Lecture 6 – Supervised Learning

Classification models

Information Systems for Sustainable Society (is3) | WiSo Faculty | Univ.-Prof. Dr. Wolfgang Ketter | 11/11/22

Agenda

- Naïve Bayes Classifier
- Classification Tree
- Non-Linear Classification
- Evaluating Classification Models

Naive Bayes modelling

- **Naive Bayes** is a machine learning algorithm that **rests heavily on probabilistic modeling**
- But, it is also interpretable according to the three ingredients of a machine learning algorithm (hypothesis function, loss, optimization)
- Basic idea is that we model input and output as random variables $X = (X_1, X_2, \ldots, X_n)$ (several Bernoulli, categorical, or Gaussian random variables), and Y (one Bernoulli or categorical random variable)
- **Goal is to find $p(Y|X)$**, i.e. the probability of a category Y, given some data X

Naive Bayes assumptions

- We're going to find $p(Y|X)$ via Bayes' rule
  - $p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} = \frac{p(X|Y)p(Y)}{\Sigma_y p(X|y)p(y)}$

- The devisor is just the sum over all values of Y of the distribution specified by the numeration, so we're just going to focus on the $p(X|Y)p(Y)$ term

- Modelling full distribution $p(X|Y)$for high-dimensional X is not practical, so we're going to make the **naive Bayes assumption**, that the elements x_i are conditionally independent given Y
  - $p(X|Y) = \prod_{i=1}^{n} p(x_i|Y)$

Naive Bayes – An example of exact computation

Fraud detection

	X		Y
Company	Prior Legal Trouble	Company Size	Status
1	Yes	Small	Truthful
2	No	Small	Truthful
3	No	Large	Truthful
4	No	Large	Truthful
5	No	Small	Truthful
6	No	Small	Truthful
7	Yes	Small	Fraudulent
8	Yes	Large	Fraudulent
9	No	Large	Fraudulent
10	Yes	Large	Fraudulent

Area of interest,
i.e., where Y=fraudulent

Exact computation

- We want to classify companies depending on whether they are fraudulent or not
- In this example we can look at the full distribution for each random variable (i.e. PriorLegal={yes, no} and Size={small, medium, large}) due to the manageable dimensionality
- The conditional probabilities are computed by simply counting instances:

$$P(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{small}) = 1/2 = 0.5$$

$$P(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{large}) = 2/2 = 1$$

$$P(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{small}) = 0/3 = 0$$

$$P(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{large}) = 1/3 = 0.33$$

Naive Bayes – An example of the Naïve Bayes assumption

Fraud detection

	X		Y
Company	Prior Legal Trouble	Company Size	Status
1	Yes	Small	Truthful
2	No	Small	Truthful
3	No	Large	Truthful
4	No	Large	Truthful
5	No	Small	Truthful
6	No	Small	Truthful
7	Yes	Small	Fraudulent
8	Yes	Large	Fraudulent
9	No	Large	Fraudulent
10	Yes	Large	Fraudulent

Area of interest,
i.e., where Y=fraudulent

Naïve Bayes assumption

- The approach outlined above amounts to finding all the records in the sample that are exactly like the new record
- When the number of predictors gets larger, many of the records to be classified will be without exact matches
- So we use the Naive Bayes Assumption of $p(X|Y) = \prod_{i=1}^{n} p(x_i|Y)$
- For the first example:

$$P(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{small}) =$$

  - $p(X|Y) = \prod_{i=1}^{n} p(x_i|Y) =$ P(PriorLegal=yes | fraudulent) * P(Size=small | fraudulent) =(3/4)*(1/4)
  - $p(Y)$ = 4/10
  - $p(X)$=(3/4)∗(1/4)*(4/10) + (1/6)*(4/6)*(6/10)

Naive Bayes – An example of the Naïve Bayes assumption

Fraud detection

	X		Y
Company	Prior Legal Trouble	Company Size	Status
1	Yes	Small	Truthful
2	No	Small	Truthful
3	No	Large	Truthful
4	No	Large	Truthful
5	No	Small	Truthful
6	No	Small	Truthful
7	Yes	Small	Fraudulent
8	Yes	Large	Fraudulent
9	No	Large	Fraudulent
10	Yes	Large	Fraudulent

Area of interest,
i.e., where Y=fraudulent

Naïve Bayes assumption

- Overall, we obtain the following:

$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{small}) = \frac{(3/4)(1/4)(4/10)}{(3/4)(1/4)(4/10) + (1/6)(4/6)(6/10)} = 0.53$$
$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{large}) = 0.87$$
$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{small}) = 0.07$$
$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{large}) = 0.31$$

- Note how close this comes to the exact values derived earlier:

$$P(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{small}) = 1/2 = 0.5$$
$$P(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{large}) = 2/2 = 1$$
$$P(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{small}) = 0/3 = 0$$
$$P(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{large}) = 1/3 = 0.33$$

Naive Bayes – An example of the Naïve Bayes assumption

Fraud detection

Company	Prior Legal Trouble	Company Size	Status
1	Yes	Small	Truthful
2	No	Small	Truthful
3	No	Large	Truthful
4	No	Large	Truthful
5	No	Small	Truthful
6	No	Small	Truthful
7	Yes	Small	Fraudulent
8	Yes	Large	Fraudulent
9	No	Large	Fraudulent
10	Yes	Large	Fraudulent

Naïve Bayes assumption

- The approach outlined above amounts to finding all the records in the sample that are exactly like the new record.
- When the number of predictors gets larger, many of the records to be classified will be without exact matches
- So we use the Naive Bayes Assumption of $p(X|Y) = \prod_{i=1}^{n} p(x_i|Y)$

$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{small}) = \frac{(3/4)(1/4)(4/10)}{(3/4)(1/4)(4/10) + (1/6)(4/6)(6/10)} = 0.53$$

$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{y}, \text{Size} = \text{large}) = 0.87$$

$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{small}) = 0.07$$

$$P_{\text{nb}}(\text{fraudulent}|\text{PriorLegal} = \text{n}, \text{Size} = \text{large}) = 0.31$$

In practice, we would typically work with distributions and estimate the model using maximum likelihood estimation (MLE)

- We're going to explicitly model the distribution of each $p(x_i|Y)$ as well as $p(Y)$
- We do this by specifying a distribution for $p(Y)$ and a separate distribution for each $p(x_i|Y = y)$
- So assuming, for instance, that Y_i and X_i are binary (Bernoulli random variables), then we would represent the distributions
  - $p(Y; \phi_0)$, $p(X_i|Y = 0; \phi_i^0)$, $p(X_i|Y = 1; \phi_i^1)$
- We then estimate the parameters of these distributions using Maximum Likelihood Estimation (MLE), i.e.
  - $\phi_0 = \frac{\sum_{j=1}^{m} y^{(i)}}{m}$, $\phi_i^y = \frac{\sum_{j=1}^{m} x_i^{(j)} .1\{y^{(i)}=y\}}{\sum_{j=1}^{m} 1\{y^{(i)}=y\}}$

Making predictions

- Given some new data point x, you can now compute the probability of each class
  - $p(Y = y|x) \propto p(Y = y) \prod_{i=1}^{m} p(x_i|Y = y) = \phi_0 \prod_{i=1}^{m} (\phi_i^y)^{x_i} (1 - \phi_1^y)^{1-x_i}$
- After you have computed the right hand side, just normalize (divide by the sum over all y) to get the desired probability
- Alternatively, if you just want to know the most likely Y, just compute each right hand side and take the maximum

Potential issues

- **Problem #1:** when computing probability, the product $p(y)\prod_{i=1}^{n} p(x_i\,|y)$ quickly goes to zero numerical precision
- **Solution:** compute log of the probabilities instead
  - $\log p(y) + \sum_{i=1}^{n} \log p(x_i\,|y)$

- **Problem #2:** If we have never seen either $X_i = 1$ or $X_i = 0$ for a given y, then the corresponding probabilities computed by MLE will be zero
- **Solution:** Laplace smoothing, „hallucinate“ one X_i =0/1 for each class
  - $\phi_i^y = \frac{\sum_{j=1}^{m} x_i^{(j)} .1\{y^{(j)}=y\} + 1}{\sum_{j=1}^{m} 1\{y^{(j)}=y\}+2}$

Other distributions

- Though naive Bayes is often presented as "just" counting, the value of the maximum likelihood interpretation is that it's clear how to model $p(X_i \mid Y)$ for non-categorical random variables
- Example: if x_i is real-valued, we can model $p(X_i \mid Y = y)$ as a Gaussian
  - $p\left(x_i \mid y ; \mu, \sigma_y^2\right) = \mathcal{N}(x_i ; \mu^y, \sigma_y^2)$
- With maximum likelihood estimates
  - $\mu^y = \frac{\sum_{j=1}^{m} x_i^{(j)} . 1\{y^{(j)}=y\}}{\sum_{j=1}^{m} 1\{y^{(j)}=y\}}, \quad \sigma_y^2 = \frac{\sum_{j=1}^{m} (x_i^{(j)} - \mu^y)^2 . 1\{y^{(j)}=y\}}{\sum_{j=1}^{m} 1\{y^{(j)}=y\}}$
- All probability computations are exactly the same as before (it doesn't matter that some of the terms are probability densities)

Agenda

- Naïve Bayes Classifier
- **Classification Tree**
- Non-Linear Classification
- Evaluating Classification Models

MegaTelCo: Predicting Customer Churn

- You just landed a great analytical job with MegaTelCo, one of the largest telecommunication firms in the US.
- The company is having a major problem with customer retention in their wireless business: In the mid-Atlantic region, 20% of cell phone customers leave when their contract expires.
- Attracting new customers is much more expensive than retaining existing ones, so a good deal of marketing budget is allocated to prevent churn.
- MegaTelCo's marketing section has already designed a special retention offer.
- Your task is to devise a precise, step-by-step plan for how the data science team should use MegaTelCo's vast data resources to solve the problem.

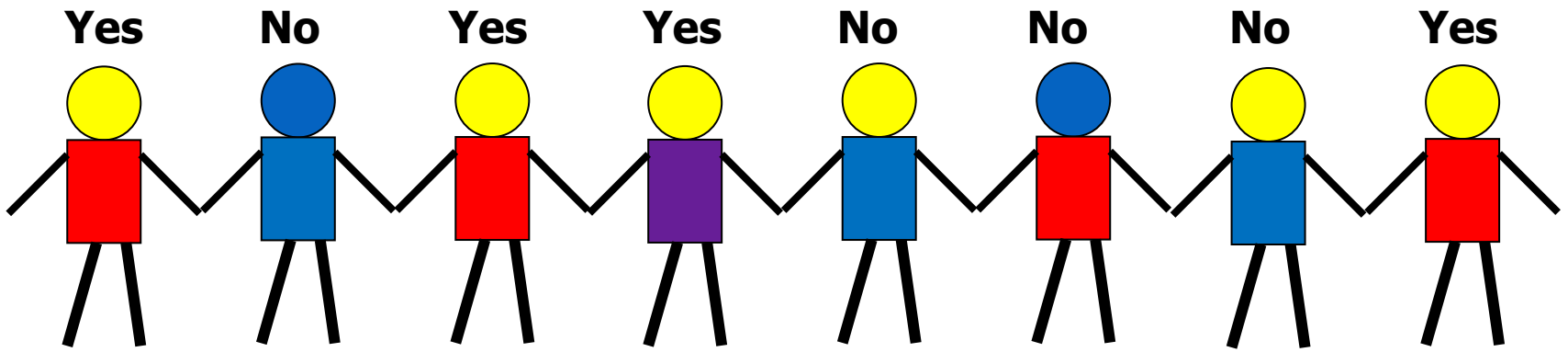
- You can find a more detailed problem description on Kaggle.

MegaTelCo: Predicting Customer Churn

- How should MegaTelCo choose customers in order to best reduce churn?
- More precisely, you should determine which customers should be offered the special retention deal that the marketing section designed.
- In terms of data analysis this means that you should segment the population into groups that differ from each other with respect to certain characteristics.
- One way to segment a given population are Tree-Structured Models.

