

Linear Machine Learning

Janis Keuper

Introduction to ML



Basic Types of Machine Learning Algorithms

Supervised Learning

Unsupervised Learning

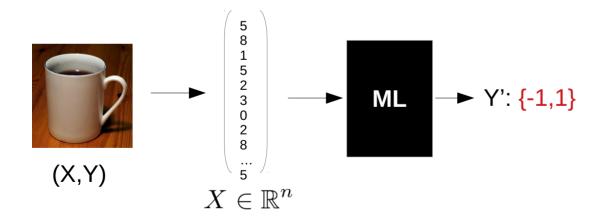
Reinforcement Learning

- Labeled data
- Direct and quantitative evaluation
- Learn model from "ground truth" examples
- Predict unseen examples

Recall Classification



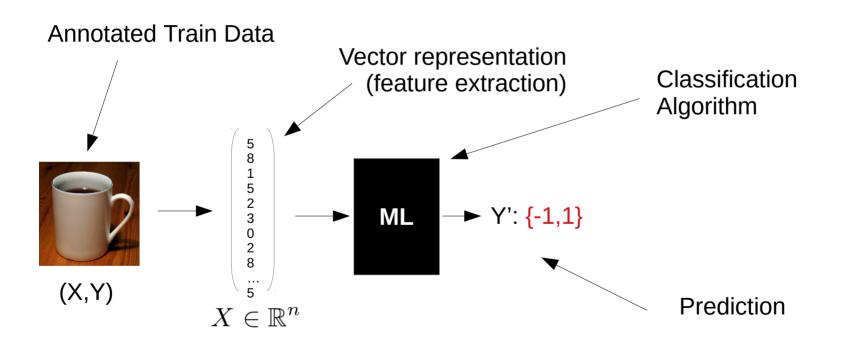
Supervised Learning: Annotated Training Data



Recall Classification



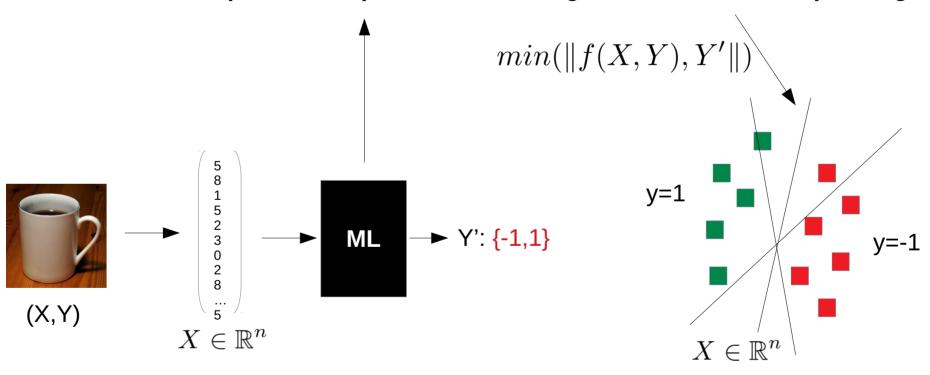
Supervised Learning: Annotated Training Data



Recall Classification



LEARNING: is a optimization problem → Finding the best function separating



Example: MNIST

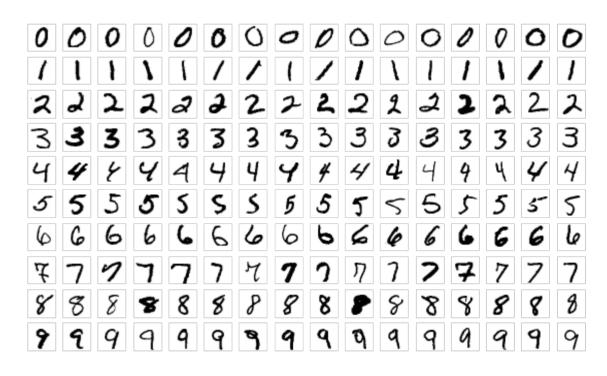


The MNIST hand written digits classification Problem

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

Data specs:

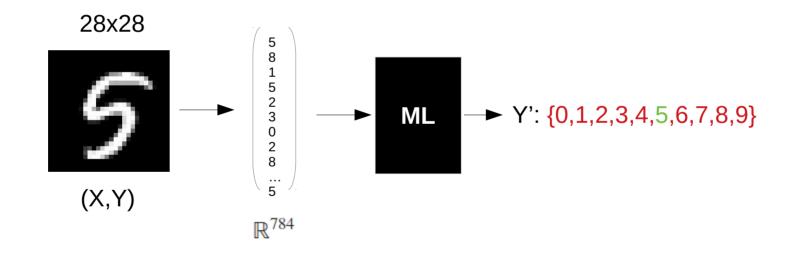
- · 10 Classes (digest 0-9)
- · 28x28 gray scale images
- · 60000 train samples
- · 10000 test samples



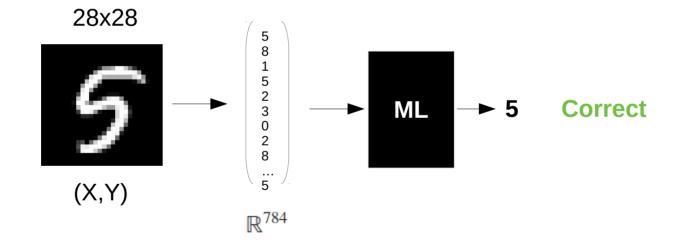
Example: MNIST



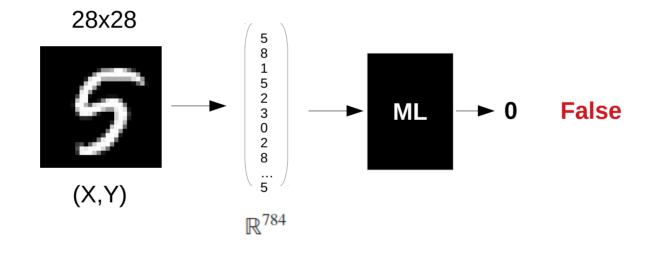
The MNIST hand written digits classification Problem











