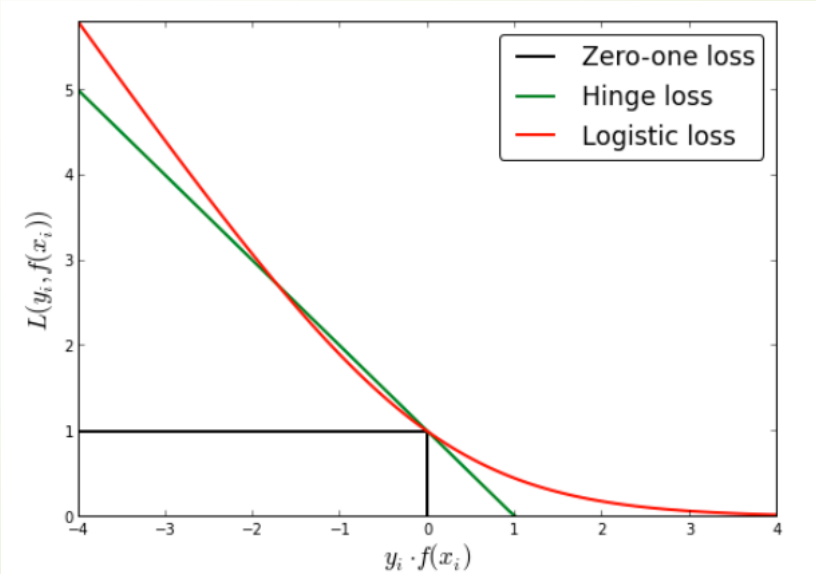


Hinge Loss

- Support Vector Machine (SVM) tries to optimize **hinge loss** and **margin** together

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

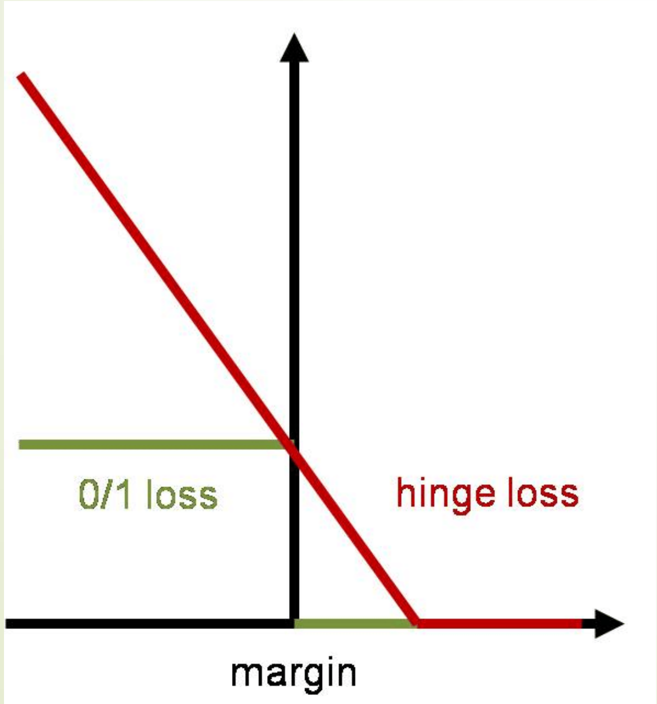
- Hinge loss will also incur penalty for an example that is on the **right side** of the decision boundary but **too close to it**
- The loss is linearly proportional to the distance from the decision boundary on the **wrong side** or on the right side but **too close to** the boundary