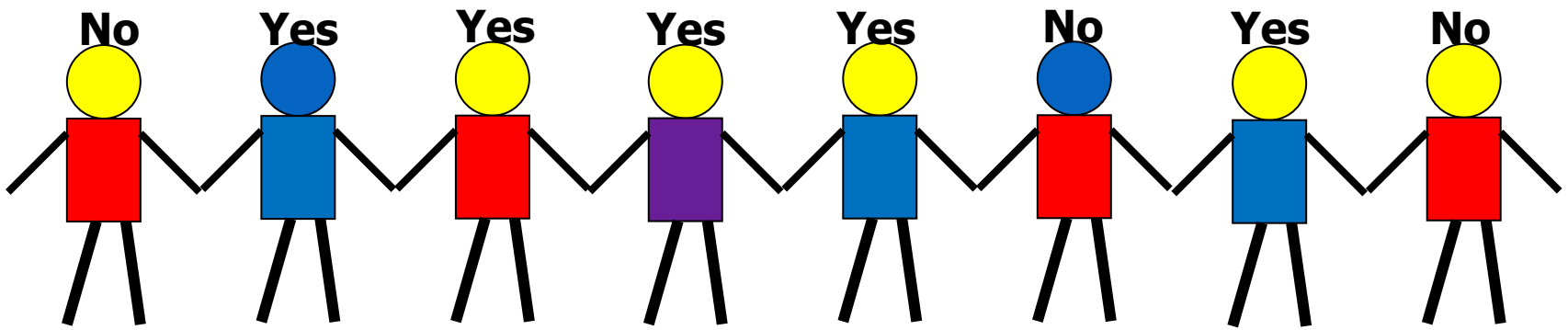
Questions to answer during this section

- How can we segment the population into groups that differ from each other with respect to certain characteristics?
- How can we judge whether an attribute contains important information about the target variable?
- If we select the *single* variable that gives the most information gain, we create a very simple segmentation. However, if we select multiple attributes each giving some information gain, how do we put them together?



Selecting Informative Attributes

- Objective: Based on customer attributes, partition the customers into subgroups that are less impure – with respect to the class (i.e., such that in each group as many instances as possible belong to the same class)

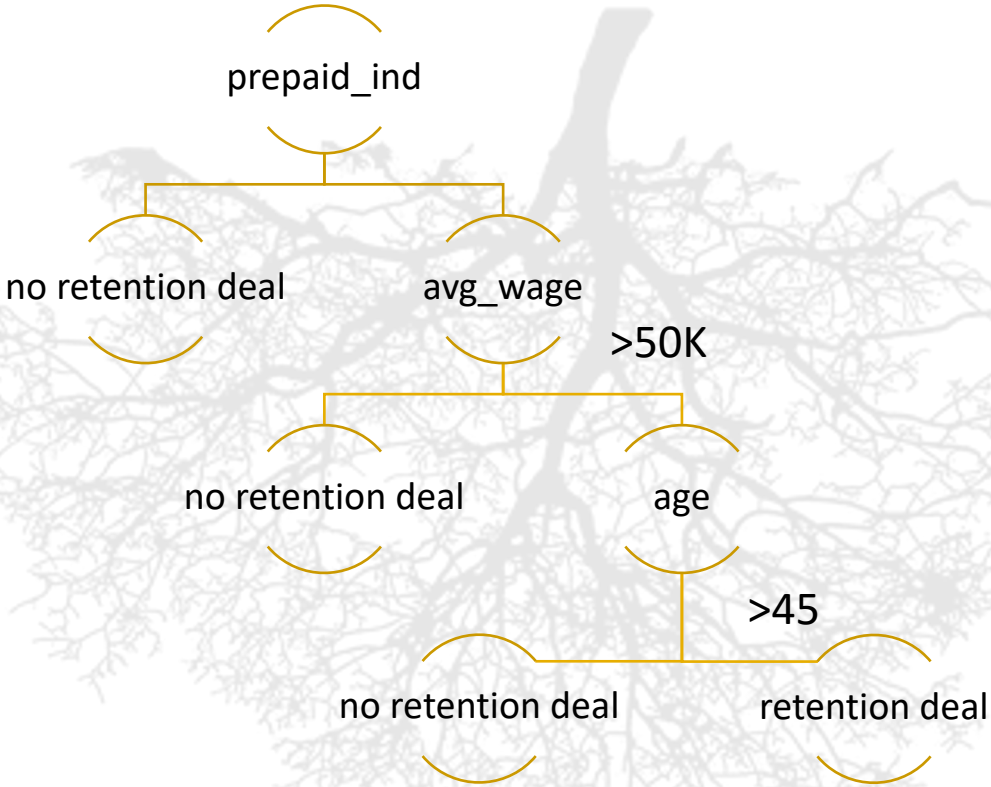


Tree-Structured Models

- Goal: Classify or predict an outcome based on a set of predictors
- The output is a set of **rules**
- Also called CART or Decision Trees
- Rules are represented by tree diagrams

Example:

- Goal: classify a person as “no retention deal” or “retention deal”
- Rule might be “IF (prepaid_ind = NO) AND (avg_wage >50K) AND (age > 45) THEN class = retention deal



Tree-Structured Models: Key ideas

Recursive partitioning: Repeatedly split the records into two parts so as to achieve maximum homogeneity within the new parts

Pruning the tree: Simplify the tree by pruning peripheral branches to avoid overfitting

Recursive Partitioning Steps

- Pick one of the predictor variables, x_i
- Pick a value of x_i, say s_i, that divides the training data into two (not necessarily equal) portions
- Measure how "pure" or homogeneous each of the resulting portions are

  "Pure" = containing records of mostly one class
- Idea is to pick x_i and s_i to maximize purity
- Repeat the process

Example: Riding Mowers

- Data: 24 households classified as owning or not owning riding mowers
- Predictors: Income, Lot Size

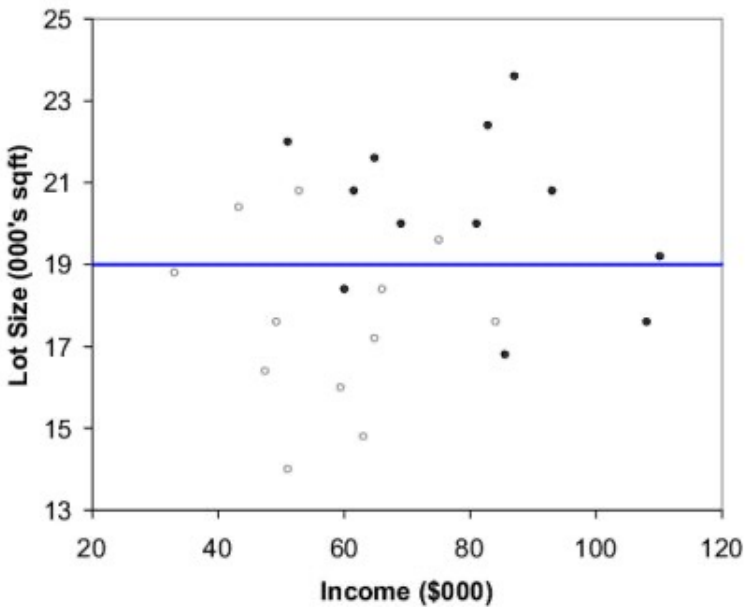
How to split:

- Order records according to one variable, say *lot size*
- Find midpoints between successive values
- E.g., first midpoint is 14.4 (halfway between 14.0 and 14.8)
- Divide records into those with *lot size* > 14.4 and those < 14.4
- After evaluating that split, try the next one, which is 15.4 (halfway between 14.8 and 16.0)

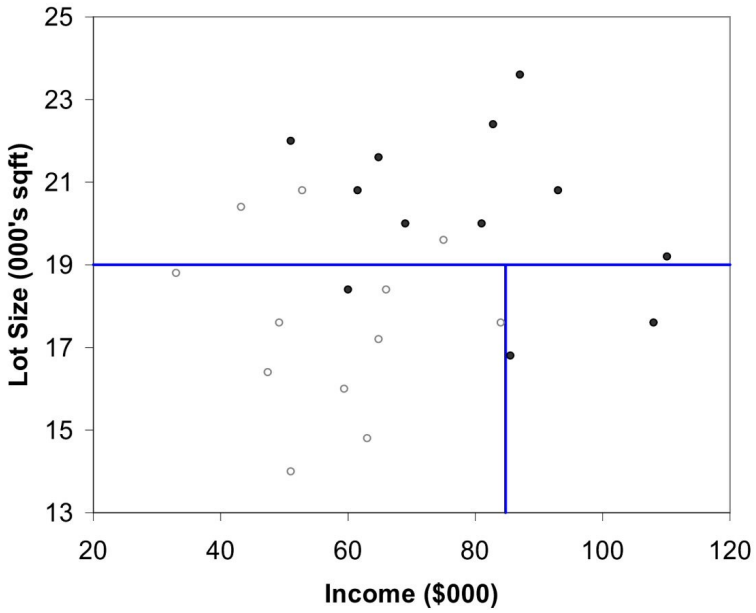
Income	Lot_Size	Ownership
60.0	18.4	owner
85.5	16.8	owner
64.8	21.6	owner
61.5	20.8	owner
87.0	23.6	owner
110.1	19.2	owner
108.0	17.6	owner
82.8	22.4	owner
69.0	20.0	owner
93.0	20.8	owner
51.0	22.0	owner
81.0	20.0	owner
75.0	19.6	non-owner
52.8	20.8	non-owner
64.8	17.2	non-owner
43.2	20.4	non-owner
84.0	17.6	non-owner
49.2	17.6	non-owner
59.4	16.0	non-owner
66.0	18.4	non-owner
47.4	16.4	non-owner
33.0	18.8	non-owner
51.0	14.0	non-owner
63.0	14.8	non-owner

Example: Riding Mowers

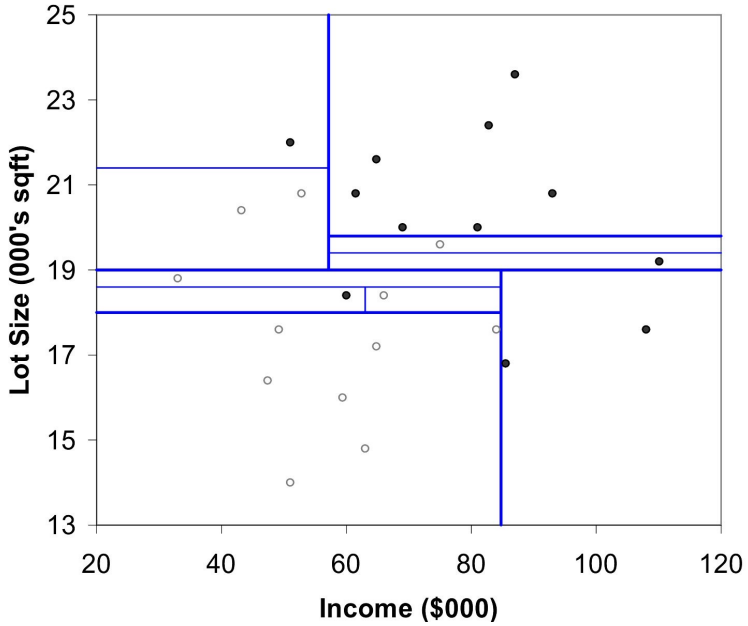
First Split: Lot Size = 19,000



Second Split: Income = $84,000



After All Splits



Splits and Impurity

- Splits should be chosen such that impurity is reduced the most
- Impurity can be measured by the **Gini Index** as well as **Entropy**
- At each successive stage, compare two measures just mentioned across all possible splits in all variables

Measuring Impurity: Gini Index

Gini Index for rectangle *A* containing *m* records

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

p = proportion of cases in rectangle *A* that belong to class *k*

- I(A) = 0 when all cases belong to same class
- I(A) at a max when all classes are equally represented (= 0.50 in binary case)