Recall: Evaluation



Basic evaluation of a model:

Train error: measure of how well the model predicts the given labels

$$Err_{train} := \frac{1}{|X_{train}|} \sum_{x_i \in X_{train}} |f(x_i) - y_i|$$

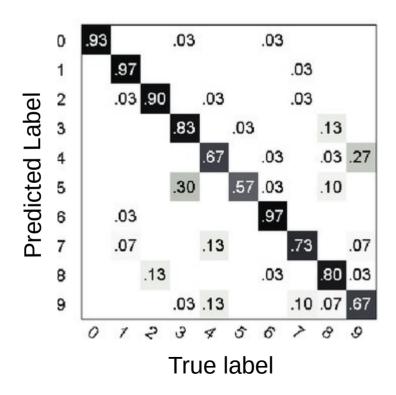
low train error is the necessary condition for a "good" model

Test error: same as train error: low test error is the sufficient condition

$$Err_{test} := \frac{1}{|X_{test}|} \sum_{x_i \in X_{test}} |f(x_i) - y_i|$$



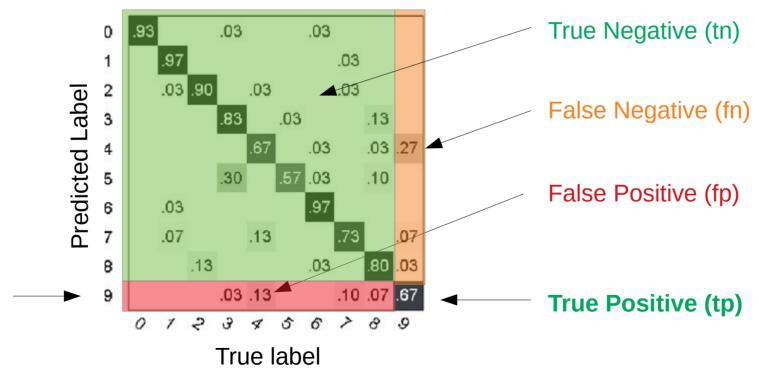
Confusion Matrix and True and False Positives/Negatives





Confusion Matrix and True and False Positives/Negatives

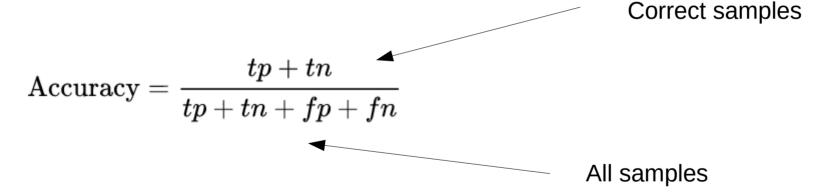
Example for true digit "9"





Accuracy:

Most commonly used error metric:





Problems with Accuracy Unbalanced classes:

If the prior probability of on class is much higher than others, *fp* will have little impact.

$$ext{Accuracy} = rac{tp+tn}{tp+tn+fp+fn}$$

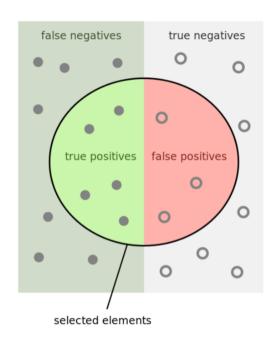
Extreme example: if 90% of the digits are "1", classifying every digit to "1" will will have 90% accuracy!



Precision and Recall

$$ext{Precision} = rac{tp}{tp+fp}$$

$$ext{Recall} = rac{tp}{tp+fn}$$



How many selected items are relevant?

How many relevant items are selected?

[image by wikipedia]



F-Measure or balanced F-score is the harmonic mean of precision and recall:

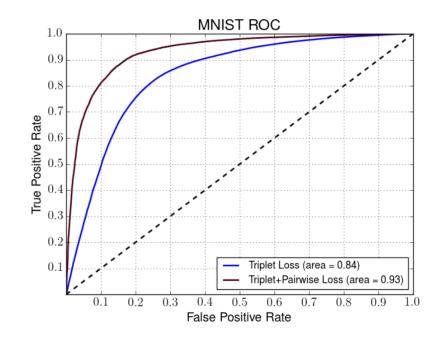
$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



Receiver Operating Characteristic Curve

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a **binary classifier** system as its discrimination threshold is varied.

