10-601
Machine Learning

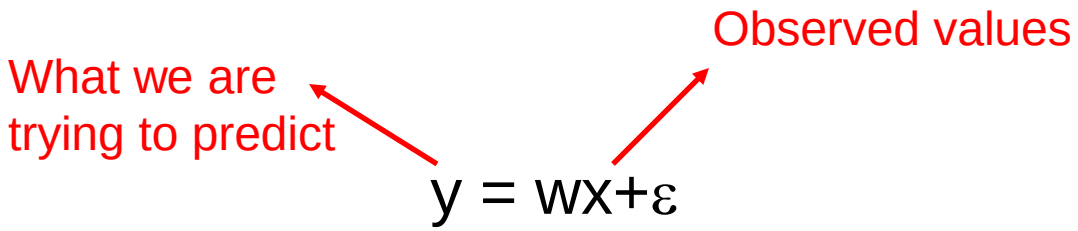
Linear regression
Logistic regression
Decision trees

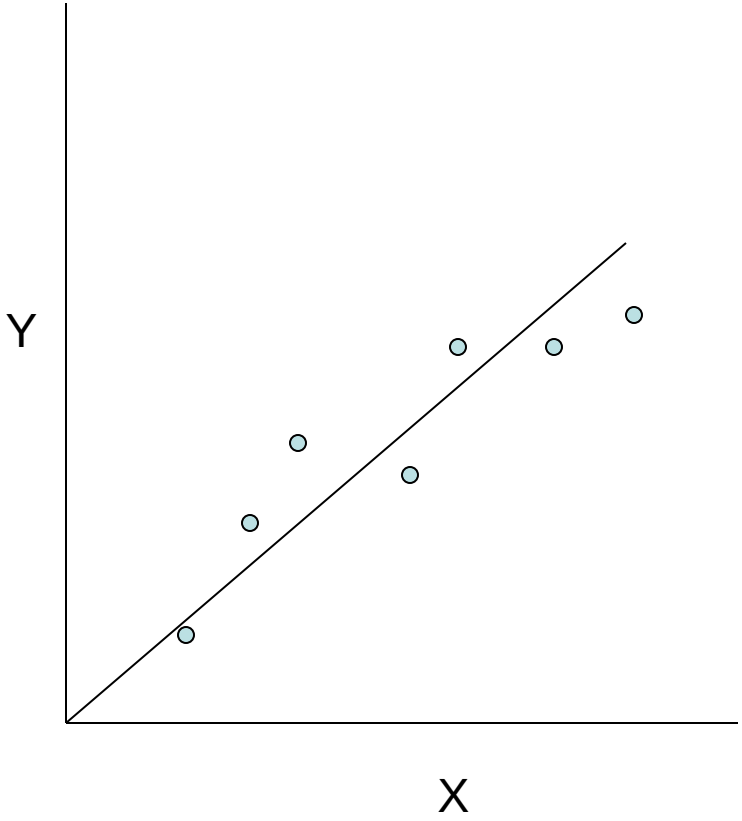
Material from Ziv Bar-Joseph's lecture slides and Christopher Bishop's textbook

Linear regression

Linear regression

- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

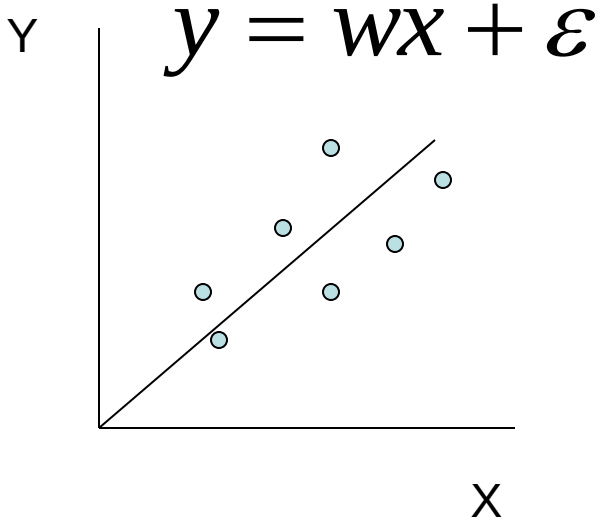




where w is a parameter and ε represents measurement or other noise

Linear regression

$$y = wx + \varepsilon$$



• Our goal is to estimate w from a training data of $<x_i,y_i>$ pairs

• This could be done using a least squares approach

$$\arg\min_w \sum_i (y_i - wx_i)^2$$

• Why least squares?

- minimizes squared distance between measurements and predicted line

- has a nice probabilistic interpretation

- easy to compute

If the noise is Gaussian with mean 0 then least squares is also the maximum likelihood estimate of w

Non-Linear basis function

- So far we only used the observed values
- However, linear regression can be applied in the same way to functions of these values
- As long as these functions can be directly computed from the observed values the parameters are still linear in the data and the problem remains a linear regression problem.
- What type of functions can we use?

Non-Linear basis function

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for j=0 … n

- Gaussian: $\phi_j(x) = \frac{(x - \mu_j)}{2\sigma_j^2}$

- Sigmoid: $\phi_j(x) = \frac{1}{1 + \exp(-s_j x)}$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

General linear regression problem

- Using our new notations for the basis function linear regression can be written as

$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

- Where $\phi_j(x)$ can be either x_j for multivariate regression or one of the non linear basis we defined
- Once again we can use ‘least squares’ to find the optimal solution.

LMS for the general linear regression problem

$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

Our goal is to minimize the following loss function:

$$J(\mathrm{w}) = \sum_i (y^i - \sum_j w_j \phi_j(x^i))^2$$

Moving to vector notations we get:

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2$$

We take the derivative w.r.t **w**

$$\frac{\partial}{\partial w} \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2 = 2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}}$$

Equating to 0 we get $2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}} = 0 \Rightarrow$

$$\sum_i y^i \phi(x^i)^{\mathrm{T}} = \mathrm{w}^{\mathrm{T}} \left[\sum_i \phi(x^i) \phi(x^i)^{\mathrm{T}} \right]$$

LMS for general linear regression problem

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2$$

We take the derivative w.r.t **w**

$$\frac{\partial}{\partial w} \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2 = 2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}}$$

Equating to 0 we get

$$2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}} = 0 \Rightarrow$$

$$\sum_i y^i \phi(x^i)^{\mathrm{T}} = \mathrm{w}^{\mathrm{T}} \left[\sum_i \phi(x^i) \phi(x^i)^{\mathrm{T}} \right]$$

Define:

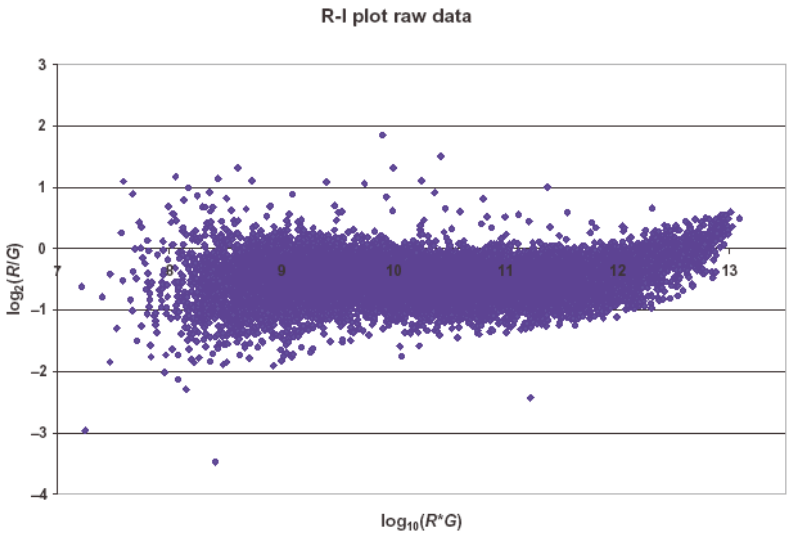
$$\Phi = \begin{pmatrix} \phi_0(x^1) & \phi_1(x^1) & \cdots & \phi_m(x^1) \\ \phi_0(x^2) & \phi_1(x^2) & \cdots & \phi_m(x^2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(x^n) & \phi_1(x^n) & \cdots & \phi_m(x^n) \end{pmatrix}$$

Then deriving w we get:

$$\mathrm{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{y}$$

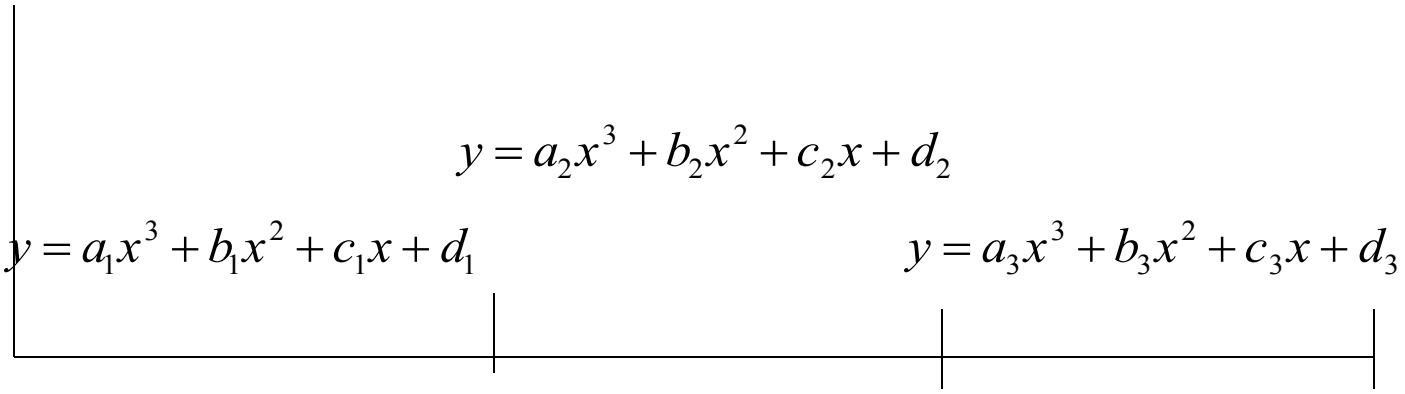
Other types of linear regression

- Linear regression is a useful model for many problems
- However, the parameters we learn for this model are **global**; they are the same regardless of the value of the input x
- Extension to linear regression adjust their parameters based on the region of the input we are dealing with



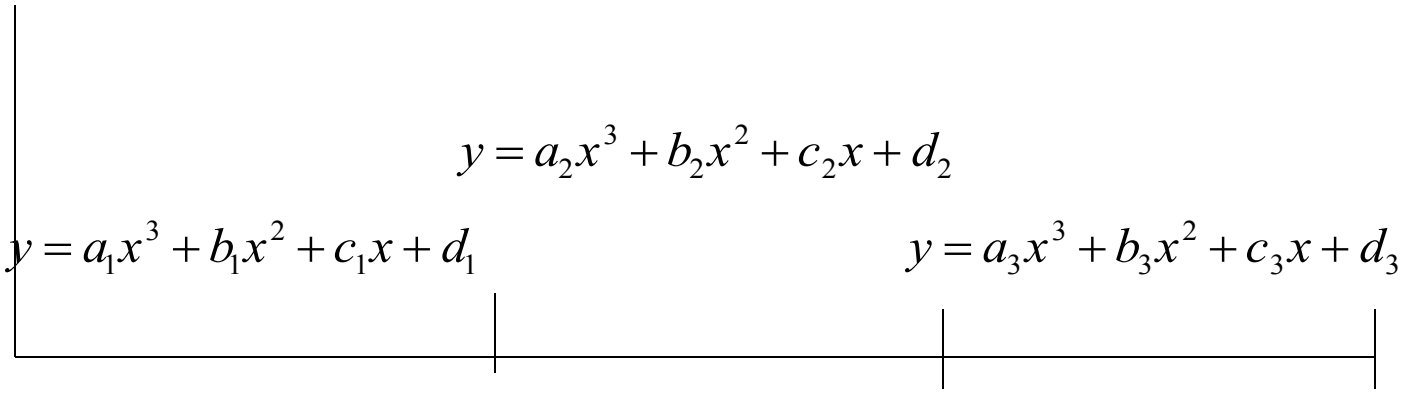
Splines

- Instead of fitting one function for the entire region, fit a set of piecewise (usually cubic) polynomials satisfying continuity and smoothness constraints.
- Results in smooth and flexible functions without too many parameters
- Need to define the regions in advance (usually uniform)



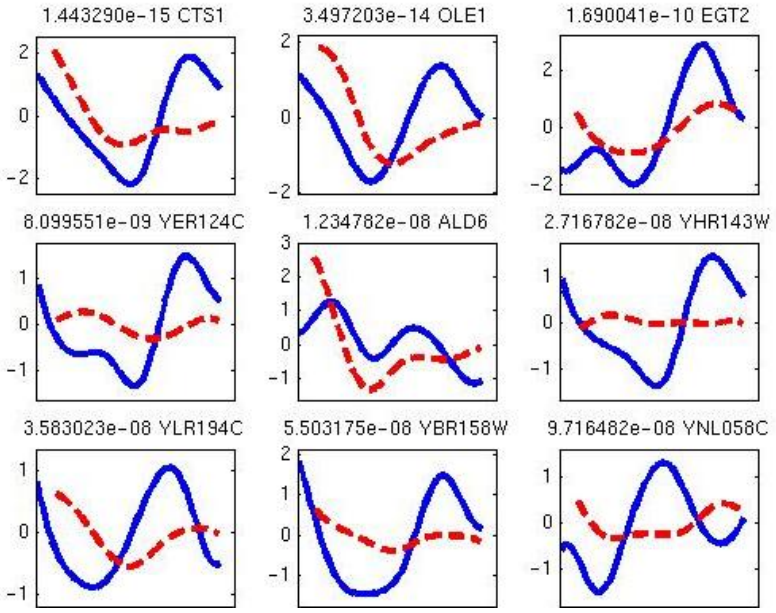
Splines

- The polynomials are not independent
- For cubic splines we require that they agree in the border point on the value, the values of the first derivative and the value of the second derivative
- How many free parameters do we actually have?