Measuring Impurity: Entropy

Entropy can be measured using the following formula

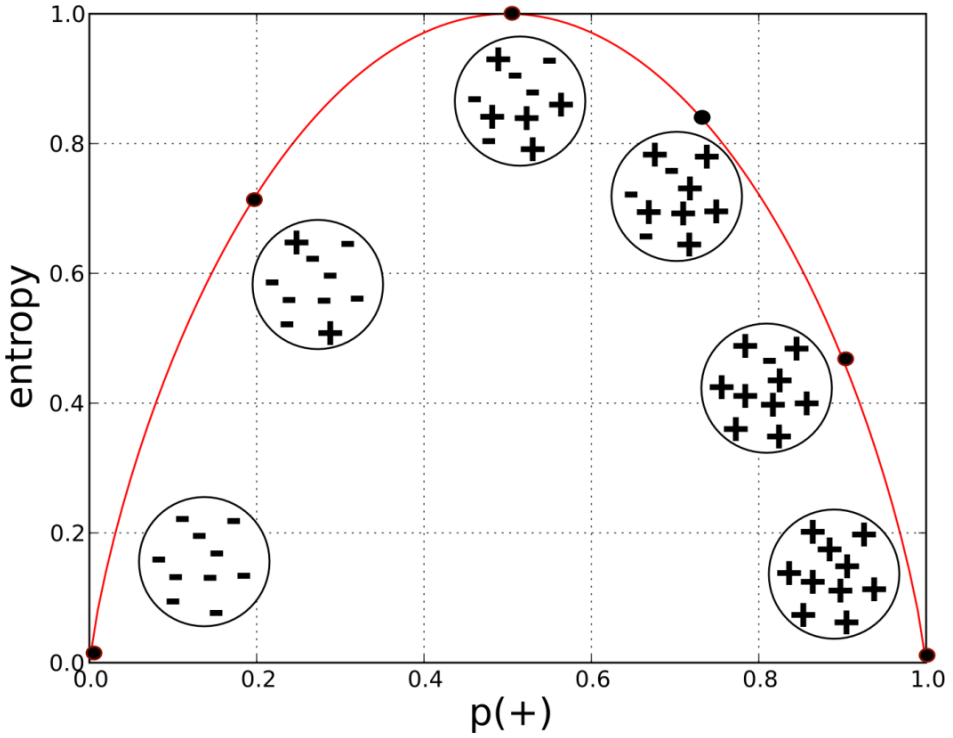
$$entropy\ (A) = -\sum_{k=1}^{m} p_k\ log_2(p_k)$$

p = proportion of cases (out of *m*) in rectangle *A* that belong to class *k*

- entropy (A) = 0 when most pure
- entropy (A) = $\log_2(m)$ when an equal representation of classes is present

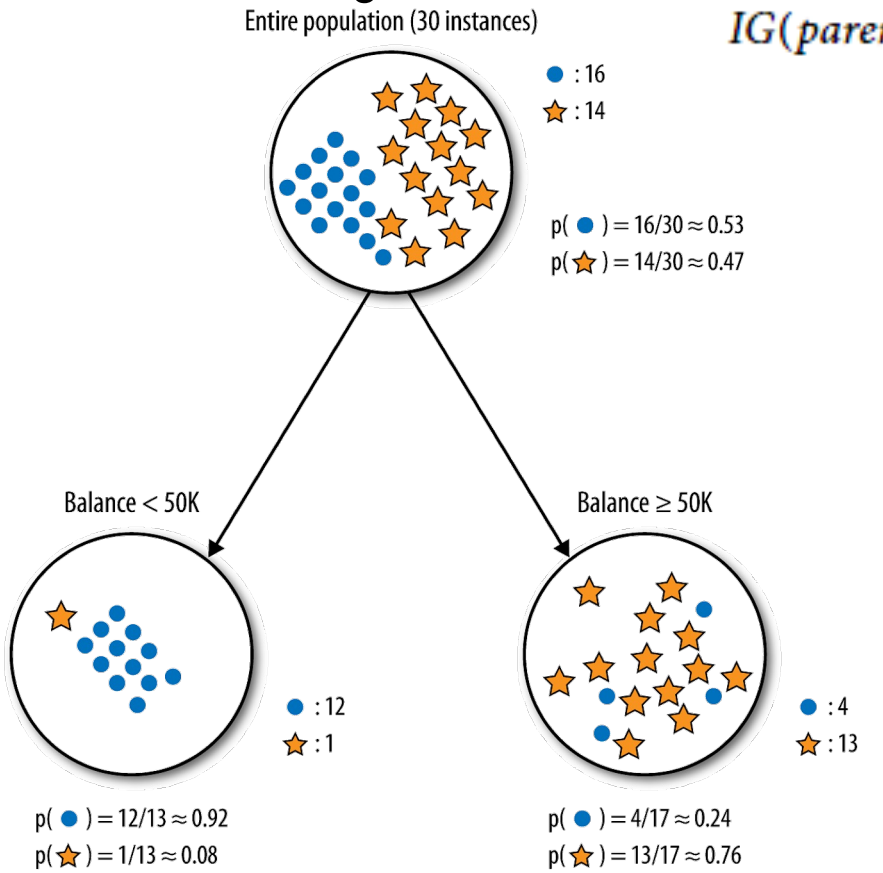
Selecting Informative Attributes

- The most common splitting criterion is called **information gain** (IG):
  - It is based on a **purity measure** called **entropy**
    - $entropy = -\mathrm{p}_1 \log_2(p_1) - p_2 \log_2(p_2) - ..$
    - Measures the general disorder of a set
  - Why using $\log_2$?
    - Shannon's information theory!
    - Based on bits!



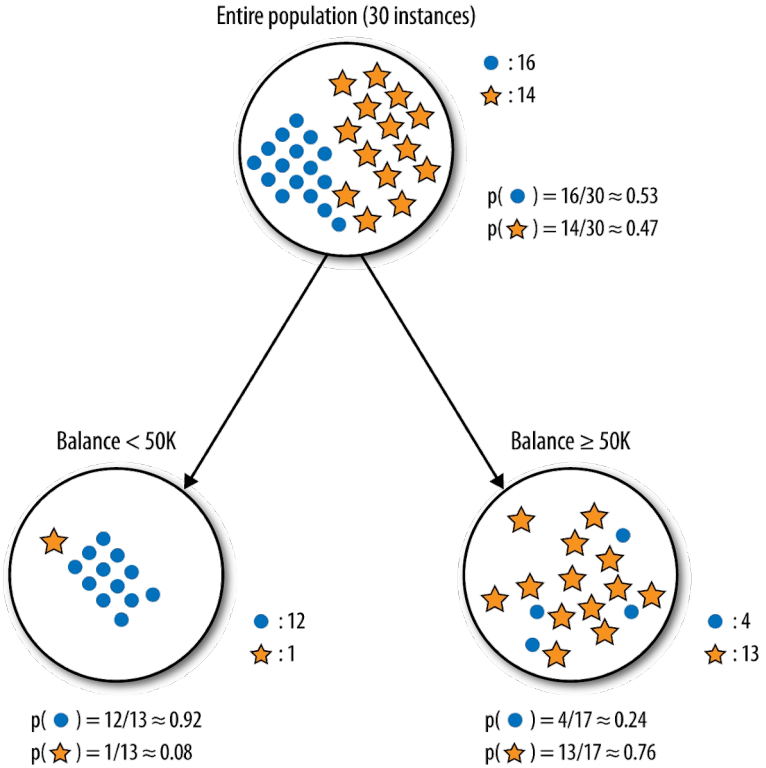
Information Gain

- Information gain measures the *change* in entropy due to any amount of new information being added



$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots]$$

Information Gain



$$entropy(parent) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.53 \times -0.9 + 0.47 \times -1.1]$$
$$\approx 0.99 \quad \text{(very impure)}$$

The entropy of the *left* child is:

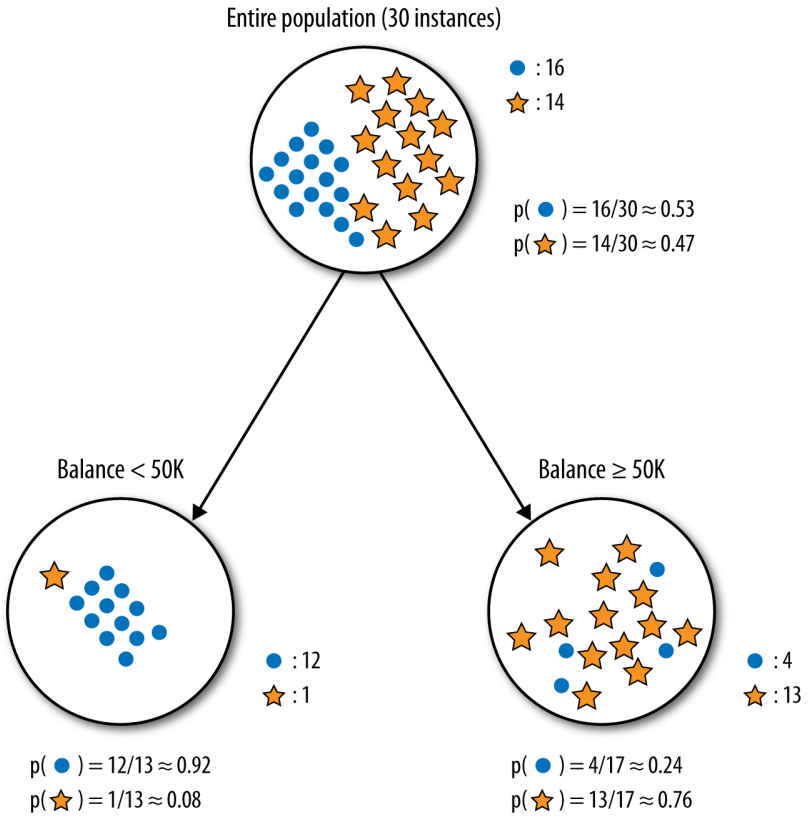
$$entropy(Balance < 50K) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.92 \times (-0.12) + 0.08 \times (-3.7)]$$
$$\approx 0.39$$

The entropy of the *right* child is:

$$entropy(Balance \geq 50K) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.24 \times (-2.1) + 0.76 \times (-0.39)]$$

≈ 0.79

Information Gain



$$
\begin{aligned}
IG &= entropy(parent) - [p(\text{Balance} < 50\text{K}) \times entropy(\text{Balance} < 50\text{K}) \\
&\quad + p(\text{Balance} \geq 50\text{K}) \times entropy(\text{Balance} \geq 50\text{K})] \\
&\approx 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\
&\approx 0.37
\end{aligned}
$$

From Splits to the Tree Structure

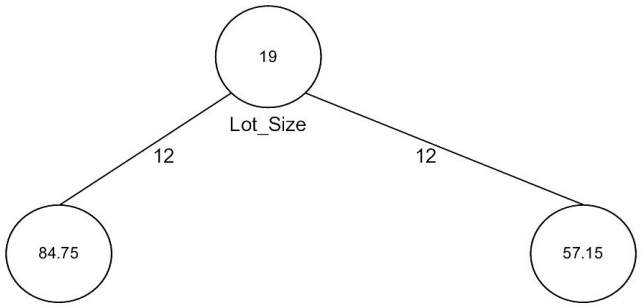
- Split points become nodes on tree (circles with split value in center)
- Rectangles represent “leaves” (terminal points, no further splits, classification value noted)
- Numbers on lines between nodes indicate # cases

Determining Leaf Node Label

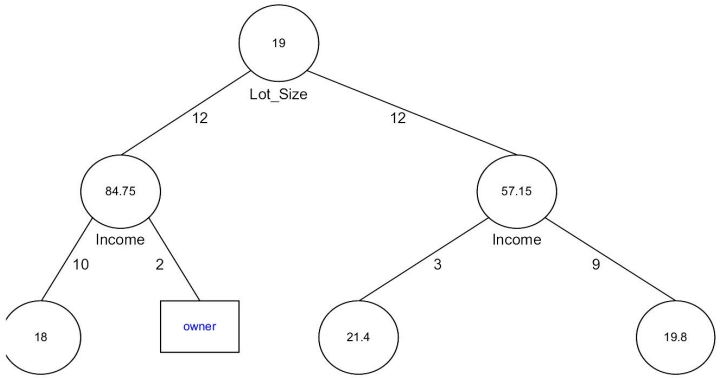
- Each leaf node label is determined by “voting” of the records within it, and by the cutoff value
- Records within each leaf node are from the training data
- Default cutoff=0.5 means that the leaf node’s label is the majority class
- Cutoff = 0.75: requires majority of 75% or more “1” records in the leaf to label it a “1” node