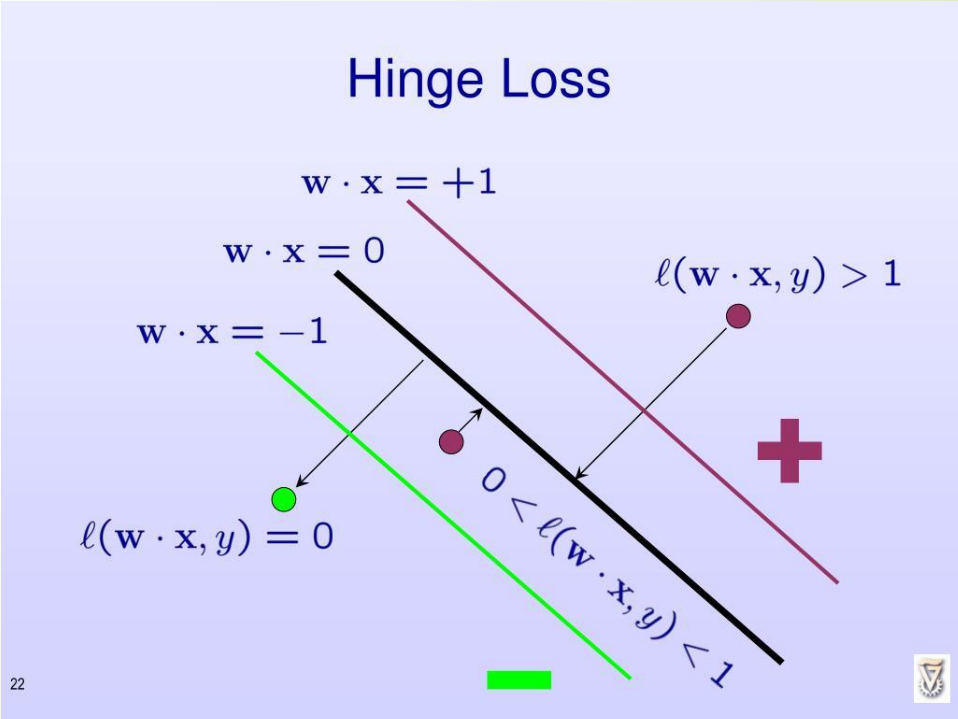


Margin Defined by Hinge Loss

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

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

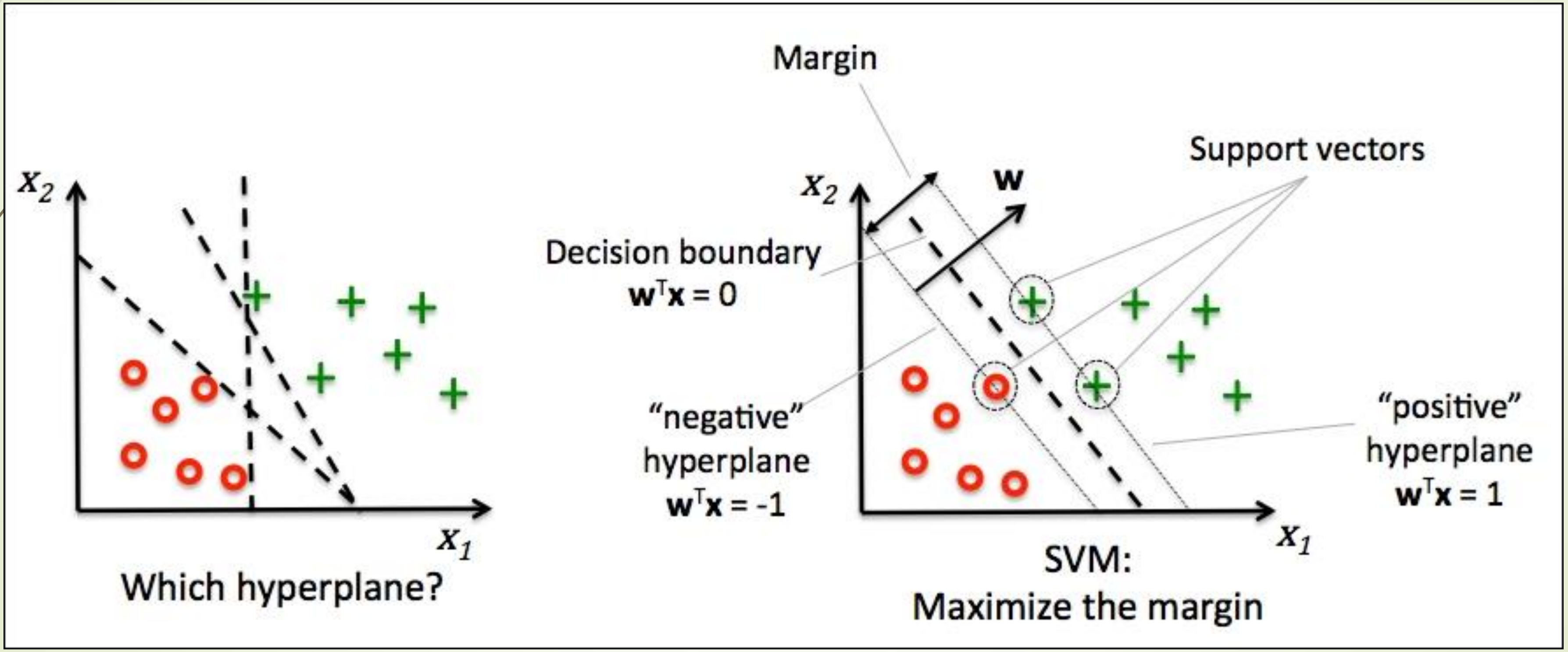




Source: Second Order Learning. Koby Crammer Department of Electrical Engineering. ECML PKDD 2013

