Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Initialization step: for each example x, set

$$D(x)=\frac{1}{N}$$, where N is the number of examples

Iteration step (for t=1…T):

1.Find best weak classifier $h_t(x)$ using weights $D_t(x)$

2. Compute the error rate ε_t as

$$\varepsilon_t = \sum_{i=1}^{N} D(x_i) . I[y_i \neq h_t(x_i)]$$

3. assign weight α_t to classifier $h_t(x)$ in the final hypothesis

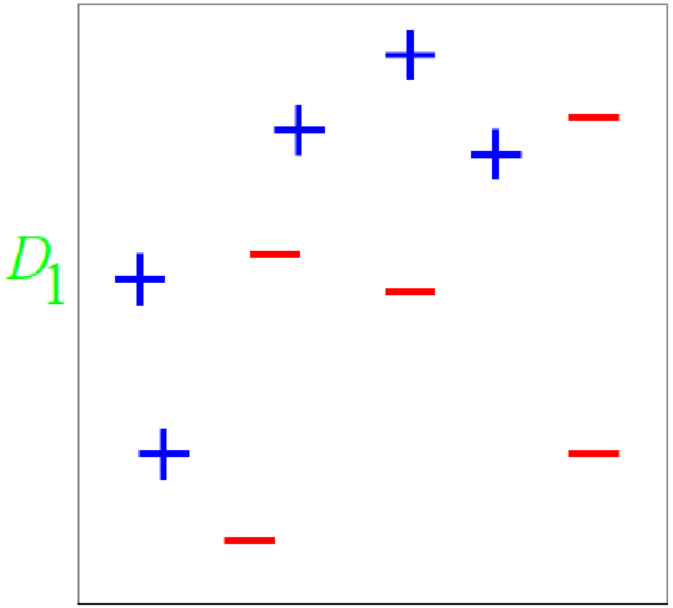
$$\alpha_t = \log((1-\varepsilon_t)/\varepsilon_t)$$

4. For each x_i , $D(x_i) = D(x_i).\exp(\alpha_t . I[y_i \neq h_t(x_i)])$

5.Normalize $D(x_i)$ so that $\sum_{i=1}^{N} D(x_i) = 1$

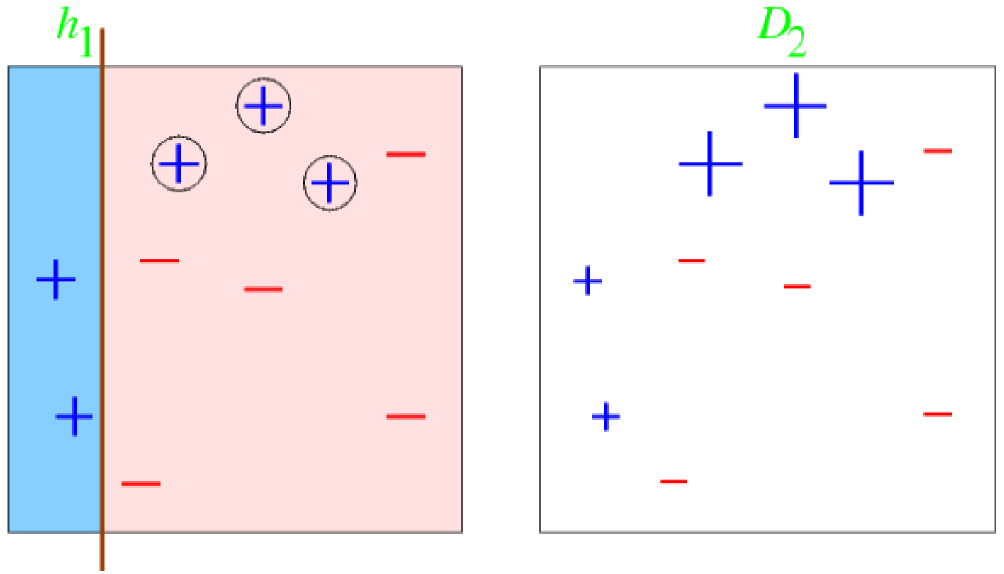
$$f_{final}(x) = sign\ [\sum \alpha_t\ h_t(x)]$$