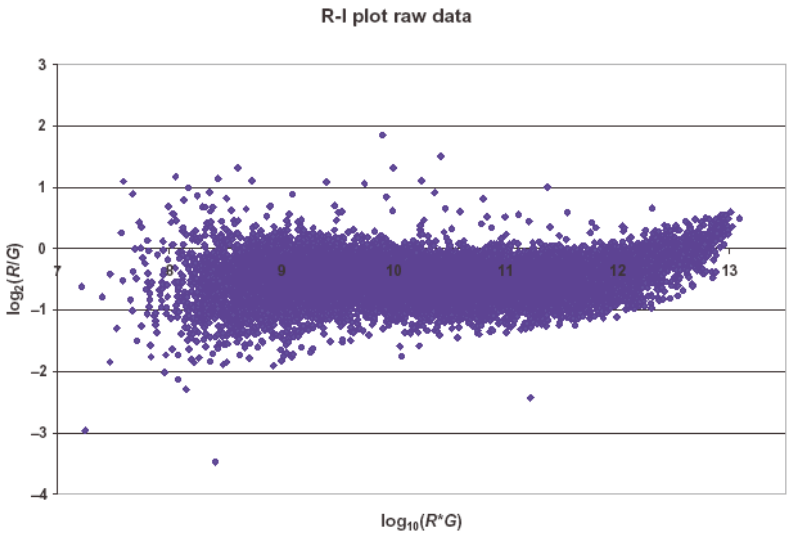
Splines

- Splines sometimes contain additional requirements for the first and last polynomial (for example, having them start at 0)
- Once Splines are fitted to the data they can be used to predict new values in the same way as regular linear regression, though they are limited to the support regions for which they have been defined
- Note the range of functions that can be displayed with relatively small number of polynomials (in the example I am using 5)



Locally weighted models

- Splines rely on a fixed region for each polynomial and the weight of all points within the region is the same.
- An alternative option is to set the region based on the density of the input data and have points closer to the point we are trying to estimate have a higher weight

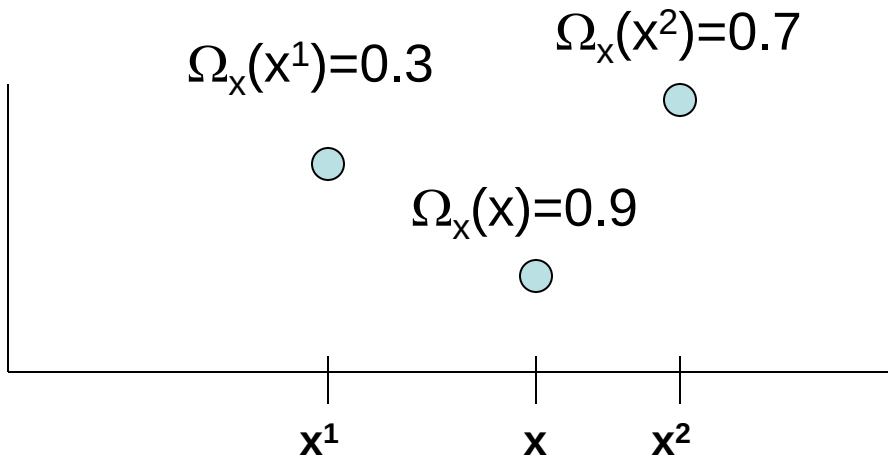


Weighted regression

- For a point x we use weight function Ω_x centered at x to assign weight to points in x's vicinity
- Next we solve the following *weighted* regression problem

$$\min_{\mathrm{w}} \sum_i \Omega_x(x^i)(y^i - \mathrm{w}^{\mathrm{T}}\phi(x^i))^2$$

- The solution is the same as our general solution (the weight is given for every input)



Determining the weights

- There are a number of ways to determine the weights
- One option is to use a Gaussian centered at *x*, such that

$$\Omega_x(x^i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x^i)^2}{2\sigma^2}}$$

σ^2 is a parameter that should be selected by the user

More on these weights when we discuss kernels

Bayesian linear regression

- Frequentist setting
  - Use MLE to calculate a single estimate of the weights as seen previously
- Bayesian setting
  - Calculate the posterior distribution of the weights