The Tree Structure

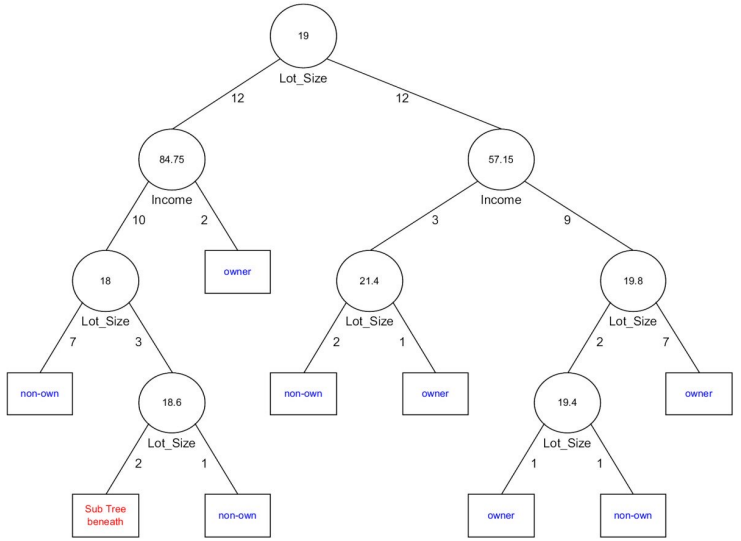
First Split



Second Split

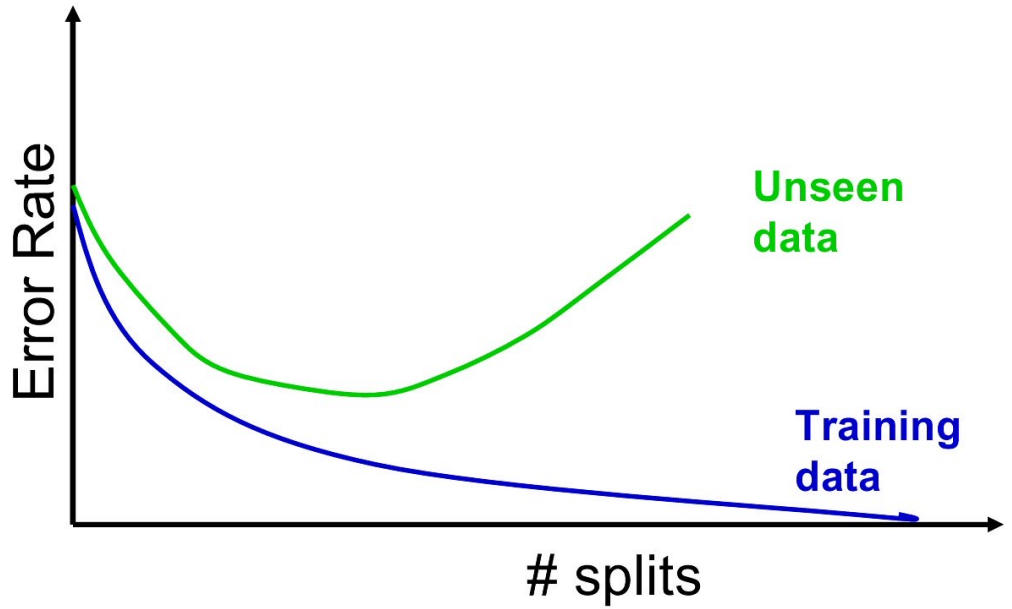


All Splits



The Overfitting Problem: Stopping Tree Growth

- Natural end of process is 100% purity in each leaf
- This overfits the data, which end up fitting noise in the data
- Overfitting leads to low predictive accuracy of new data
- Past a certain point, the error rate for the validation data starts to increase



The Overfitting Problem: Pruning

- CART lets tree grow to full extent, then prunes it back
- Idea is to find that point at which the validation error begins to rise
- Generate successively smaller trees by pruning leaves
- At each pruning stage, multiple trees are possible
- Use *cost complexity* to choose the best tree at that stage

The Overfitting Problem: Cost Complexity

$$CC(T) = Err(T) + \alpha L(T)$$

CC(T) = cost complexity of a tree

Err(T) = proportion of misclassified records

α = penalty factor attached to tree size (set by user)

- Among trees of given size, choose the one with lowest CC
- Do this for each size of tree

The Overfitting Problem: Pruning Results

This process yields a set of trees of different sizes and associated error rates.

Two trees of interest:

- Minimum error tree
  - Has lowest error rate on validation data
- Best pruned tree
  - Smallest tree within one std. error of min. error
  - This adds a bonus for simplicity/parsimony

Advantages and Disadvantages of Trees

Advantages

- Easy to use, understand
- Produce rules that are easy to interpret & implement
- Variable selection & reduction is automatic
- Do not require the assumptions of statistical models
- Can work without extensive handling of missing data

Disadvantages

- May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits
- Since the process deals with one variable at a time, no way to capture interactions between variables

Summery

- Classification and Regression Trees are an easily understandable and transparent method for predicting or classifying new records
- A tree is a graphical representation of a set of rules
- Trees must be pruned to avoid over-fitting of the training data
- As trees do not make any assumptions about the data structure, they usually require large samples

Agenda

- Naïve Bayes Classifier
- Classification Tree
- **Non-Linear Classification**
- Evaluating Classification Models

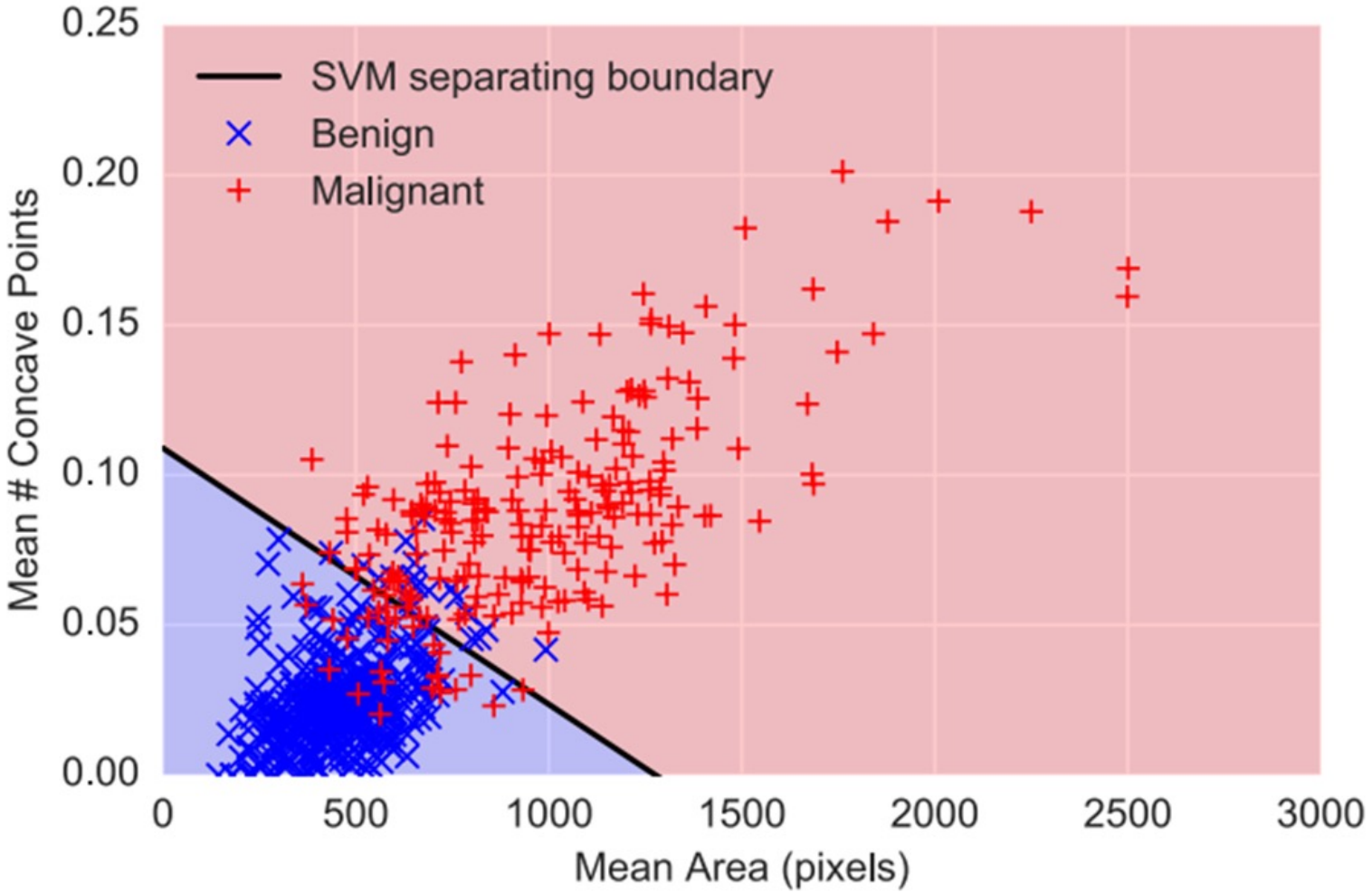
Nonlinear classification

- Just like linear regression, the nice thing about using nonlinear features for classification is that our algorithms remain exactly the same as before
  - I.e., for an SVM, we just solve (using gradient descent)

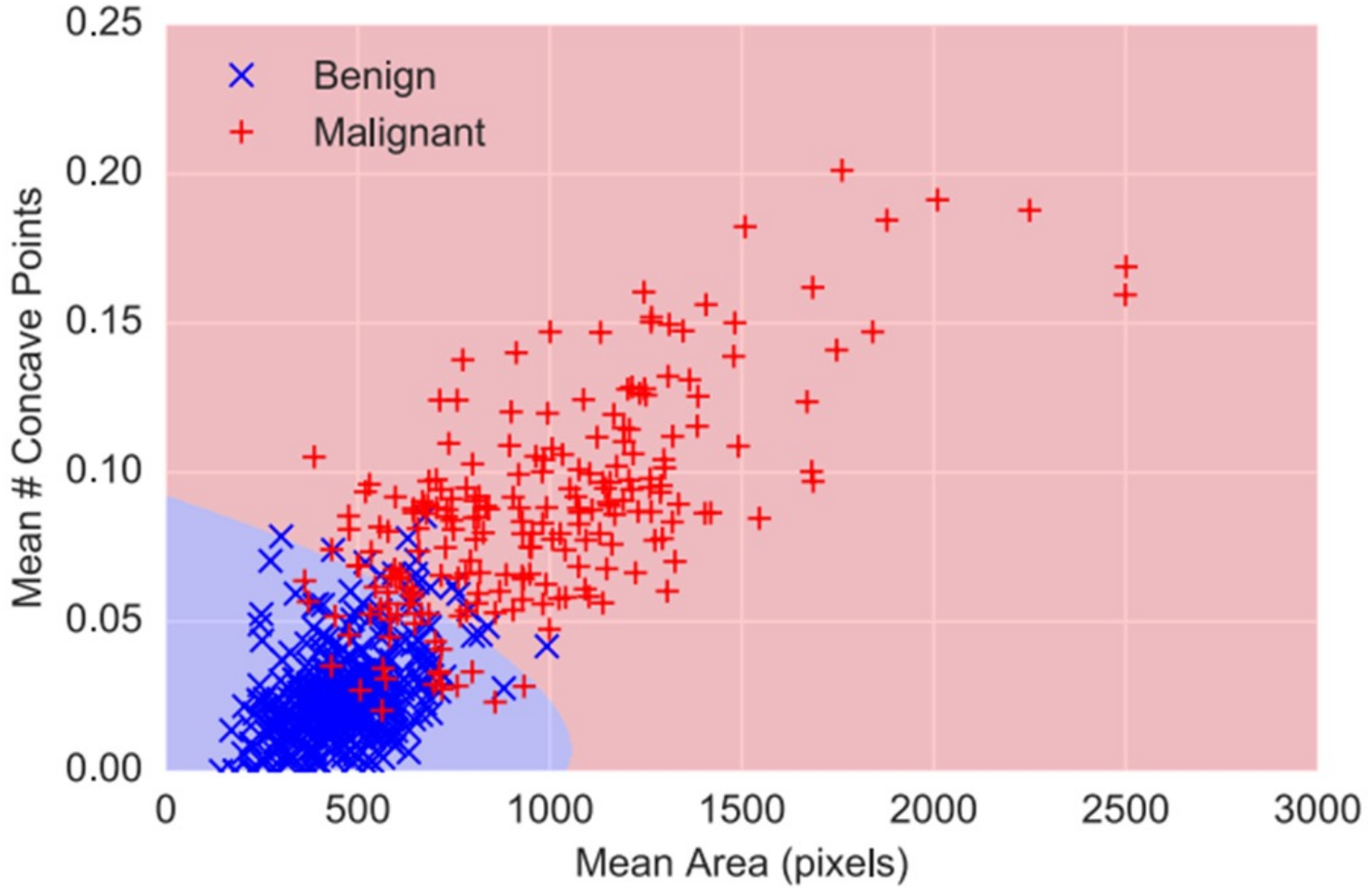
$$\underset{\theta}{Minimize} \sum_{i=1}^{m} \max\{1 - y^{(i)}.\theta^T x^{(i)}, 0\} + \frac{\lambda}{2}\|\theta\|_2^2$$