Example: comparing two different models

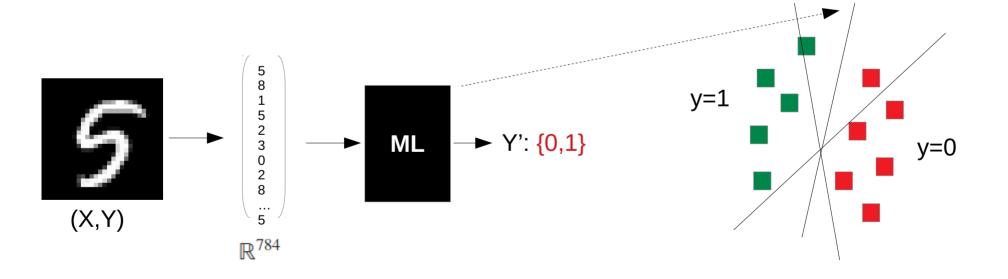




A Simple Linear Model: binary classification

Example: "5" vs "other digit"

Model: hyper plane





A Simple Linear Model: binary classification

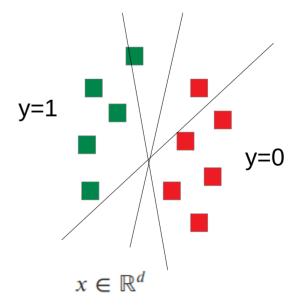
Parameterization of prediction function f with d-dimensional data as:

$$f(x) = y' = w^T x = \sum_{j=0}^d w_j x_j$$

With data samples $x \in \mathbb{R}^d$

Model parameters $w \in \mathbb{R}^d$

Model: hyper plane





A Simple Linear Model: binary classification

Parameterization of prediction function f with d-dimensional data as:

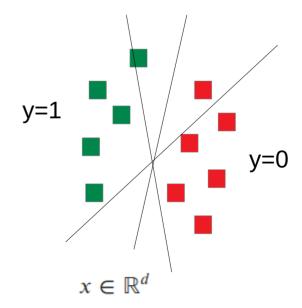
$$f(x) = y' = w^T x = \sum_{j=0}^d w_j x_j$$

With data samples $x \in \mathbb{R}^d$

Model parameters $w \in \mathbb{R}^d$

How to find the parameters?

Model: hyper plane





Optimization problem to find parameters

$$\underset{w}{\arg\min} \sum_{i=0}^{N} L(y_i, w^T x_i)$$

With a differential Loss function like

$$L(y = 1, y') := \frac{1}{1 + e^{-y'}}$$

$$L(y = 0, y') := 1 - L(y = 1, y')$$



Optimization problem to find parameters

$$\underset{w}{\arg\min} \sum_{i=0}^{N} L(y_i, w^T x_i)$$

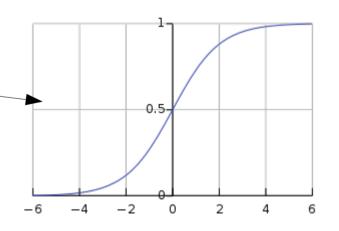
With a differential Loss function like: logistic function

$$L(y = 1, y') := \frac{1}{1 + e^{-y'}}$$

$$L(y = 0, y') := 1 - L(y = 1, y')$$

- pseudo probability: Out put always between 0 and 1
- Apply threshold function on probability that class label =1

Only one of many possible Loss functions, but common choice





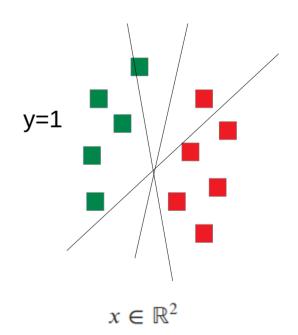
Goal: find w to minimize $\arg \min \sum_{i=0}^{N} L(y_i, w^T x_i)$

$$\underset{w}{\arg\min} \sum_{i=0}^{N} L(y_i, w^T x_i)$$



Goal: find w to minimize $\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

2D Example:



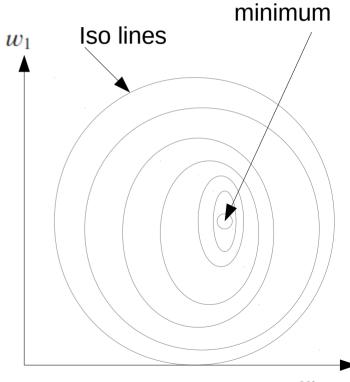
Feature Space



Goal: find w to minimize $\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

2D Example:

- How many (and which) parameters do we have to find?
- *L* spans a (loss) surface in the d-dimensional space of the data X (parameter space)
- We can evaluate L at each point w
- We can compute the gradient at each point w in L (assuming L to be Lipschitz)



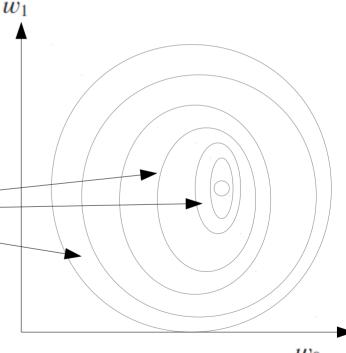
Model Parameter Space



Goal: find w to minimize $\arg \min \sum_{i=0}^{N} L(y_i, w^T x_i)$

2D Example:

- How many (and which) parameters do we have to find?
- L spans a (loss) surface in the d-dimensional space of the data X (parameter space)
- We can evaluate L at each point w
- We can compute the gradient at each point w in L (assuming L to be Lipschitz)



 w_0



Goal: find w to minimize $\underset{w}{\operatorname{arg min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

2D Example:

- How many (and which) parameters do we have to find?
- *L* spans a (loss) surface in the d-dimensional space of the data X (parameter space)
- We can evaluate L at each point w
- We can compute the gradient at each point w in L (assuming L to be Lipschitz) $\nabla L = \frac{dL}{dw}$

 ∇L

 w_1

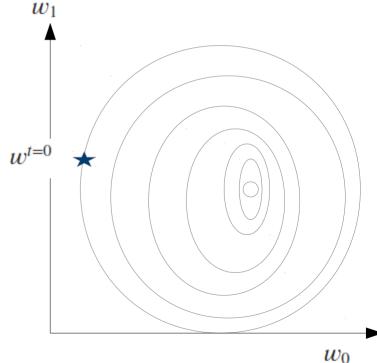
 w_0



Goal: find w to minimize $\arg \min \sum_{i=0}^{N} L(y_i, w^T x_i)$

Gradient Descent Algorithm:

I. Start with random $w^{t=0}$



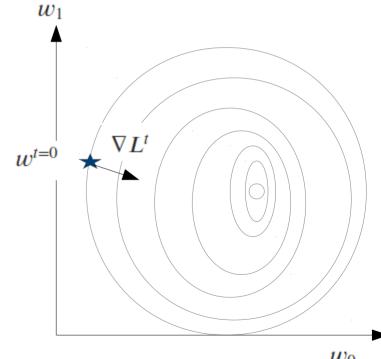


Goal: find w to minimize $\arg \min \sum_{i=0}^{N} L(y_i, w^T x_i)$

Gradient Descent Algorithm:

- I. Start with random $w^{t=0}$
- II. Compute gradient for all training samples

$$\nabla L^t = \sum_{i=0}^{|(X,y)|} \frac{dL(y_i, w^t x_i)}{dw^t}$$





Goal: find w to minimize $\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

Gradient Descent Algorithm:

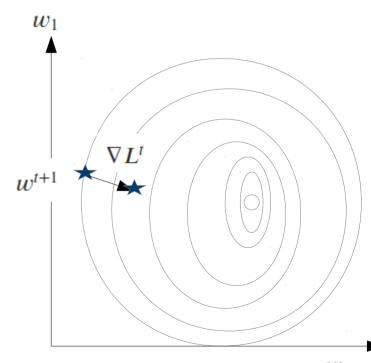
- I. Start with random $w^{t=0}$
- II. Compute gradient for all training samples

$$\nabla L^t = \sum_{i=0}^{|(X,y)|} \frac{dL(y_i, w^t x_i)}{dw^t}$$

III. Update parameters

$$w^{t+1} = w^t + \lambda \nabla L^t$$

Step size or Learning rate
Usually quite small scalar like 0.001
ML Summer School -Janis Keuper





Goal: find w to minimize $\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

Gradient Descent Algorithm:

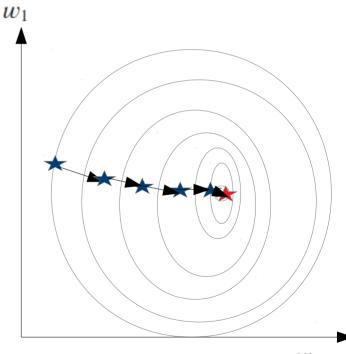
- I. Start with random $w^{t=0}$
- II. Compute gradient for all training samples

$$\nabla L^t = \sum_{i=0}^{|(X,y)|} \frac{dL(y_i, w^t x_i)}{dw^t}$$

III. Update parameters

$$w^{t+1} = w^t + \lambda \nabla L^t$$

IV. Repeat II-III till convergence

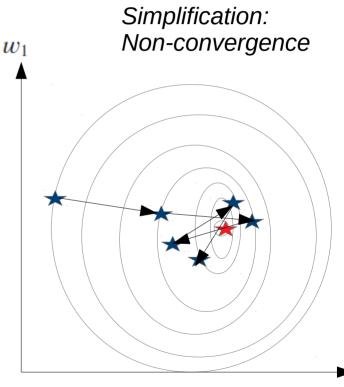




Goal: find w to minimize
$$\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$$

Convergence

I. Theory: need to decrease λ to guarantee convergence to minimum

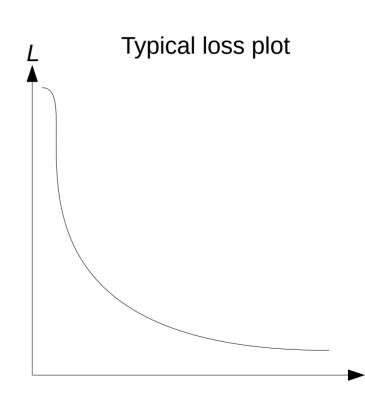




Goal: find w to minimize $\underset{w}{\operatorname{arg \, min}} \sum_{i=0}^{N} L(y_i, w^T x_i)$

Convergence

- I. Theory: need to decrease λ to guarantee convergence to minimum
- II. How to know when to stop?
 - I. Pre set number of iterations
 - II. Loss limit
 - III.Loss not changing



Multi Class Problems



What if we have more than two classes? → simple extension of our model

$$f(x) = y' = argmax(Wx)$$



- → One vector per class
- → Matrix vector Multiplication
- → returns vector with class-wise response
- → argmax selects maximum class label

Multi Class Problems



What if we have more than two classes? → simple extension of our model

$$f(x) = y' = argmax(Wx)$$

Optimization problem is almost the same

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, Wx_i)$$

Change Loss to SOFTMAX function to normalize sum aver all probabilities to one

$$L(y^{i}, y^{i'}) := \frac{e^{y^{i'}}}{\sum_{i}^{k} e^{y^{j'}}}$$

Use "one-hot" coding of y

Multi Class Problems



What if we have more than two classes? → simple extension of our model

$$f(x) = y' = argmax(Wx)$$

Optimization problem is almost the same

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, Wx_i)$$

Change Loss to SOFTMAX function to normalize sum aver all probabilities to one

$$L(y^{i}, y^{i'}) := \frac{e^{y^{i'}}}{\sum_{j}^{k} e^{y^{j'}}}$$

Use "one-hot" coding of y ◀

Y is now a vector with k (number of classes) entries and y^i is the kth class label