Bayesian linear regression

As we observe more data, our estimate of the weights becomes more sharply peaked

$$y = w_0 + w_1 x$$

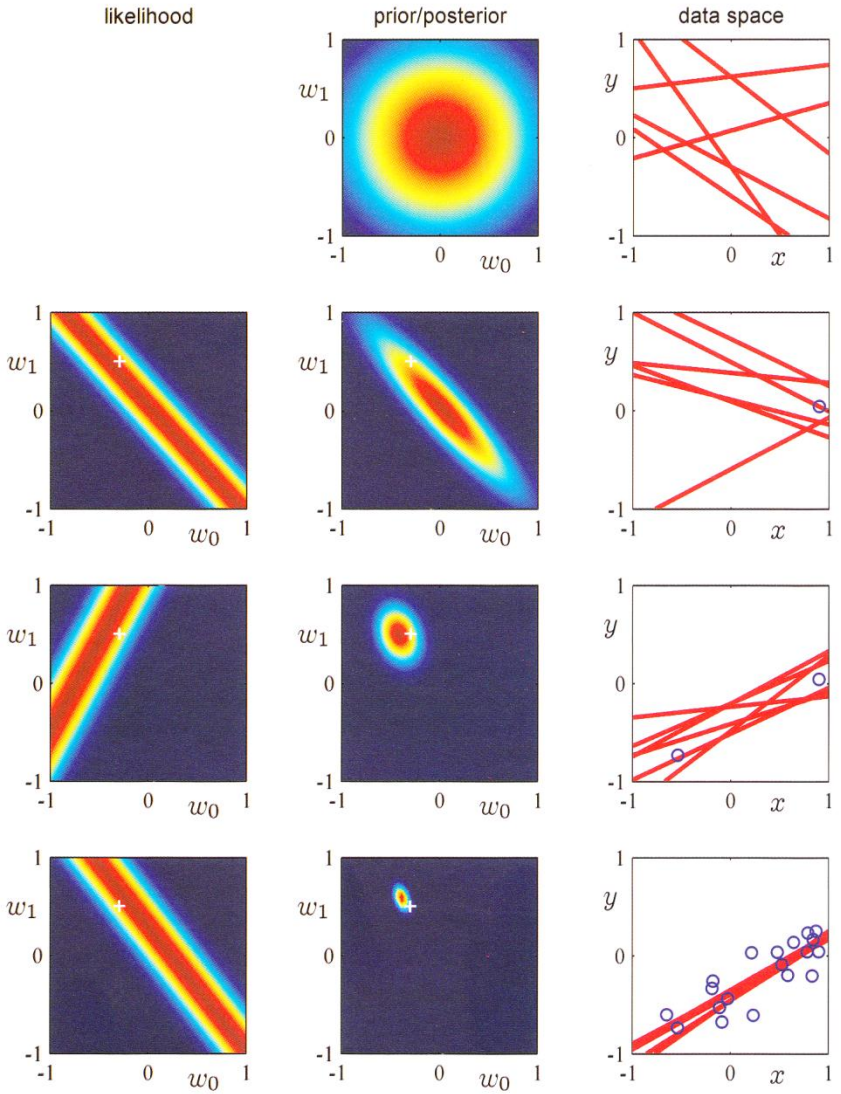


Image from Bishop

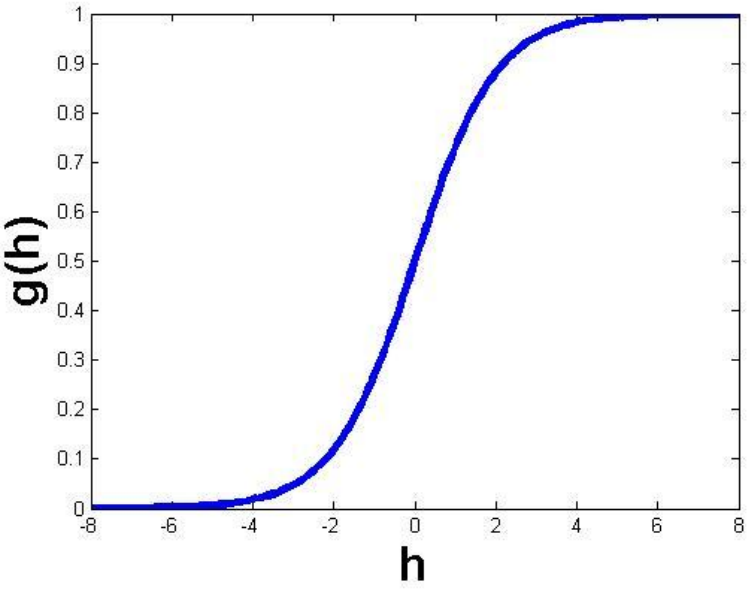
Logistic regression

The sigmoid function

$p(y \mid x;\theta)$

- To classify using regression models we replace the linear function with the sigmoid function:

Always between 0 and 1 $\longrightarrow$ $g(h) = \dfrac{1}{1+e^{-h}}$



- Using the sigmoid we set (for binary classification problems)

$$p(y=0 \mid x;\theta) = g(\mathrm{w}^{\mathrm{T}}x) = \frac{1}{1+e^{\mathrm{w}^{\mathrm{T}}x}}$$

$$p(y=1 \mid x;\theta) = 1 - g(\mathrm{w}^{\mathrm{T}}x) = \frac{e^{\mathrm{w}^{\mathrm{T}}x}}{1+e^{\mathrm{w}^{\mathrm{T}}x}}$$

Regularization

- Like with other data estimation problems, we may not have enough data to learn good models
- One way to overcome this is to ‘regularize’ the model, impose additional constraints on the parameters we are fitting.
- For example, lets assume that w_i comes from a Guassian distribution with mean 0 and variance σ^2 (where σ^2 is a user defined parameter): $w_i \sim N(0, \sigma^2)$
- In that case we have:

$$p(y=1,\theta \mid x) \propto p(y=1 \mid x;\theta)\, p(\theta)$$

Regularization

- If we regularize the parameters we need to take the prior into account when computing the posterior for our parameters

$$p(y=1,\theta \mid x) \propto p(y=1 \mid x;\theta)p(\theta)$$

- Here we use a Gaussian model for the prior.
- Thus, the log likelihood changes to :

$$LL(y;w \mid x) = \sum_{i=1}^{N} y^i \mathrm{w}^{\mathrm{T}} x^i - \ln(1+e^{w^T x^i}) - \sum_j \frac{w_j^2}{2\sigma^2}$$

After removing terms that are not dependent on w

- And the new update rule (after taking the derivative w.r.t. w_i) is:

$$w_j \leftarrow w_j + \varepsilon \sum_{i=1}^{N} x_j^i \{y^i - (1 - g(x^i;w))\} - \varepsilon \frac{w_j}{\sigma^2}$$

Also known as the MAP estimate

The variance of our prior model

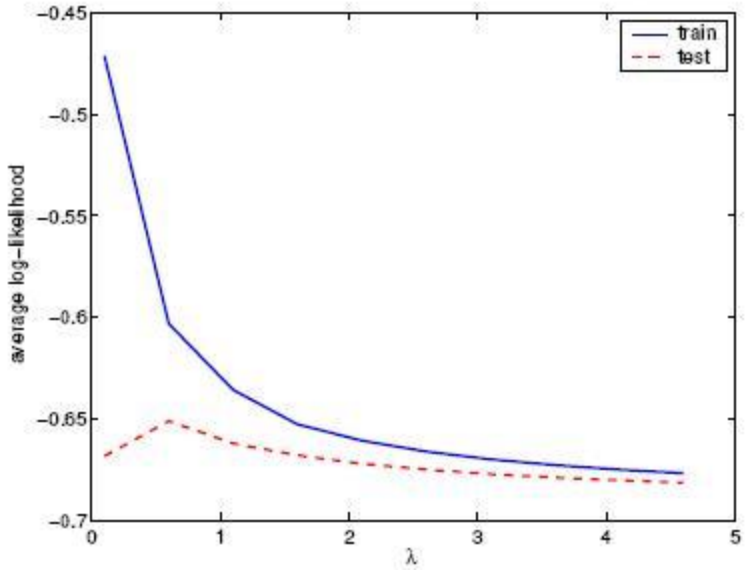
Regularization

- There are many other ways to regularize logistic regression
- The Gaussian model leads to an L2 regularization (we are trying to minimize the square value of *w*)
- Another popular regularization is an L1 which tries to minimize |w|

The importance of the regularization parameter

- Too small does not have a big impact
- Too large overrides the data
- An example of the training/test conditional log likelihoods as a function of the regularization parameter σ^2