Boosting: Illustration



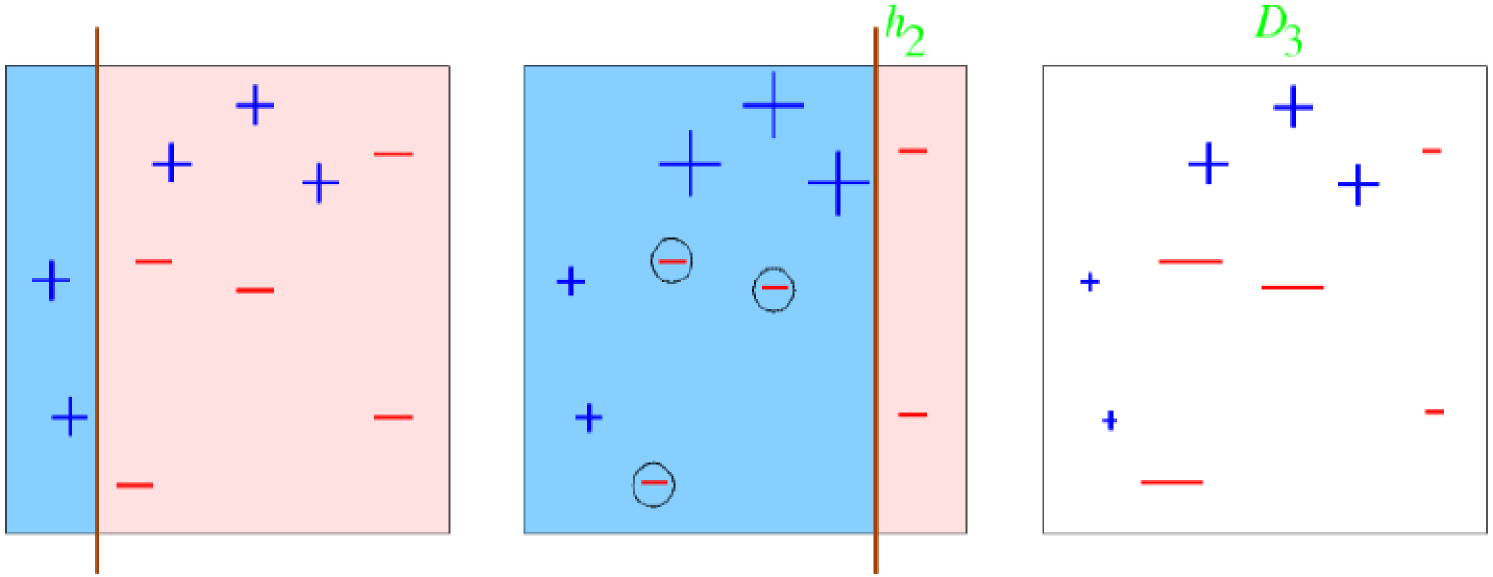
- Consider binary classification with 10 training examples
- Initial weight distribution D1 is uniform (each point has equal weight = 1/10)
- Each of our weak classifiers will be an axis-parallel linear classier

After round 1



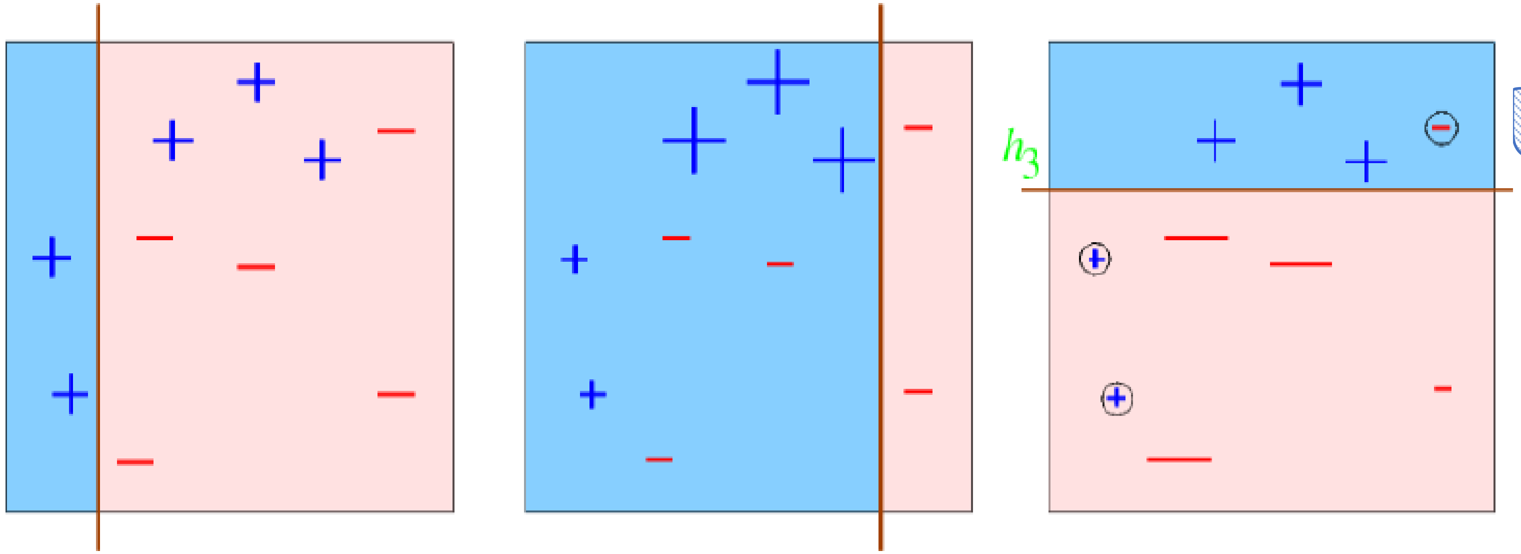
- Error rate of h1: e1 = 0.3; weight of h1: α1 = 0.42
- Each misclassified point upweighted (weight multiplied by exp(α1))
- Each correctly classified point downweighted (weight multiplied by exp(- α1))