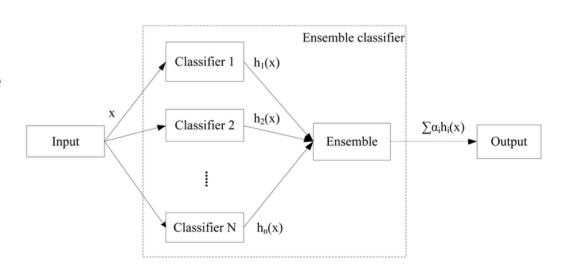


Discussion



Ensemble Learning

- Very popular method based on ensemble learning
 - → many weak models decide together (by voting)
- Simple but powerful method
- Easy to implement and to parallelize
- Does not tend to overfit
- Build in Feature-Selection (next Lecture)

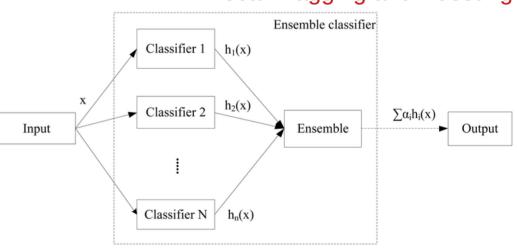




Ensemble Learning

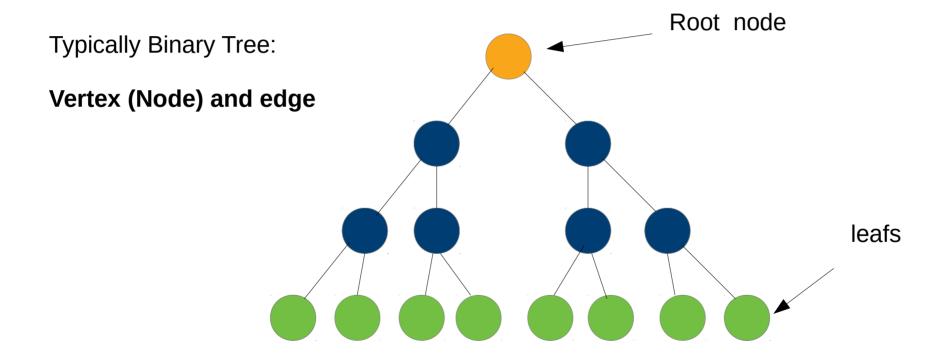
- Very popular method based on ensemble learning
 - → many weak models decide together (by voting)
- Simple but powerful method
- Approximating non-linear decision function by combination of piecewise linear functions
- Easy to implement and to parallelize
- Build in Feature-Selection (next Lecture)

Recall Bagging and Boosting





Decision Trees: the base classifier for Random Forests

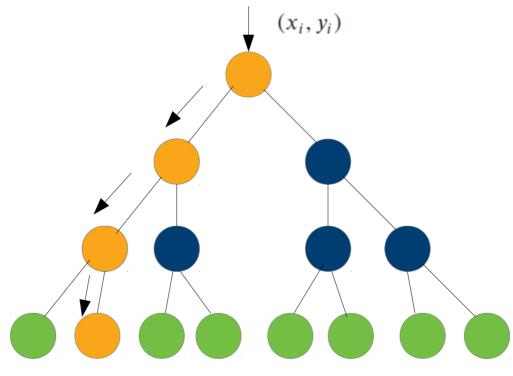




Goal: divide training data samples X at each node such that leafs have (mostly)

Pure class labels

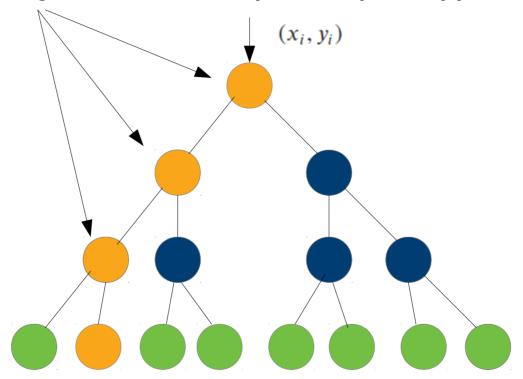
Classification: label y' is the label of the leaf node



$$\forall y_i, y_j \in L_1 : y_i = y_j$$



All we need is a splitting function that will produce (almost) pure class labels



$$\forall y_i, y_j \in L_1 : y_i = y_j$$



Top Down Training:

Assume k classes and data of dimension d

- Fill tree with ALL training samples from the root down
- In each node: compute probability for all class labels in node n

$$p_i := p(y = i) = \frac{|(x,i)| \in X_n}{|X_n|}$$

- Compute node purity based on class probabilities
- Split node along one dimension such that purity of children is increasing ← optimization



All we need is a splitting function that will produce (almost) pure class labels.

Entropy (a way to measure impurity):

$$Entropy = -\sum_{j} p_{j} \log_{2} p_{j}$$

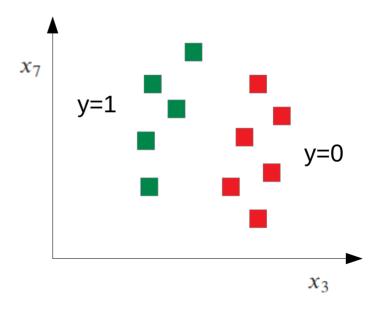
Gini index:

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



Split optimization is a simple line search (Example):

I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3





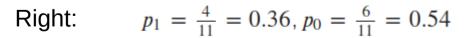
Split optimization is a simple line search (Example):

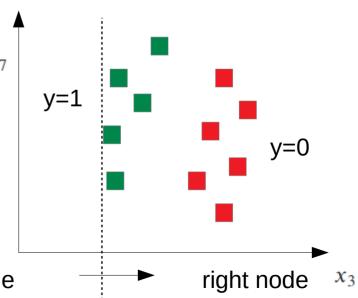
I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3