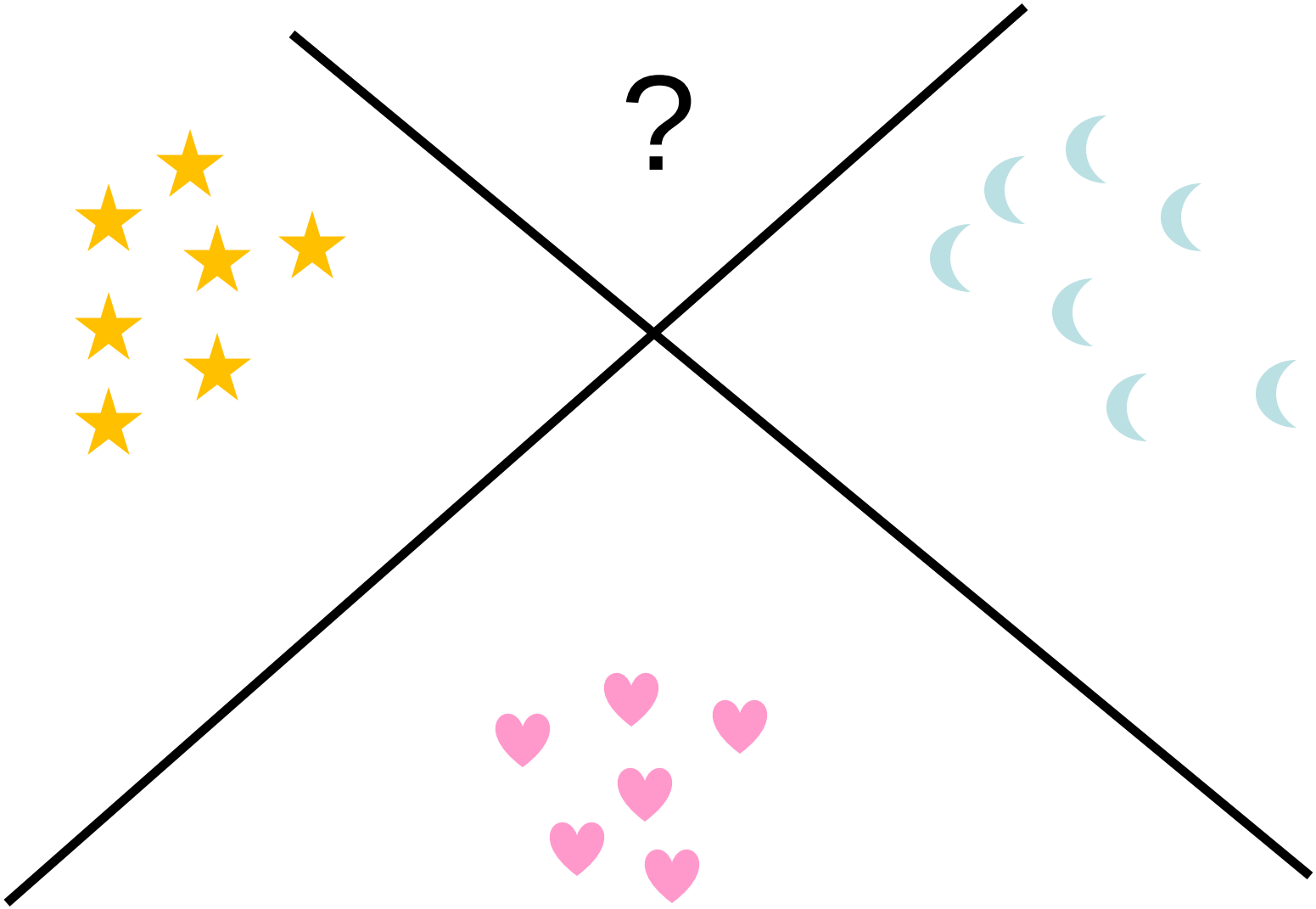
Average log likelihood for data only →



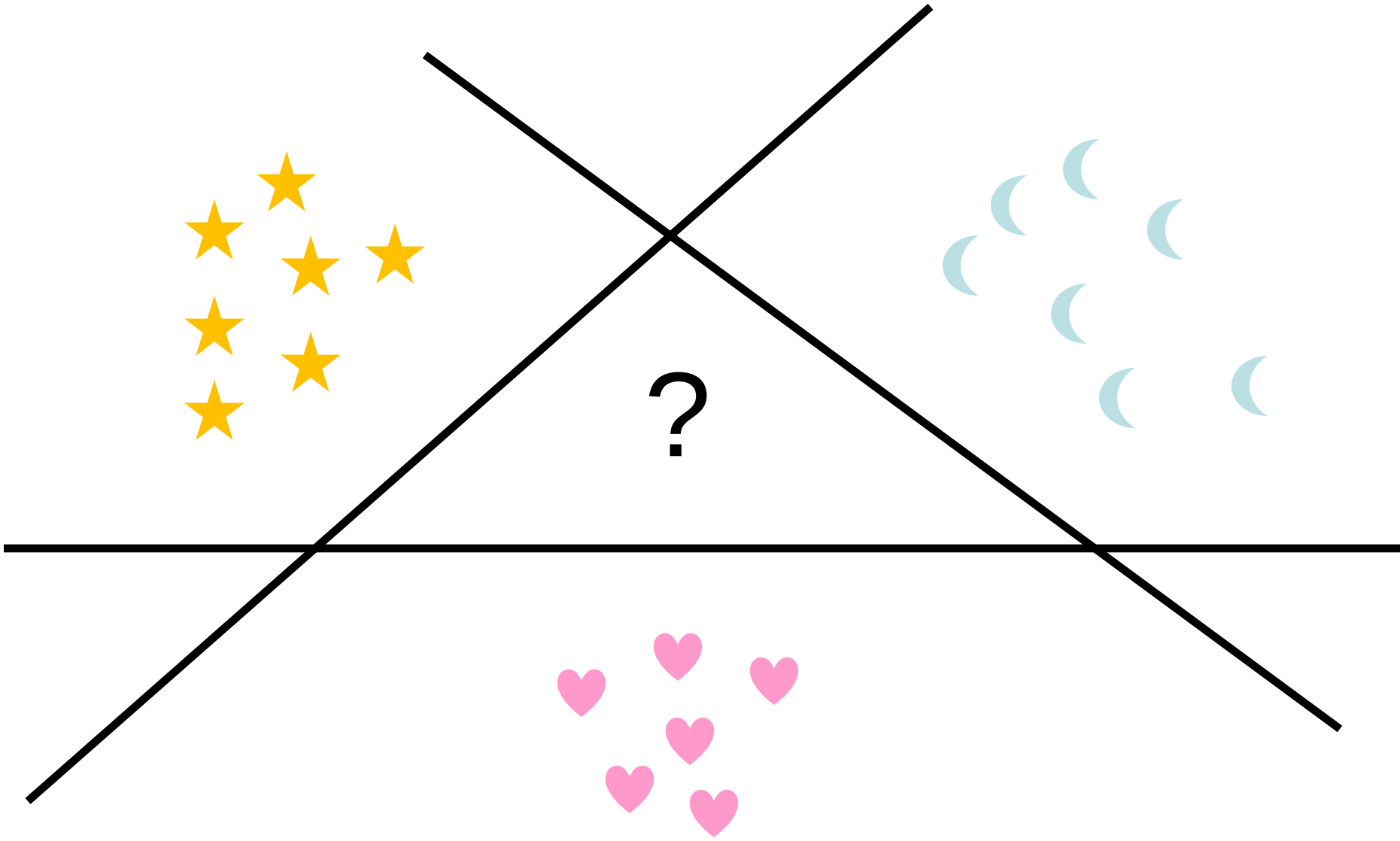
Logistic regression for more than 2 classes

- What if we have more than 2 classes?
- Can we use our existing binary logistic regression classifier?
- Should we?