- Only difference is that $x^{(i)}$ now contains non-linear functions of the input data

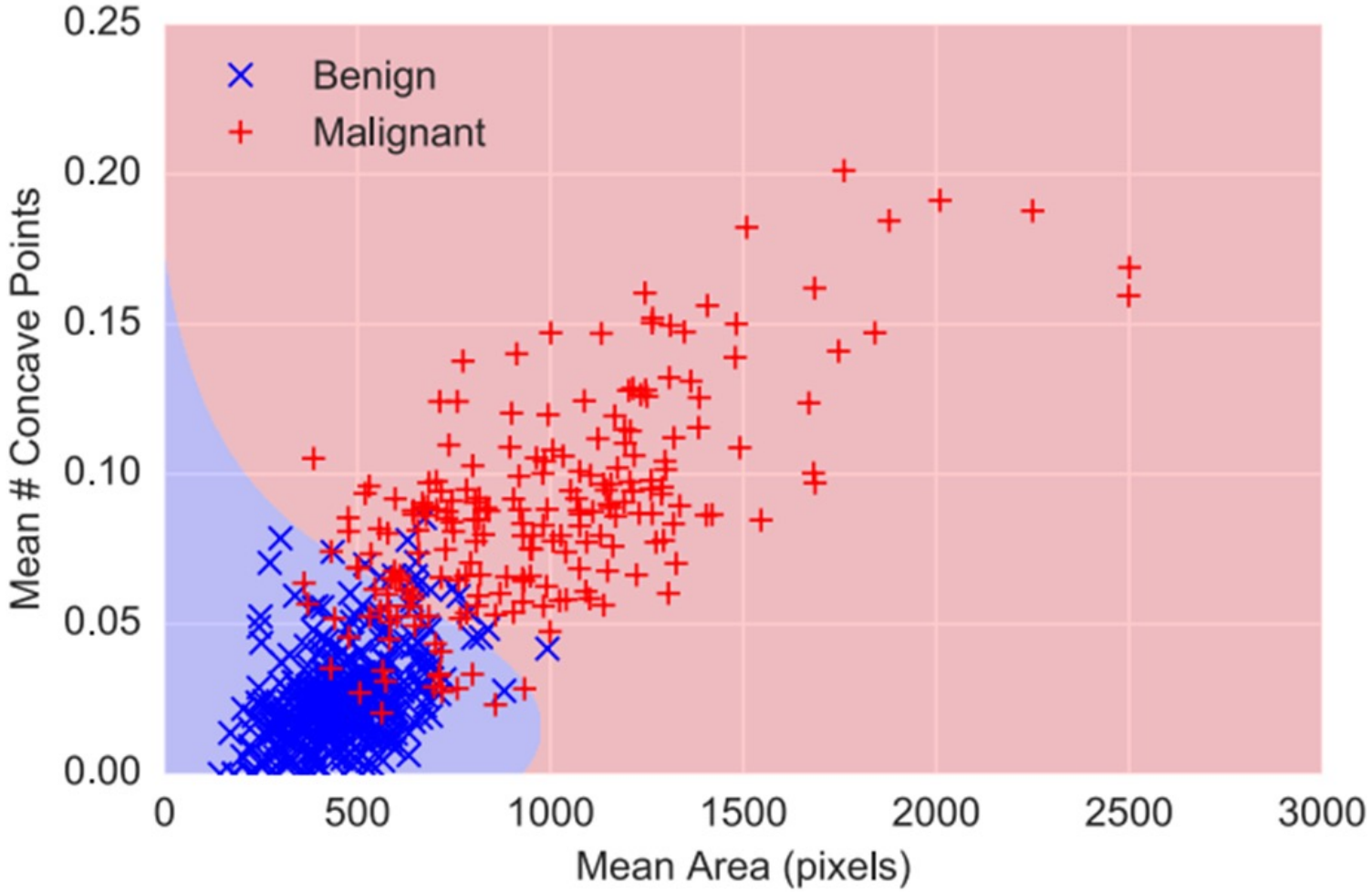
Linear SVM on cancer data set



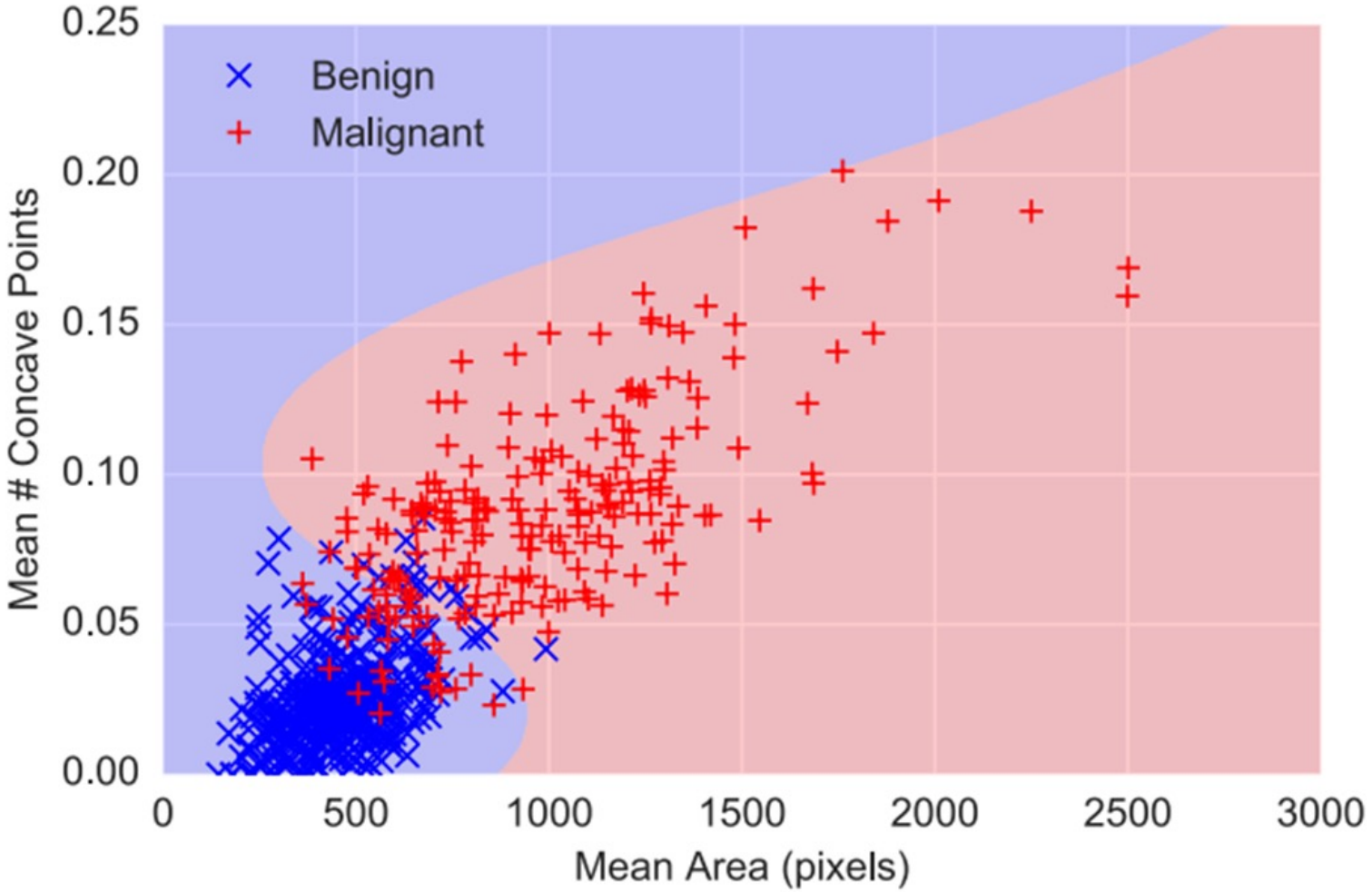
Polynomial features d = 2



Polynomial features d = 3



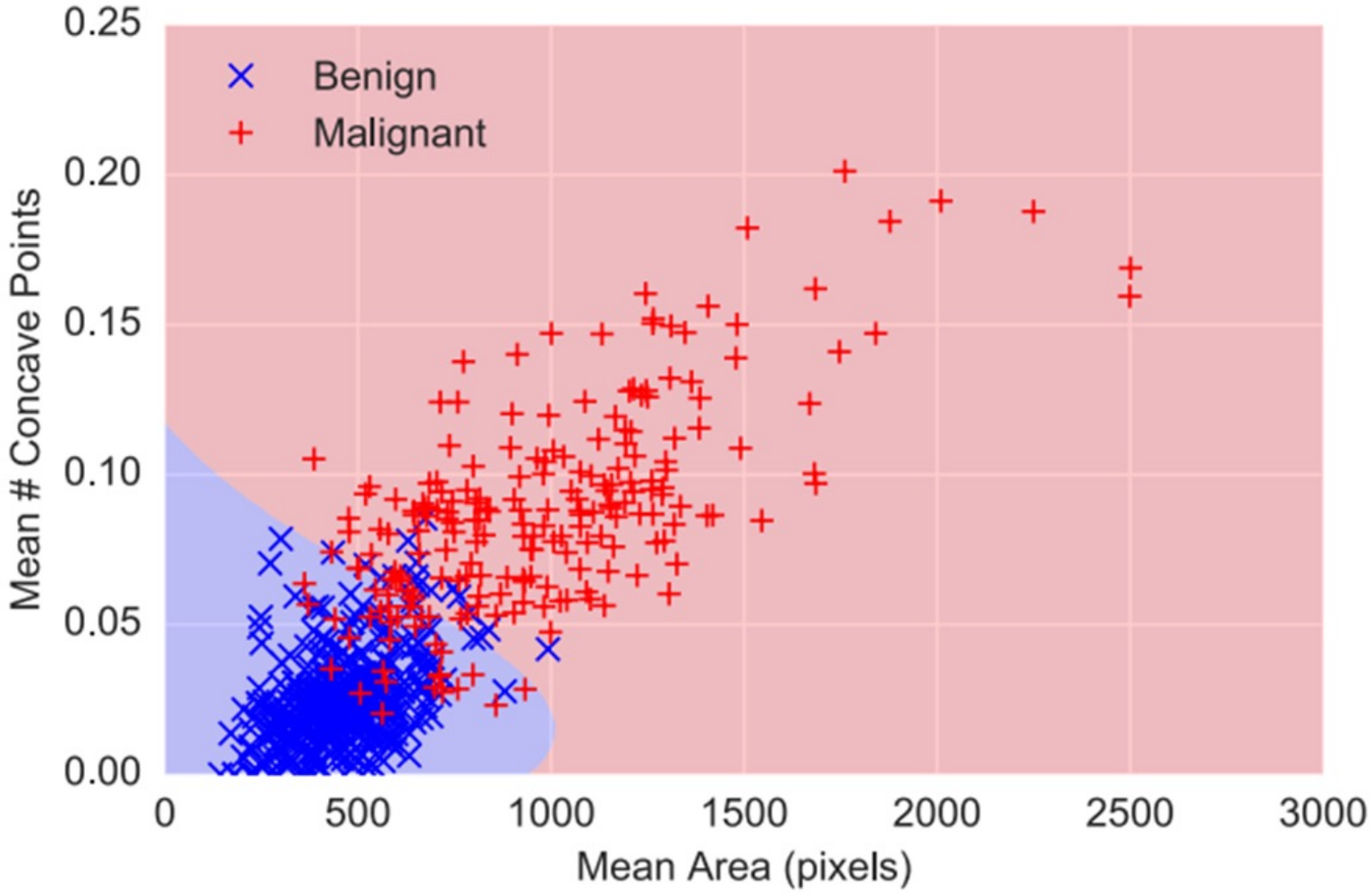
Polynomial features d = 10



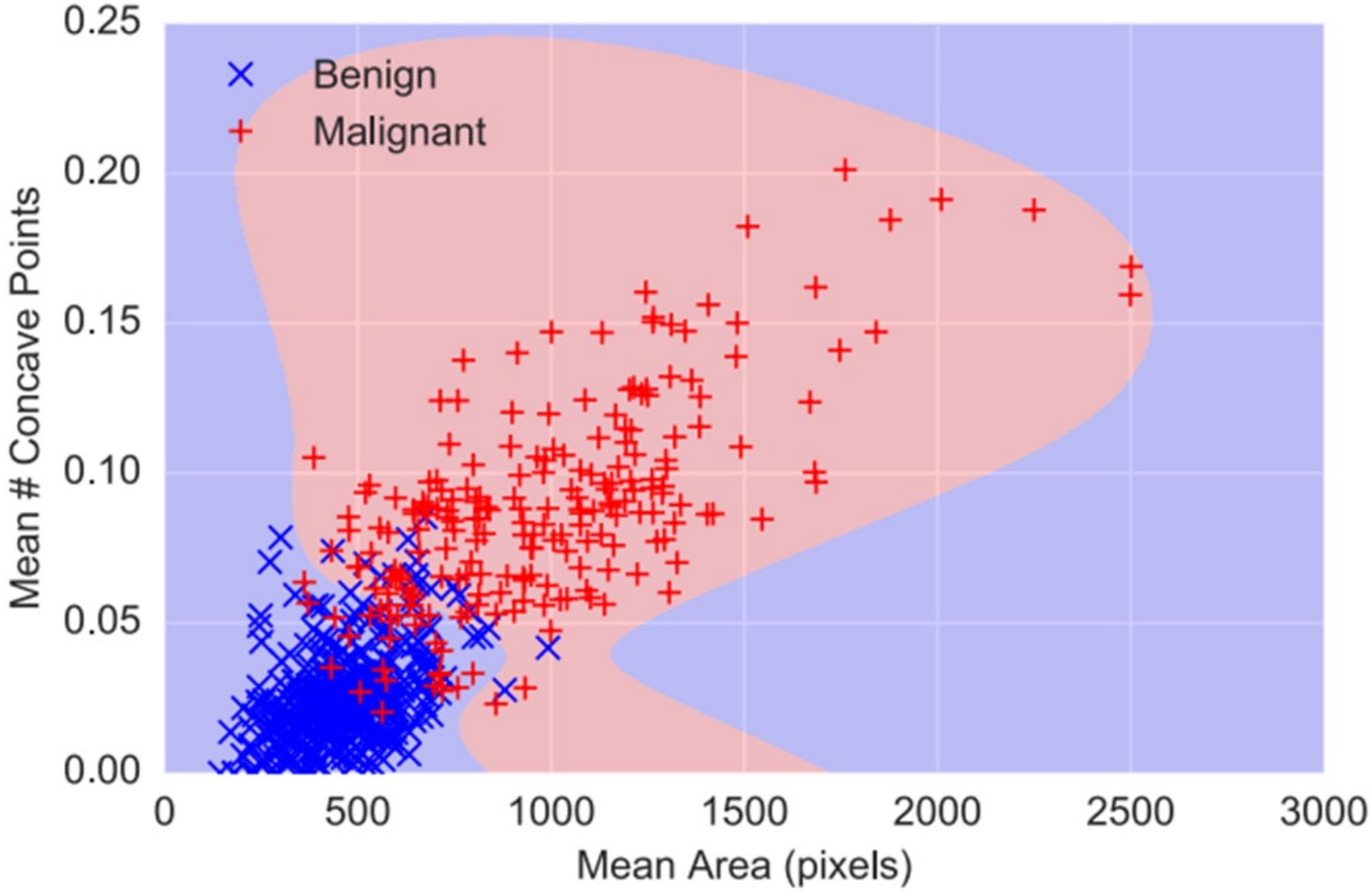
RBF features

- In the following, we assume that X has been normalized so that each feature lies between [−1, +1] (same as we did for polynomial features)
- We observe how the classifier changes as we change different parameters of the RBFs
- p will refer to total number of centers, d will refer to the number of centers along each dimensions, assuming centers form a regular grid (so since we have two raw inputs, $p = d^2$)