II. For each variable: find best split via line search e.g. for x_3

Left:

Undefined (div by zero)





Left node

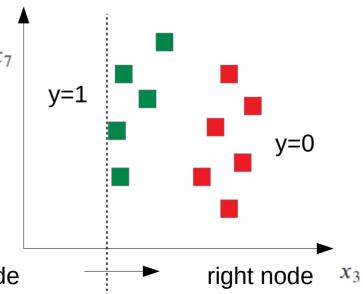


Split optimization is a simple line search (Example):

- I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3
- II. For each variable: find best split via line search e.g. for x_3

Left:
$$gini = 1 - (0^2 + 0^2) = 1$$

Right:
$$gini = 1 - (0.36^2 + 0.54^2) = 0.57$$



Left node

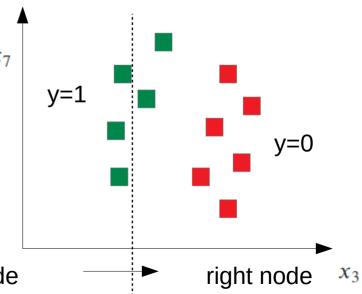


Split optimization is a simple line search (Example):

- I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3
- II. For each variable: find best split via line search e.g. for x_3

Left:
$$p_1 = \frac{3}{3} = 1, p_0 = \frac{0}{3} = 0$$

Right:
$$p_1 = \frac{2}{8} = 0.25, p_0 = \frac{6}{8} = 0.75$$



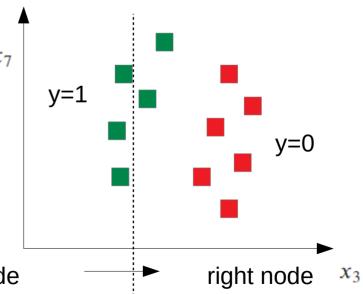


Split optimization is a simple line search (Example):

- I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3
- II. For each variable: find best split via line search e.g. for x_3

Left:
$$gini = 1 - (1^2 + 0^2) = 0$$

Right:
$$gini = 1 - (0.25^2 + 0.75^2) = 0.375$$



Left node

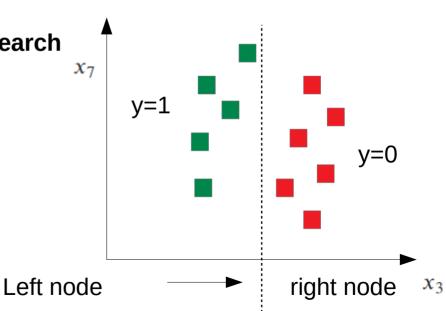


Split optimization is a simple line search (Example):

- I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3
- II. For each variable: find best split via line search e.g. for x_3

Left:
$$p_1 = \frac{5}{5} = 1, p_0 = \frac{0}{5} = 0$$

Right:
$$p_1 = \frac{0}{6} = 0, p_0 = \frac{6}{6} = 1$$



ML Summer School -Janis Keuper

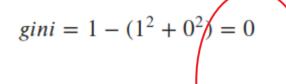


Split optimization is a simple line search (Example):

I. Select a random subset of variables (feature dimensions) from the data X e.g. x_7 x_3

II. For each variable: find best split via line search e.g. for x_3

Left:

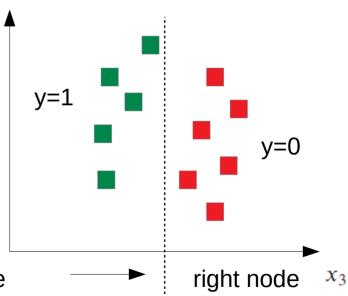


Right:

$$gini = 1 - (0^2 + 1^2) = 0$$

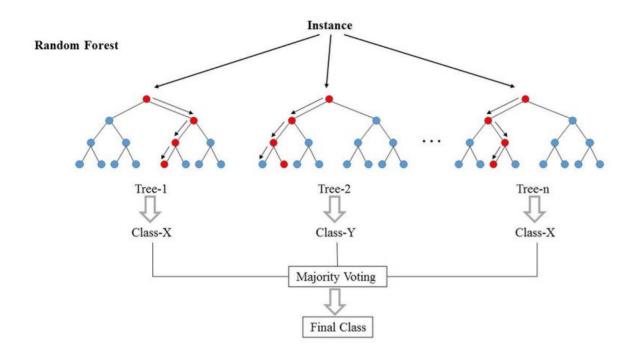
Perfect split!