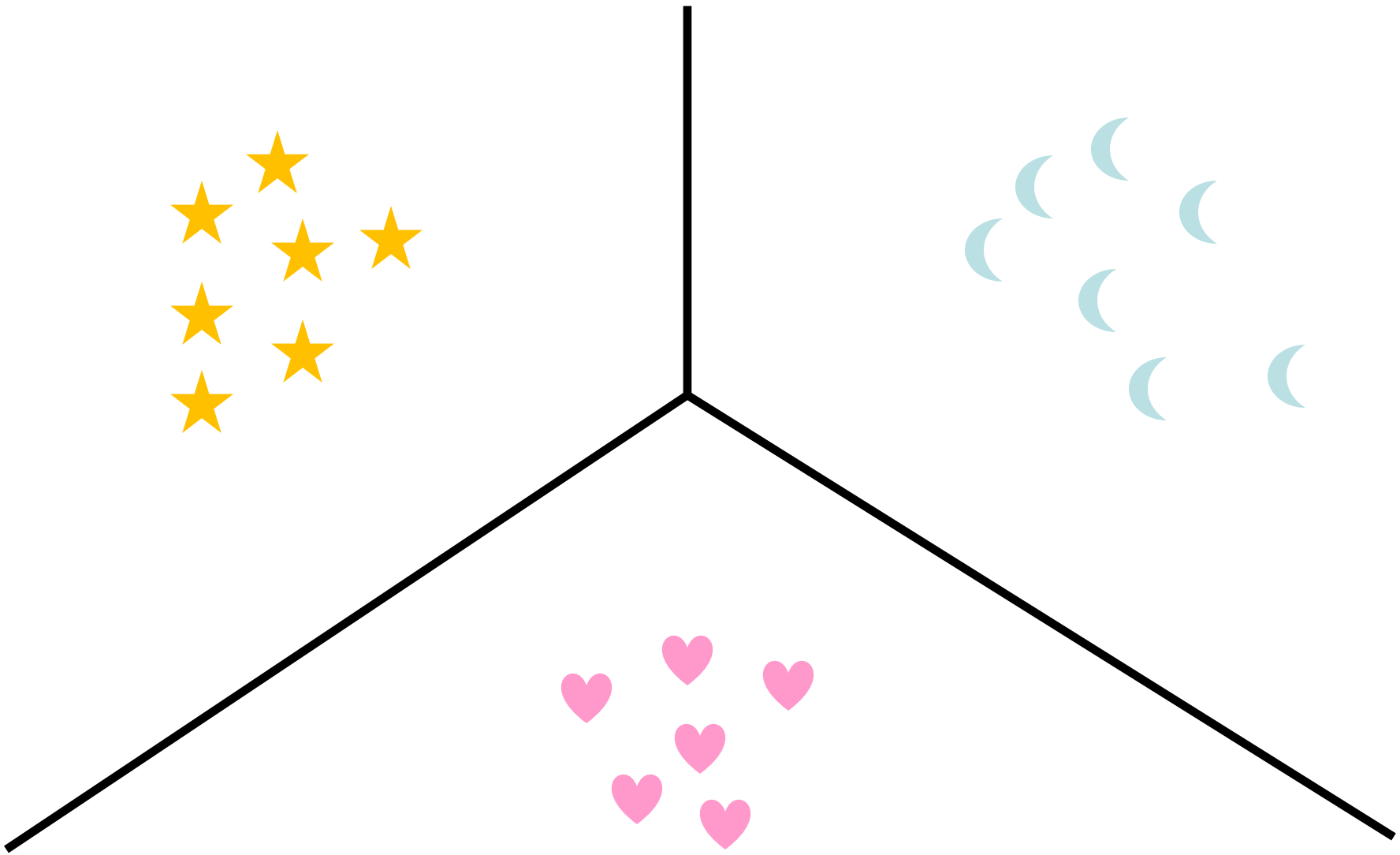
One versus all



One versus one



What we would like



Logistic regression for more than 2 classes

- Logistic regression can be used to classify data from more than 2 classes:
- For *i*<*k* we set

$$p(y=i \mid x;\theta) = g(w_{i0} + w_{i1}x_1 + \ldots + w_{id}x_d) = g(\mathrm{w}_\mathrm{i}^\mathrm{T} x)$$

where $g(z_i) = \dfrac{e^{z_i}}{1+\sum_{j=1}^{k-1} e^{z_j}}$ $z_i = w_{i0} + w_{i1}x_1 + \ldots + w_{id}x_d$

- And for k we have $p(y=k \mid x;\theta) = 1 - \sum_{i=1}^{k-1} p(y=i \mid x;\theta) \Rightarrow$

$$p(y=k \mid x;\theta) = \frac{1}{1+\sum_{j=1}^{k-1} e^{z_j}}$$

Logistic regression for more than 2 classes

- Logistic regression can be used to classify data from more than 2 classes:
- For *i*<*k* we set

$$p(y=i \mid x;\theta) = g(w_{i0} + w_{i1}x_1 + \ldots + w_{id}x_d) = g(\mathrm{w}_\mathrm{i}^\mathrm{T} x)$$

where $g(z_i) = \dfrac{e^{z_i}}{1+\sum_{j=1}^{k-1} e^{z_j}}$ $z_i = w_{i0} +$

Binary logistic regression is a special case of this rule

- And for k we have $p(y=k \mid x;\theta) = 1 - \sum_{i=1}^{k-1} p(y=i \mid x;\theta) \Rightarrow$

$$p(y=k \mid x;\theta) = \frac{1}{1+\sum_{j=1}^{k-1} e^{z_j}}$$

Update rule for logistic regression with multiple classes

$$\frac{\partial}{\partial w_{m,j}} l(w) = \sum_{i=1}^{N} x_j^i \{\delta_m(y^i) - p(y^i = m \mid x^i; w)\}$$

Where $\delta(y^i)=1$ if $y^i=m$ and $\delta(y^i)=0$ otherwise

The update rule becomes:

$$w_{m,j} \leftarrow w_{m,j} + \varepsilon \sum_{i=1}^{N} x_j^i \{\delta_m(y^i) - p(y^i = m \mid x^i; w)\}$$