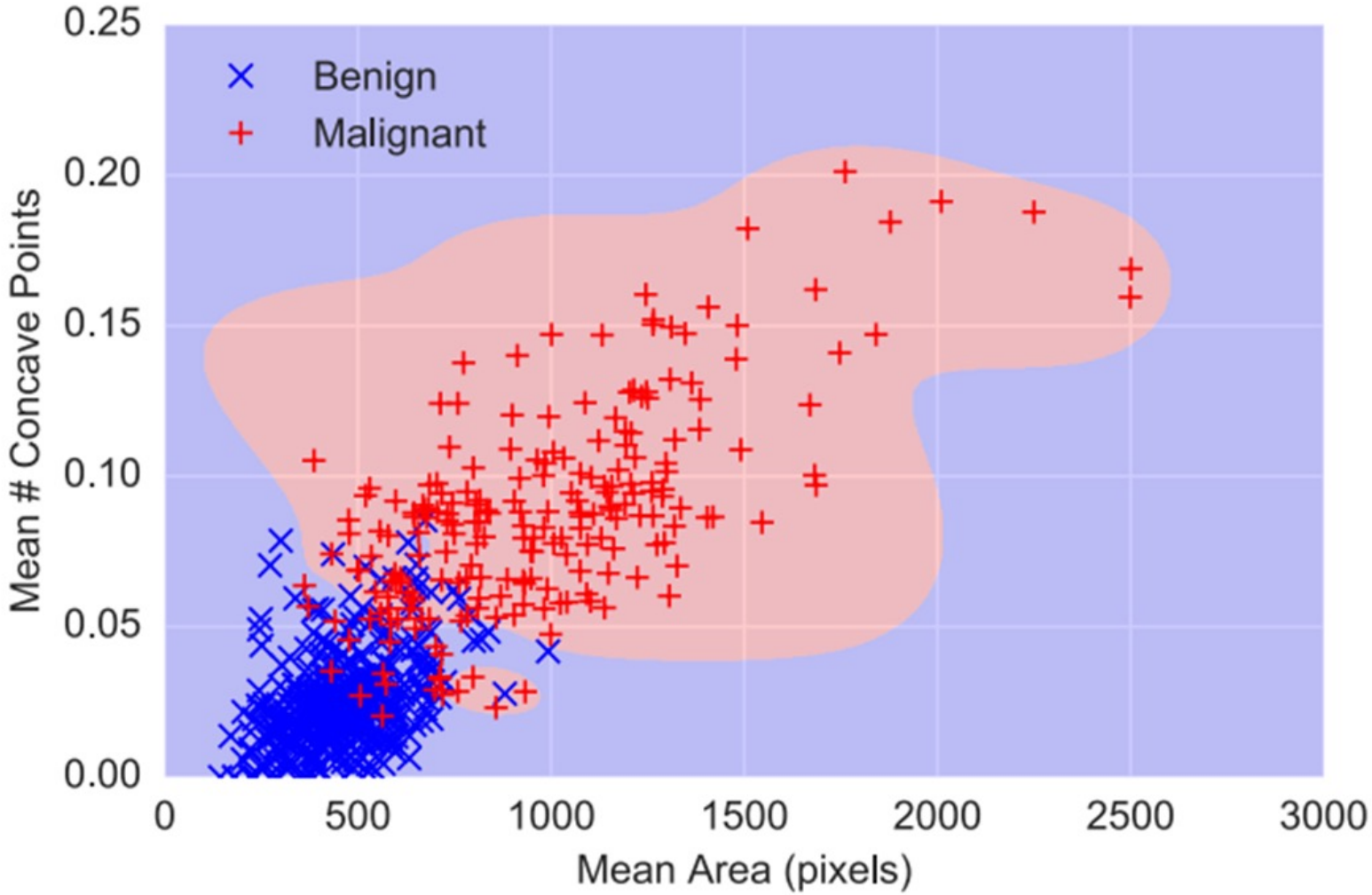
RBF features, d = 3, σ = 2/d



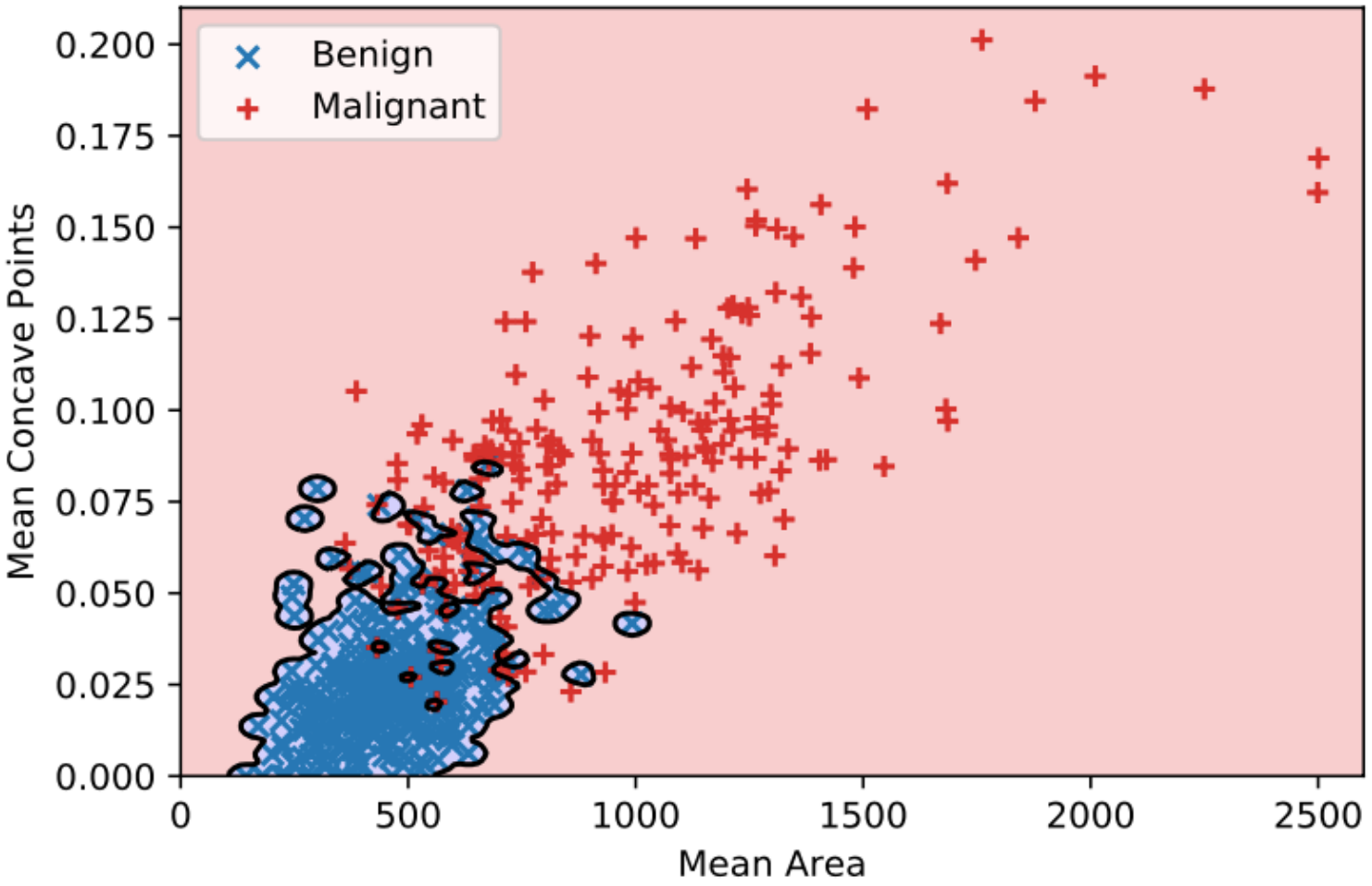
RBF features, d = 10, σ = 2/d



RBF features, d = 20, σ = 2/d



Overfitting taken to the extreme

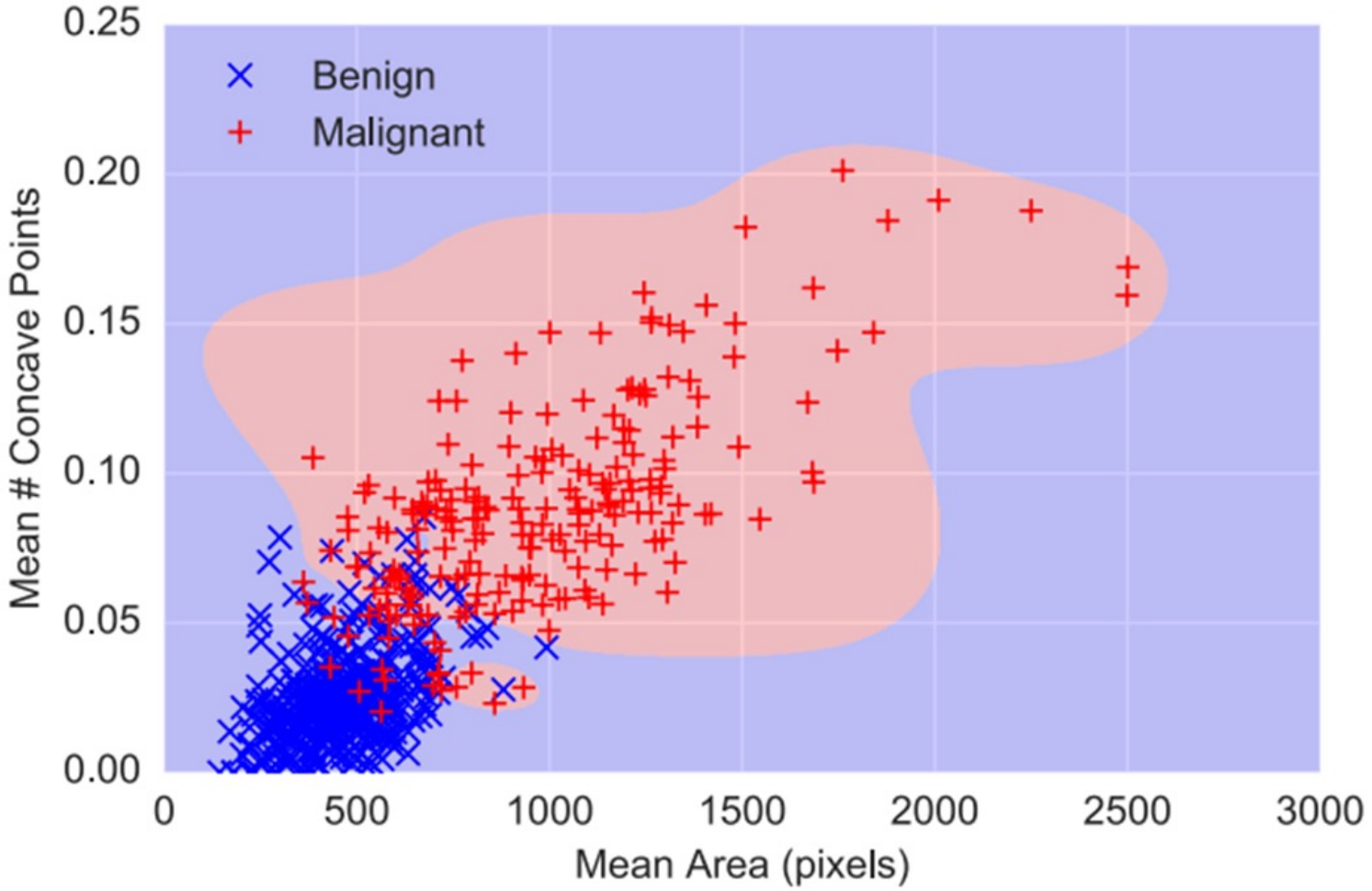


- Because the preceding examples are all quite similar (the decision boundary is roughly linear) let's introduce overfitting for illustrative purposes