After round 2



- Error rate of h2: e1 = 0.21; weight of h1: α1 = 0.65
- Each misclassified point upweighted (weight multiplied by exp(α2))
- Each correctly classified point downweighted (weight multiplied by exp(- α2))

After round 3



- Error rate of h3: e3 = 0.14; weight of h3: α3 = 0.92
- Suppose we decide to stop after round 3
- Our ensemble now consists of 3 classifiers: h1; h2; h3

Ilustrasi: gunakan data bankloan.csv

```
bankloan <- read.csv("D:/bankloan.csv", header=TRUE)
head(bankloan)

library(caret)
set.seed(100)
test_idx <- createDataPartition(bankloan$default, p=0.3, list=FALSE)

cnth_tst <- bankloan[test_idx,]  #membuat data testing
cnth_trn <- bankloan[-test_idx,]  #membuat data training

nrow(cnth_trn) #banyaknya observasi data training
nrow(cnth_tst)

library(ada)
model.boost <- ada(default~.,data=cnth_trn,type="discrete")
prediksi <- sign(predict(model.boost,cnth_tst, type="F"))
prediksi <- ifelse(prediksi==-1, 0, 1)
confusionMatrix(prediksi, cnth_tst$default, positive = "1")

library(randomForest)
model.forest <- randomForest(as.factor(default)~., data=cnth_trn)
pred.forest <- predict(model.forest, cnth_tst)
confusionMatrix(pred.forest, cnth_tst$default, positive = "1")
```

bagusco

Illustration: Binary Classification

- Suppose we want to predict default status of customer based on the following predictors:

age	Age in years
ed	Level of education
employ	Years with current employer
address	Years at current address
income	Household income in thousands
debtinc	Debt to income ratio (x100)
creddebt	Credit card debt in thousands
othdebt	Other debt in thousands

Boosting

```
bankloan <- read.csv("D:/bankloan.csv", header=TRUE)
head(bankloan)

library(caret)
library(caretEnsemble)
bankloan$status[bankloan$default == 1] <- "default"
bankloan$status[bankloan$default == 0] <- "no.default"

set.seed(200)
inTrain <- createDataPartition(y = bankloan$status, p = .75, list = FALSE)
training <- bankloan[ inTrain,]
testing <- bankloan[-inTrain,]

boost.caret <- train(status~ age+ed+employ+address+income+debtinc+creddebt+othdebt,
                     data=training, method='gbm')

boost.caret.pred <- predict(boost.caret, testing)
confusionMatrix(boost.caret.pred, testing$status)
```

bagusco

```
Confusion Matrix and Statistics

            Reference
Prediction   default no.default
  default         21         10
  no.default      24        119

               Accuracy : 0.8046
                 95% CI : (0.7378, 0.8607)
    No Information Rate : 0.7414
    P-Value [Acc > NIR] : 0.03178

                  Kappa : 0.433
 Mcnemar's Test P-Value : 0.02578

            Sensitivity : 0.4667
            Specificity : 0.9225
         Pos Pred Value : 0.6774
         Neg Pred Value : 0.8322
             Prevalence : 0.2586
         Detection Rate : 0.1207
   Detection Prevalence : 0.1782
      Balanced Accuracy : 0.6946

       'Positive' Class : default
```