Left node



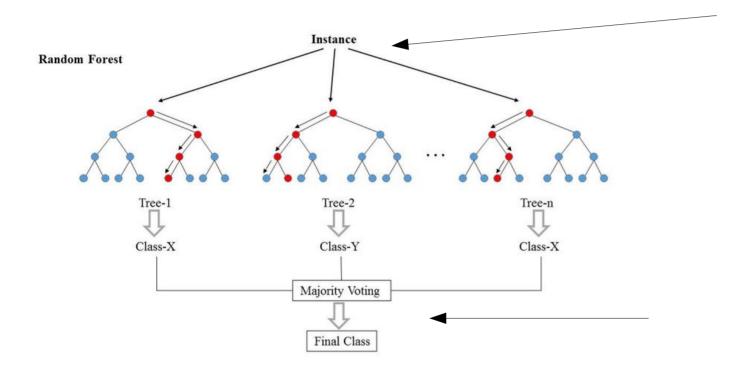


Ensemble Learning: A Forest of Trees





Ensemble Learning: A Forest of Trees



Split Training data into random subsets

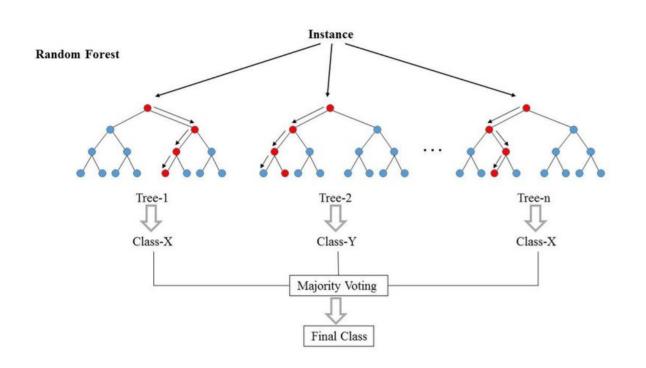
→ Bootstrap

Combine Models

→ Bagging



Ensemble Learning: A Forest of Trees

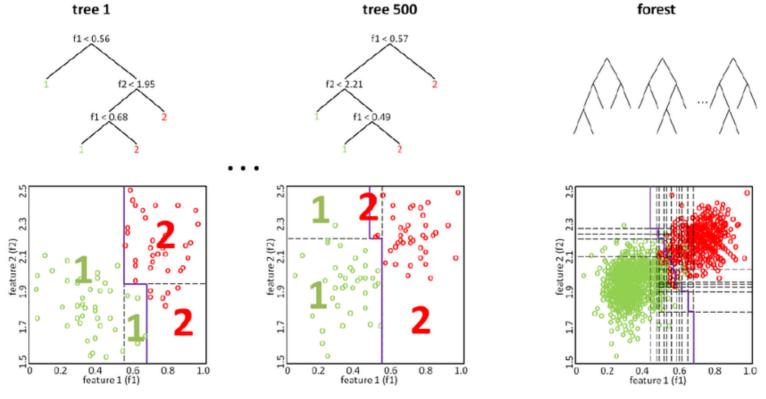


Parameters:

- #trees
- Portion of data per tree
- #vars per split
- Stopping
 - max depth
 - min samples per node



Classification example



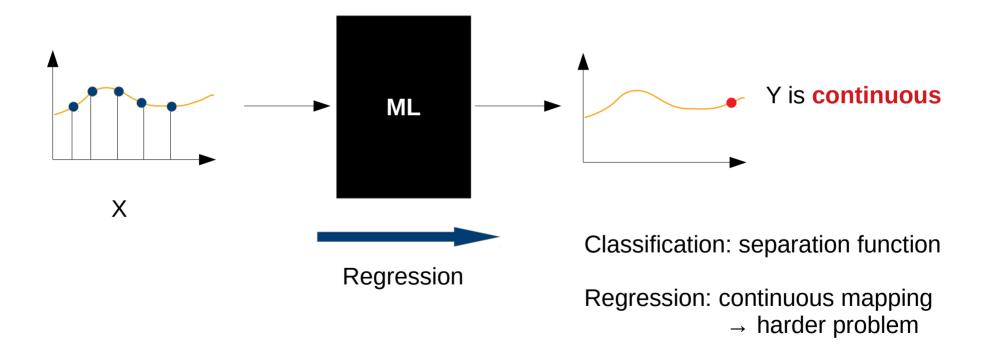


Discussion

Regression

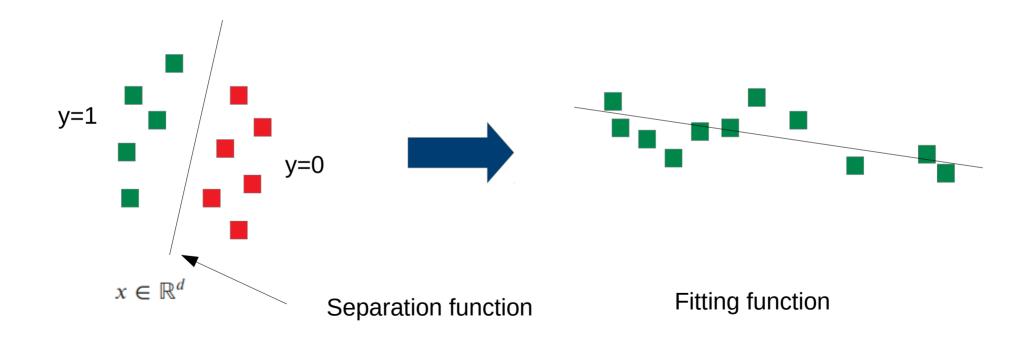


Recall:





How do we have to change out linear classifier to predict continuous values?

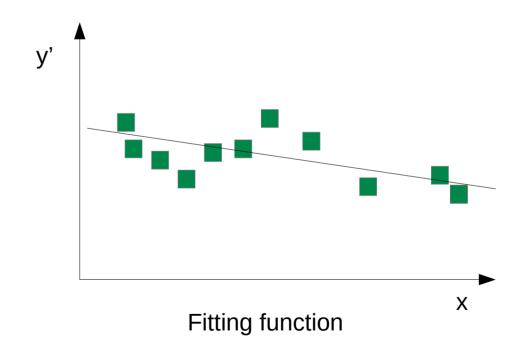




How do we have to change out linear classifier to predict continuous values?