Additive models

- Similar to what we did with linear regression we can extend logistic regression to other transformations of the data

$$p(y=1 \mid x;w) = g(w_{i0} + w_1\phi_1(x) + \ldots + w_d\phi_d(x))$$

- As before, we are free to choose the basis functions

Logistic regression's name

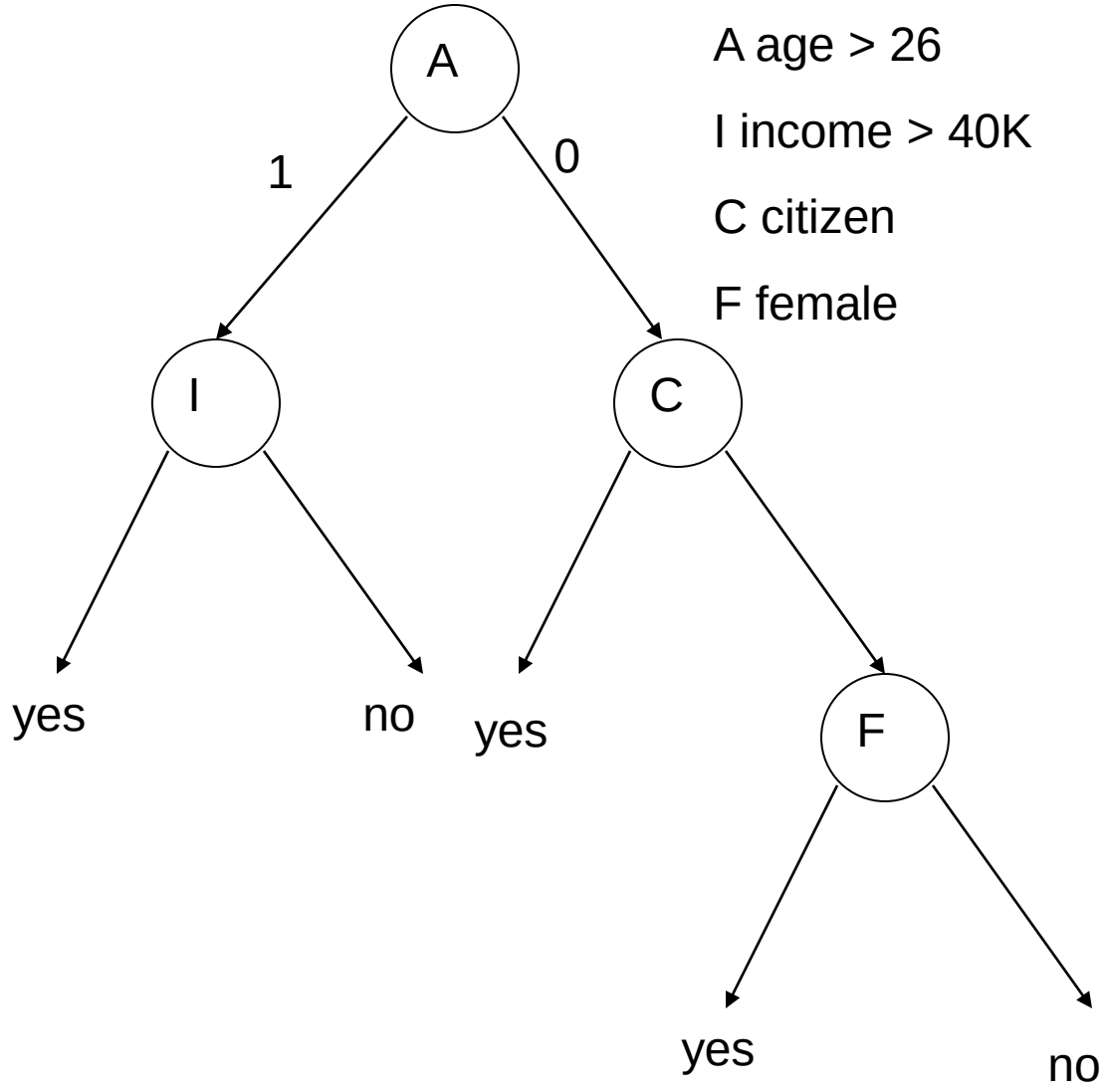
- The name comes from the **logit** transformation:

$$\log \frac{p(y=i \mid x;\theta)}{p(y=k \mid x;\theta)} = \log \frac{g(z_i)}{g(z_k)} = w_{i0} + w_{i1}x_1 + \cdots + w_{id}x_d$$

Decision trees

Structure of a decision tree

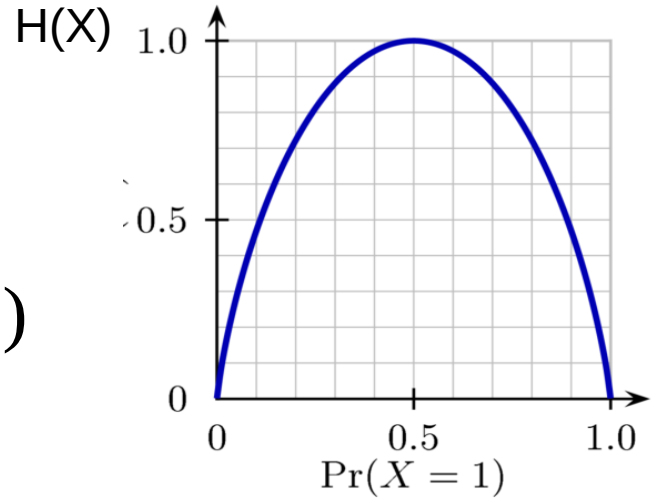
- Internal nodes correspond to attributes (features)
- Leafs correspond to classification outcome
- Edges denote assignment



Building a decision tree

```
Function BuildTree(n,A)    // n: samples (rows), A: attributes
  If empty(A) or all n(L) are the same
      status = leaf
      class = most common class in n(L)
  else
      status = internal
      a ⇐ bestAttribute(n,A)
      LeftNode = BuildTree(n(a=1), A \ {a})
      RightNode = BuildTree(n(a=0), A \ {a})
   end
end
```

Entropy



- Definition

$$H(X) = \sum_i - p(X = i) \log_2 p(X = i)$$

- So, if P(X=1) = 1 then

$$H(X) = -p(x = 1) \log_2 p(X = 1) - p(x = 0) \log_2 p(X = 0)$$
$$= -1 \log 1 - 0 \log 0 = 0$$

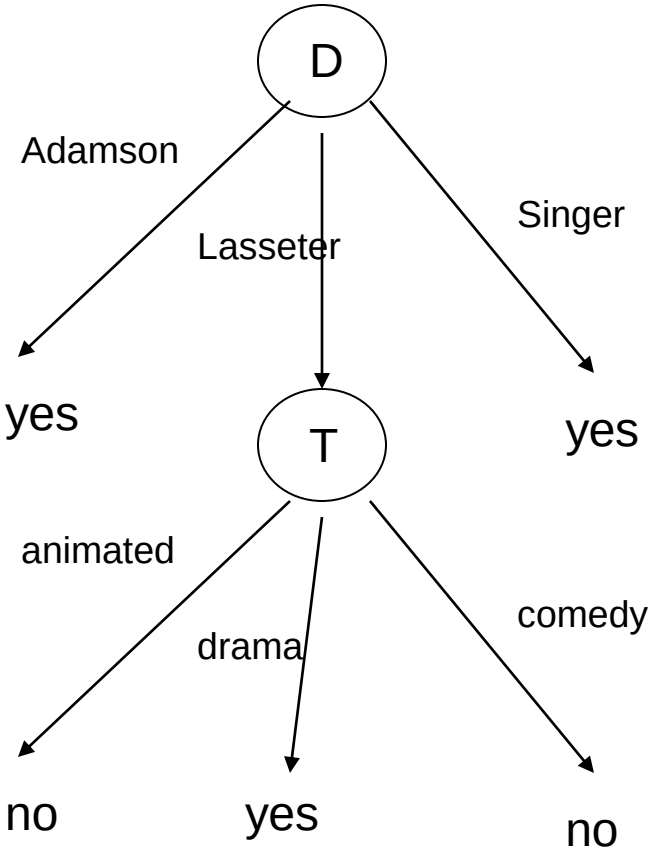
- If P(X=1) = .5 then

$$H(X) = -p(x = 1) \log_2 p(X = 1) - p(x = 0) \log_2 p(X = 0)$$
$$= -.5 \log_2 .5 - .5 \log_2 .5 = -\log_2 .5 = 1$$