Hinge Loss and Clean Margin

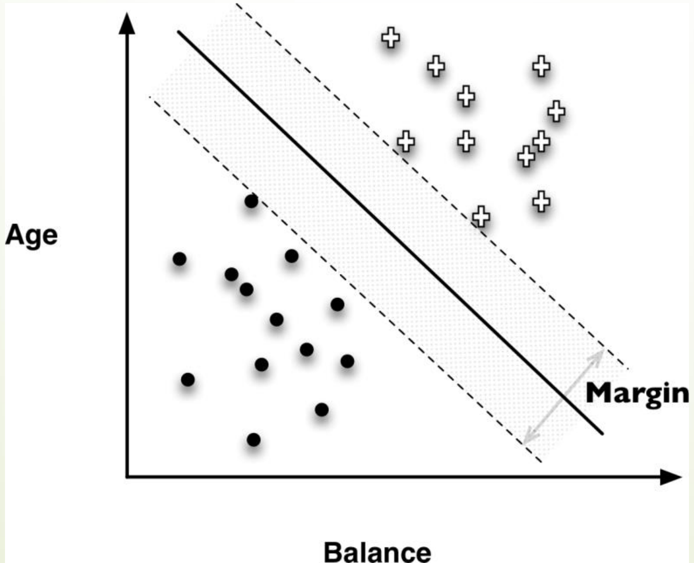
- By minimizing hinge loss is actually trying to find decision boundary with a relative clean margin



Source: Python Machine Learning By Sebastian Raschka

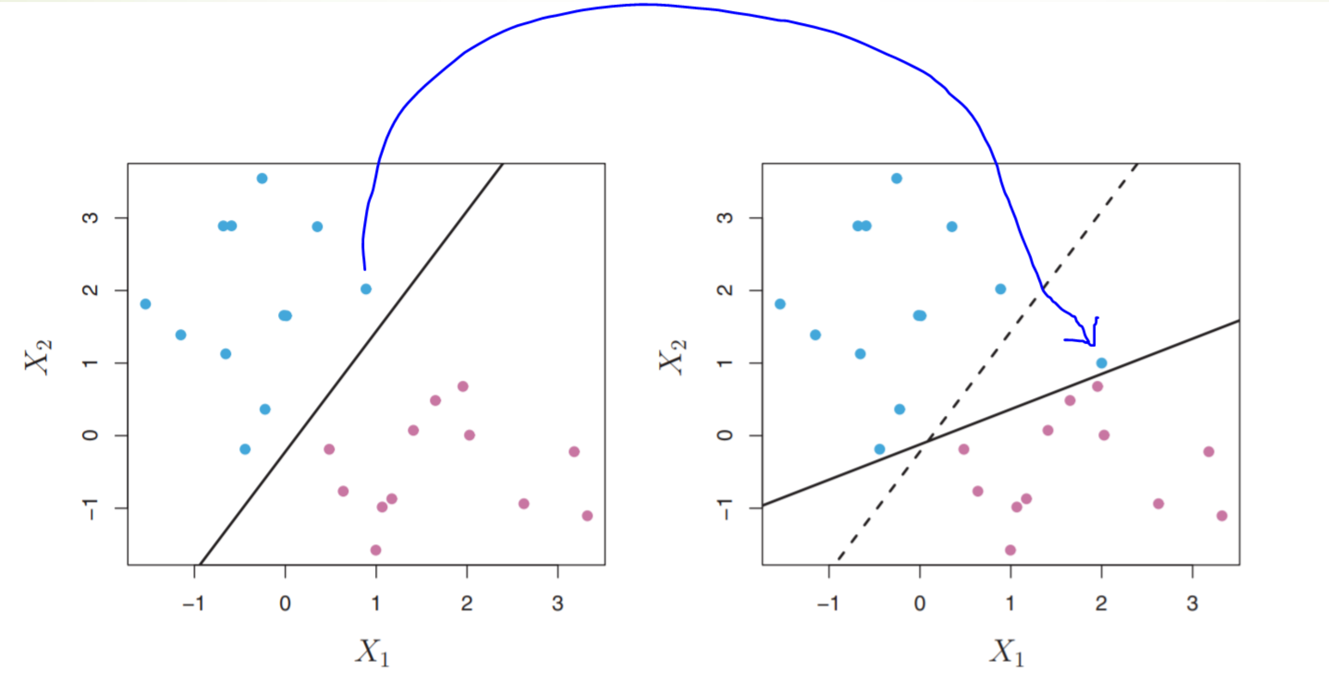
Intuition of Support Vector Machine

- Support Vector Machine (SVM) is a maximal margin classifier with linear discriminant function.
- Objective function based on a simple idea:
  - ✓ Fit the **fattest** bar between classes as much as possible
  - ✓ Once the widest bar is found, the linear discriminant will be the center line through the bar



Intuition of Fat Margin

- We may want a classifier that **misclassify a few** observations in order for:
  - Robustness for individual observation.
  - Better classification of most of the training observation.



Outline

- Decision Tree (Cont'd)
- Loss Function & Linear Regression
- Logistic Regression & SVM (Intuition)
- Quiz

Lab Quiz-5

- **Deadline**: 17:59 p.m., Mar. 13, 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

- Note： **MUST attach ALL** the required files in every submission/resubmission, otherwise other files will be missing.

Quiz Notes

- **Only the answers in answer sheet** will be referred for grading
- Python code is used for verification
  - Whether the code will generate the expected output
  - Whether the code is just **too similar** to some others
- It is **NOT reasonable** to judge the grade based on how many lines of your code are correct even if the final output is wrong or even the code is not runnable