- We make the bandwidth very small, and the regularization small, so that the classifier actually manages to get 100% accuracy on the training data

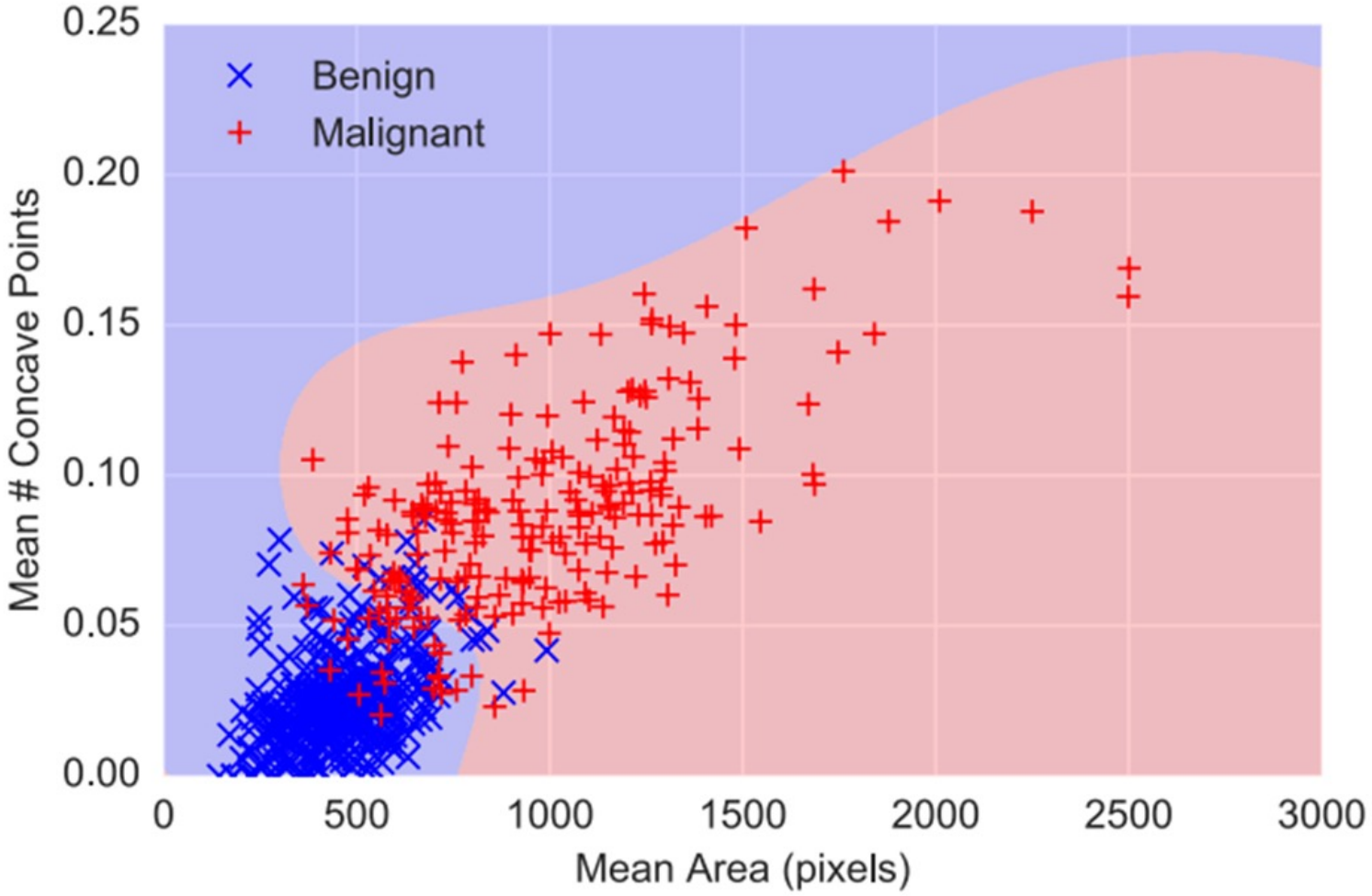
Model complexity and bandwidth

- As seen previously in a regression setting, we can control model complexity with RBFs in three ways: two of which we have already seen
  1. Choose number of RBF centers
  2. Increase/decrease regularization parameter
  3. Increase/decrease bandwidth