Conditional entropy

- We can generalize the conditional entropy idea to determine H(Li | Le)
- That is, what is the expected number of bits we need to transmit if both sides know the value of Le for each of the records (samples)
- Definition: $H(X \mid Y) = \sum_i P(Y = i) H(X \mid Y = i)$

Decision tree from class



Movie	Type	Length	Director	Famous actors	Liked ?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	animated	long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
M7	animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

Additional points

- The algorithm we gave reaches homogonous nodes (or runs out of attributes)
- This is dangerous: For datasets with many (non relevant) attributes the algorithm will continue to split nodes
- This will lead to overfitting!

Avoiding overfitting: Tree pruning

- Split data into train and test set
- Build tree using training set

- For all internal nodes (starting at the root)
  - remove sub tree rooted at node
  - assign class to be the most common among training set
  - check test data error
    - if error is lower, keep change
    - otherwise restore subtree, repeat for all nodes in subtree

Continuous values

- Either use threshold to turn into binary or discretize
- Its possible to compute information gain for all possible tresholds (there are a finite number of training samples)
- Harder if we wish to assign more than two values (can be done recursively)

The ‘best’ classifier

- There has been a lot of interest lately in decision trees.
- They are quite robust, intuitive and, surprisingly, very accurate

Ranking classifiers

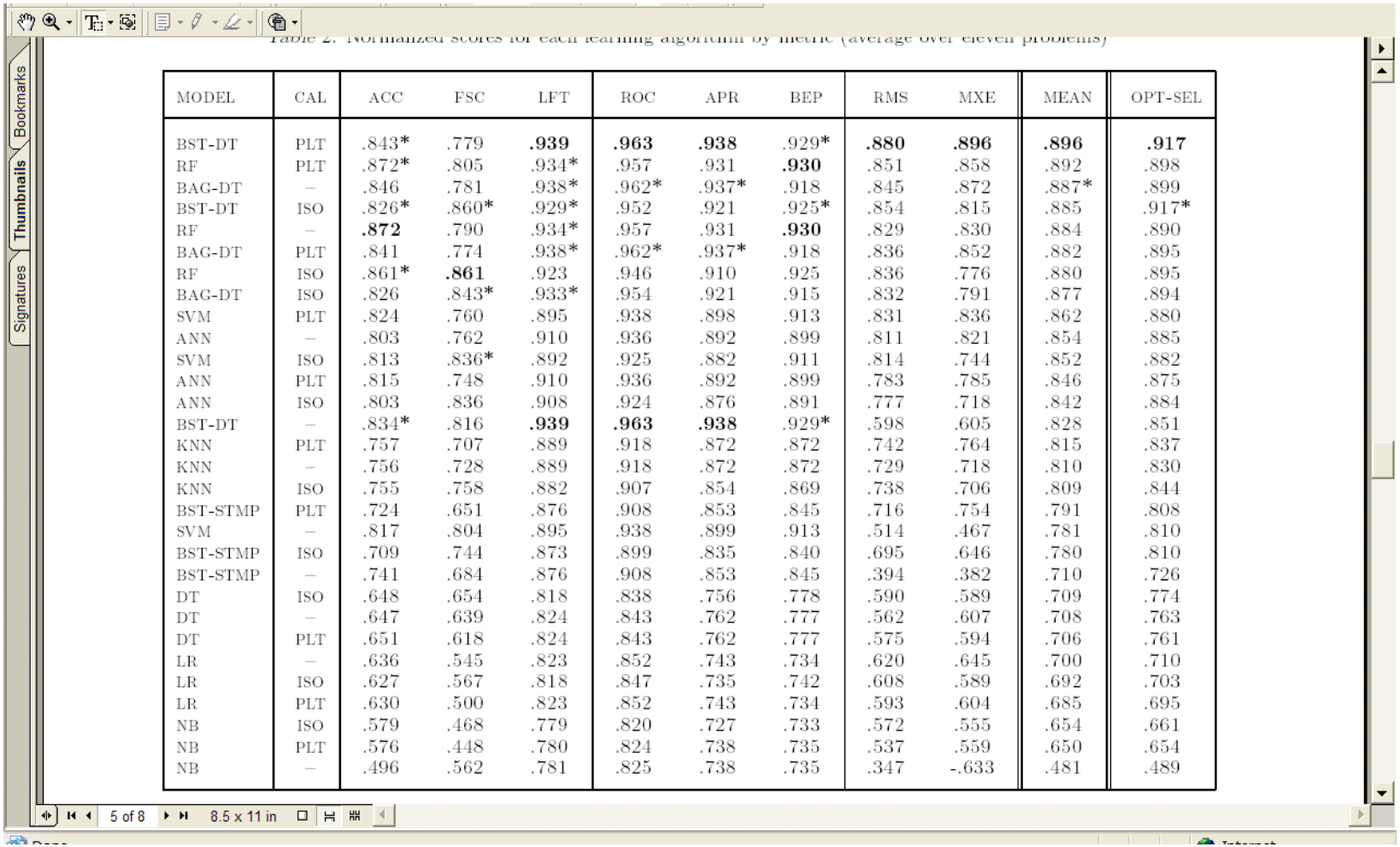


Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489

Rich Caruana & Alexandru Niculescu-Mizil, An Empirical Comparison of Supervised Learning Algorithms, ICML 2006

Miscellaneous

We also discussed…

- Using only k nearest neighbors in the locally weighted linear regression weight function
- Differences between Frequentist and Bayesian statistics
- Why the decision tree algorithm from class (ID3) doesn't guarantee the shortest possible consistent tree
- Rule-based decision tree pruning
- Splitting continuous values multiple times during decision tree building
- Aggregating predictions from multiple weak classifiers, especially decision trees (more on this in future lectures)
- Visualizing decision boundaries in 2d feature space for classifiers we have studied

Problem set 2 clarifications

- 1.1.1 – When considering how many distinct thresholds are needed, keep in mind we're only interested in thresholds that could potentially yield the maximum information gain
- 4.4 – The fold vector is used to divide the data into train and test sets. Your outer loop will look like:

```
for f = 1 to 10
    train = data where fold[:] != f;
    test = data where fold[:] == f;
    …
end
```