Still can use the same framework

$$f(x) = y' = w^T x = \sum_{j=0}^d w_j x_j$$





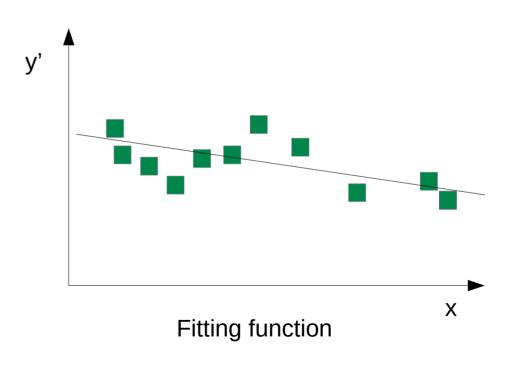
How do we have to change out linear classifier to predict continuous values?

Still can use the same framework

$$f(x) = y' = w^T x = \sum_{j=0}^d w_j x_j$$

Simply need new loss function in the optimization

$$\underset{w}{\arg\min} \sum_{i=0}^{N} L(y_i, w^T x_i)$$





Loss functions for regression:

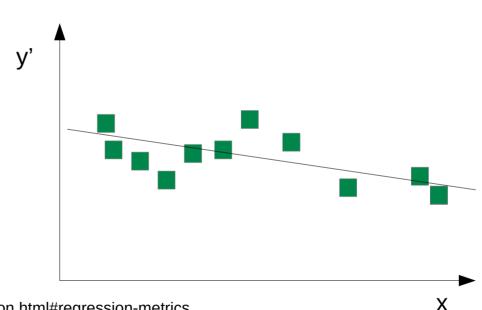
$$\underset{w}{\arg\min} \sum_{i=0}^{N} L(y_i, w^T x_i)$$

As simple as least squares error

$$L_{LSE}(y, y') := ||y - y'||^2$$

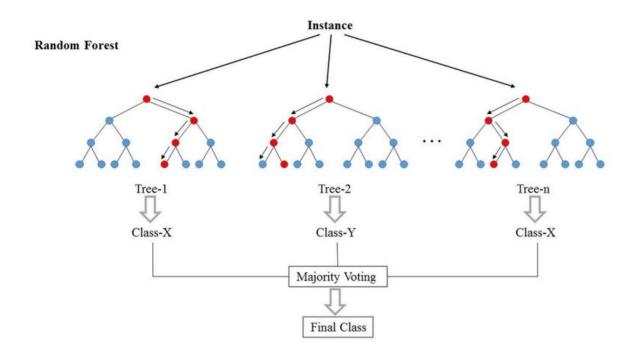
Many other error measures possible

- L1 (Histogram intersection)
- ...
 - → See https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics



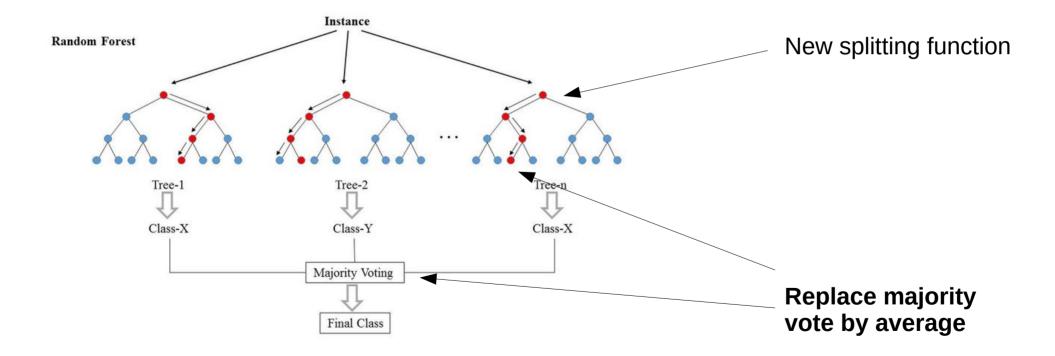


Recall:





Recall:





Splitting functions for regression:

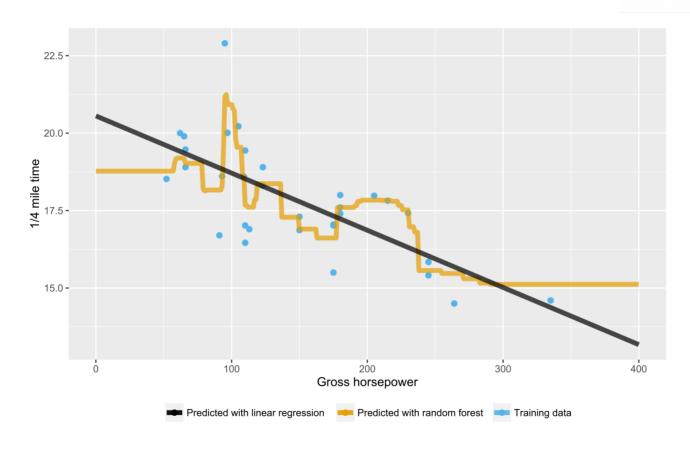
Goal: reduce "data spread" in node

→ use simple statistical measure like "mean square error"

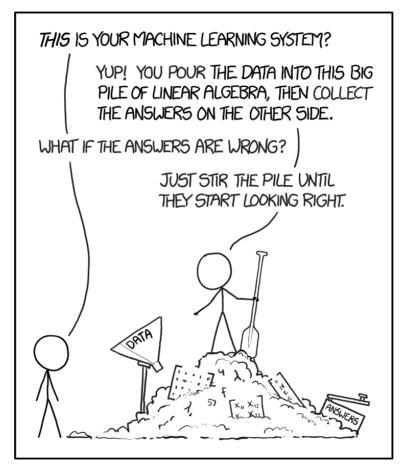
$$MSE : \sum_{(x_i, y_i) \in X_n} \|\mu_y - y_i\|^2$$



Example







https://xkcd.com/1838/