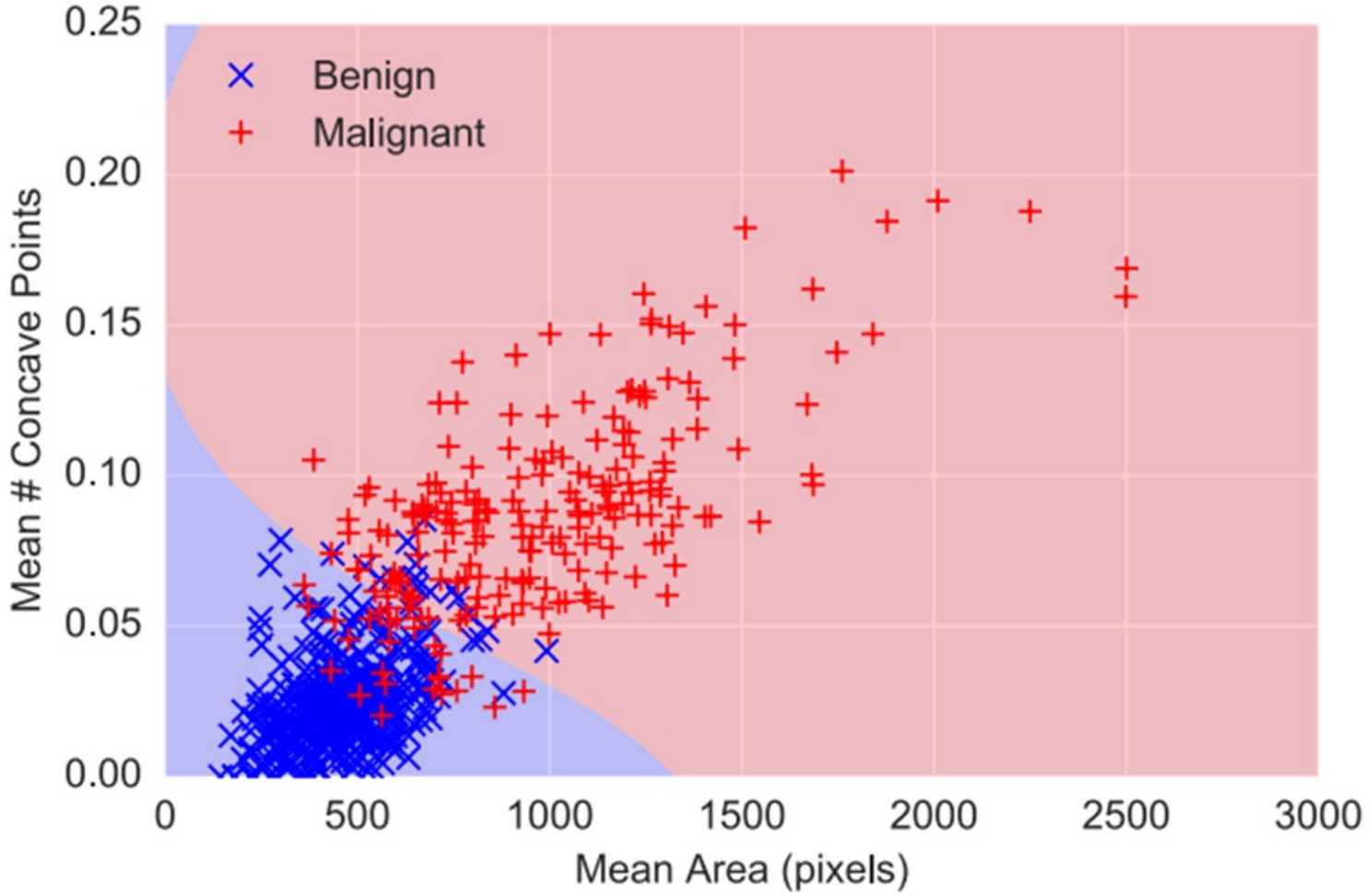
RBF features, d = 20, σ = 0.1



RBF features, d = 20, σ = 0.5



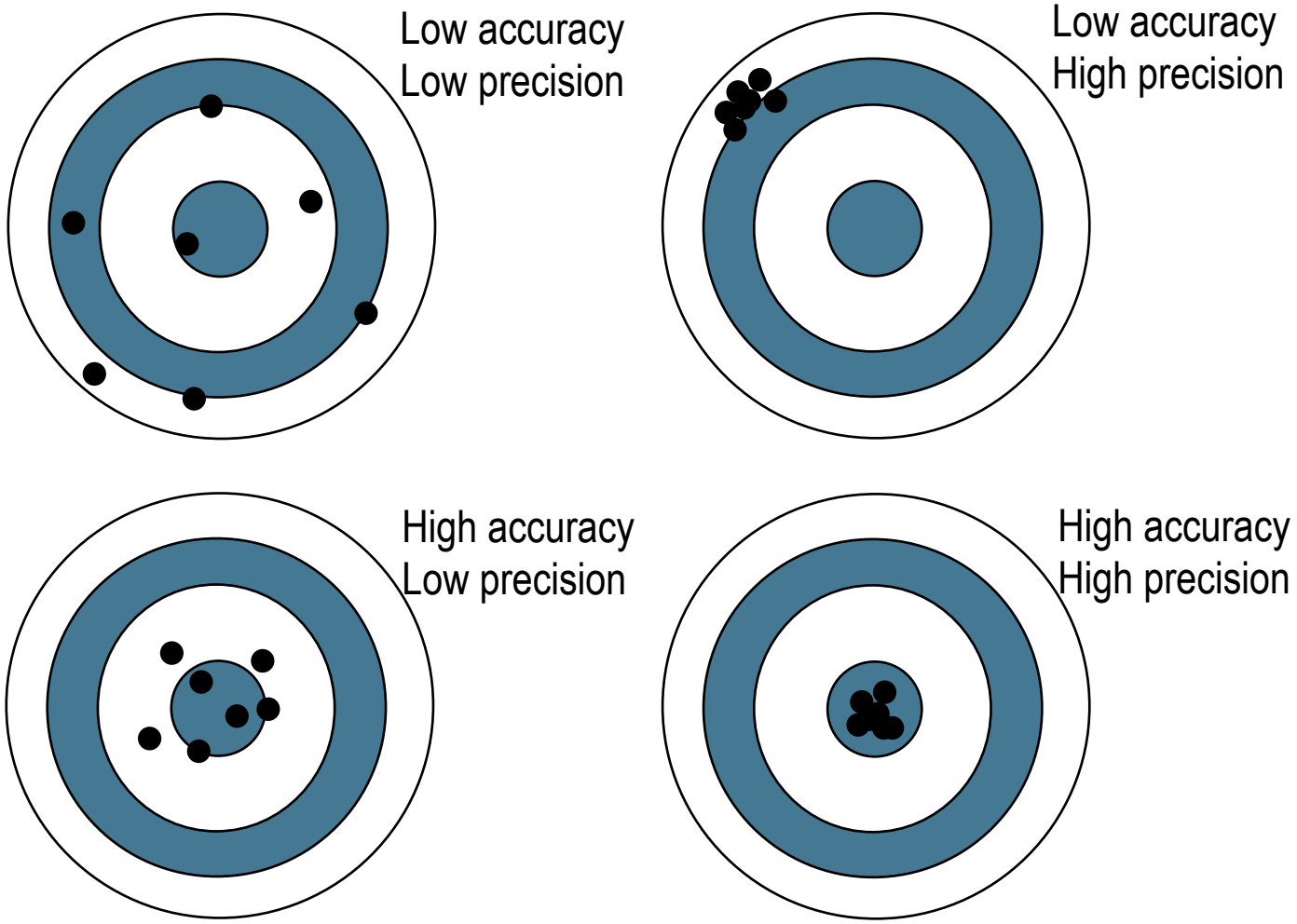
RBF features, d = 20, σ = 1.07 (median trick)



Agenda

- Naïve Bayes Classifier
- Classification Tree
- Non-Linear Classification
- **Evaluating Classification Models**

Precision vs. Accuracy



- **Accuracy**: ratio of correctly classified samples over the full sample size – Measure of how well the classifier performs in general
- **Precision**: ratio of true positives versus all positives as classified by the classifier – Measure of how precisely the classifier identifies positives

Classification metrics

- So far, we have considered accuracy (0/1 loss) as the primary method for evaluating classifiers
- However, sometimes the benefits for correctly classifying positive and negative examples are different, as are the costs for predicting a positive example to be negative, and vice versa
- In cancer dataset, it is a very different thing (in terms of real-world effects) to predict that an actually malignant tumor is benign, versus predicting a benign tumor is malignant

Confusion matrix

- A **confusion matrix** explicitly lists the number of examples for each actual class and each prediction
- Can compute these (and all associated metrics) on training / holdout / testing sets, but we'll just show examples on training sets here

Confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Common classification error metrics derived from the confusion matrix

- Several common metrics are associated with entries of the confusion matrix (TP = true positive, FP = false positive, TN = true negative, FN = false negative)
  - TP Rate (also called Recall) = $\frac{TP}{TP+FN}$
  - FP Rate = $\frac{FP}{FP+TN}$
  - Precision = $\frac{TP}{TP+FP}$
  - Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Different metrics can be standard for different domains

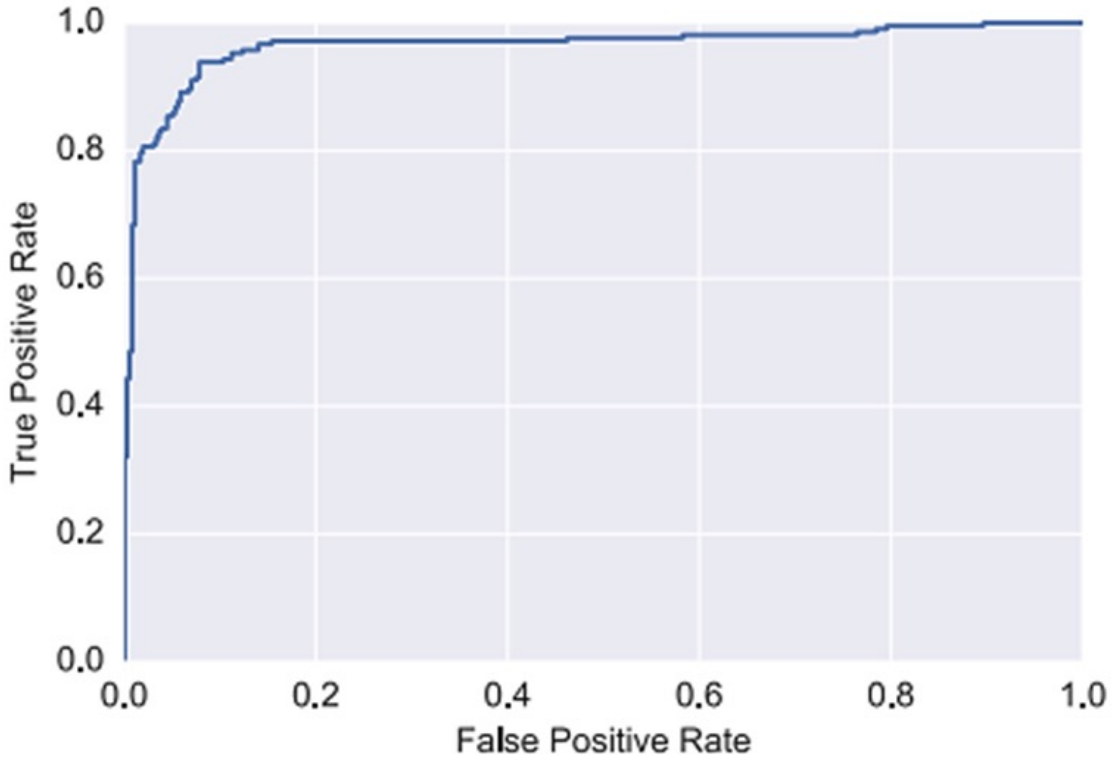
Confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Changing the prediction threshold

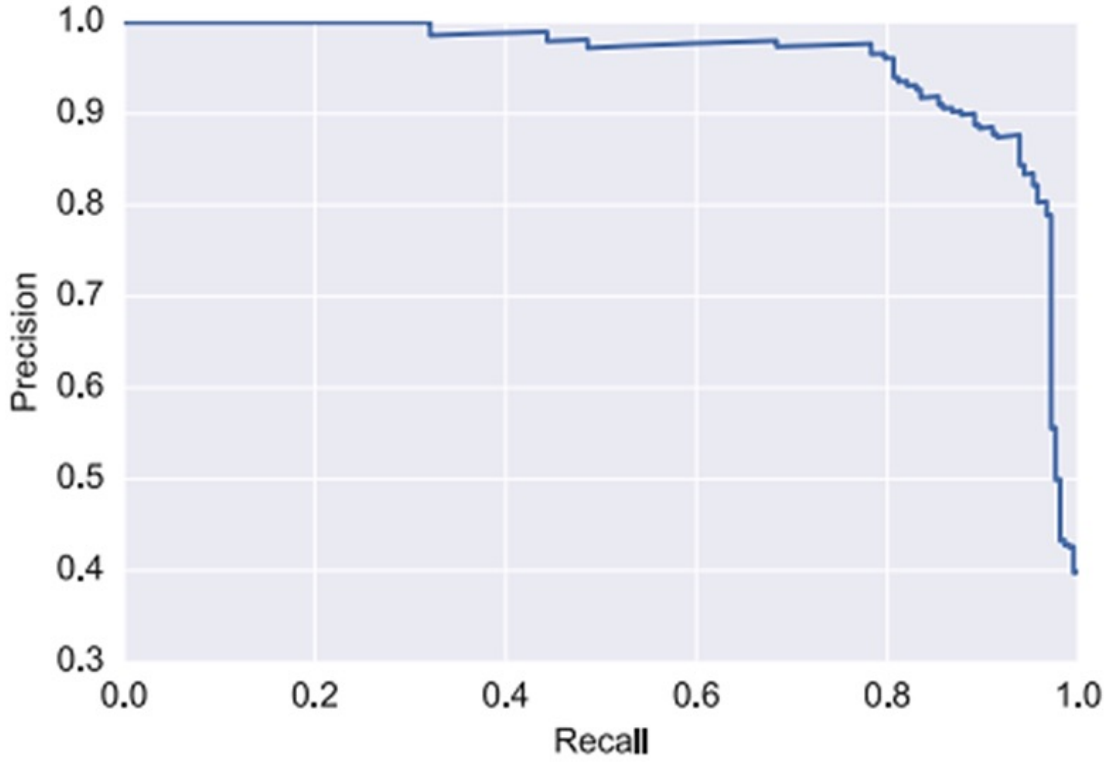
- Classifiers are implicitly trained around a “threshold” of zero (positive hypothesis means predict positive, negative means predict negative)
- But there is no reason to use only this threshold when we want to make predictions (may want to “overpredict” one class or the other)
- **Key idea:** by sorting the hypothesis function outputs, and adjusting the threshold at which we call something positive or negative, we can sweep out **all** possible classifications that a classifier can produce

ROC Curve



- If we plot the true positive rate versus the false positive rate for this procedure, we get a figure known as an ROC (receiver operating characteristic) curve

Precision recall curves



- We can perform similar operations for other metrics, to for e.g. a precision-recall curve (plot of recall vs. precision as threshold varies)
- Where
  - Recall = TP rate
  - Precision = TP/(TP+FP)

Contact

For general questions and enquiries on **research**, **teaching**, **job openings** and new **projects** refer to our website at www.is3.uni-koeln.de

For specific enquiries regarding this course contact us by sending an email to the **IS3 teaching** address at is3-teaching@wiso.uni-koeln.de