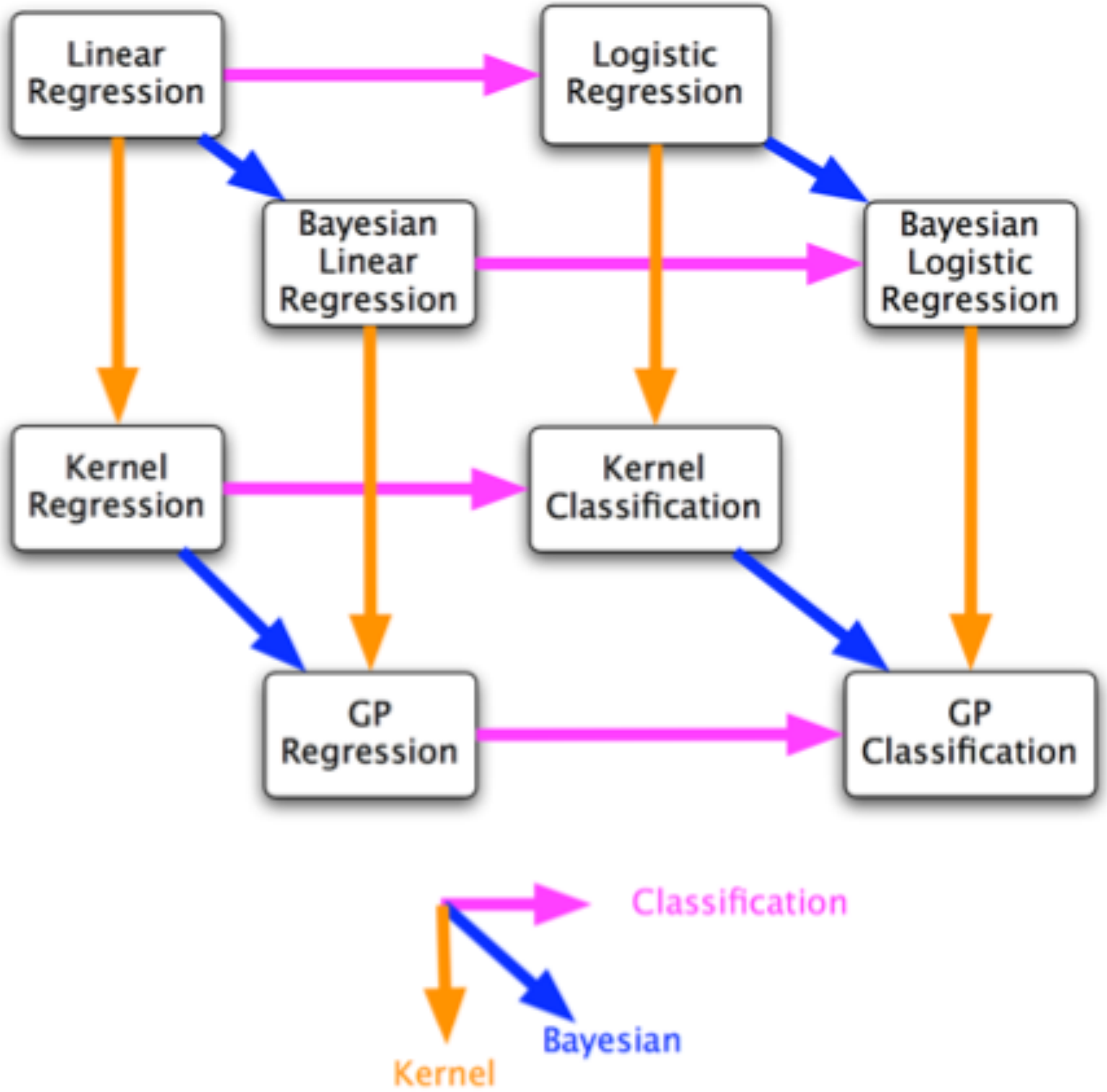
LECTURE 13: CLASSIFICATION

- Classification goal is to classify data into discrete classes
- This appears to be very different from regression and indeed some methods are specific to classification
- But classification can also be achieved as a rather simple extension of regression, starting from logistic regression, and continuing to Bayesian logistic regression and Gaussian process classification. From there there it is a simple connection to SVM (and on to neural networks…)

Connections between regression/ classification models



Bishop 2006

Logistic Regression

- We want a probabilistic description of discrete (class) outcomes. For simplicity we start with binary outcomes and we would like to predict outcome based on input data
- Example: Will tomorrow snow? Binary outcome: Yes (Y=1) or No (Y=0). Input data X: weather report, temperature… Y is called indicator variable
- Saying yes or no is crude: let's give it a probability $p(Y|X)$: e.g. tomorrow will snow with 60% prob.
- $E(Y) = p(Y=1)$ *and* $E(Y|X=x) = p(Y=1|X=x)$
- Conditional expectation is conditional probability of indicator: this means we can do the usual regression of some objective function to get E(Y). But since we want this to also be a conditional probability we need to map it in an interval between 0 and 1
- Probabilities are normalized: $P(Y=0|x)=1-P(Y=1|x)$

Logistic Regression

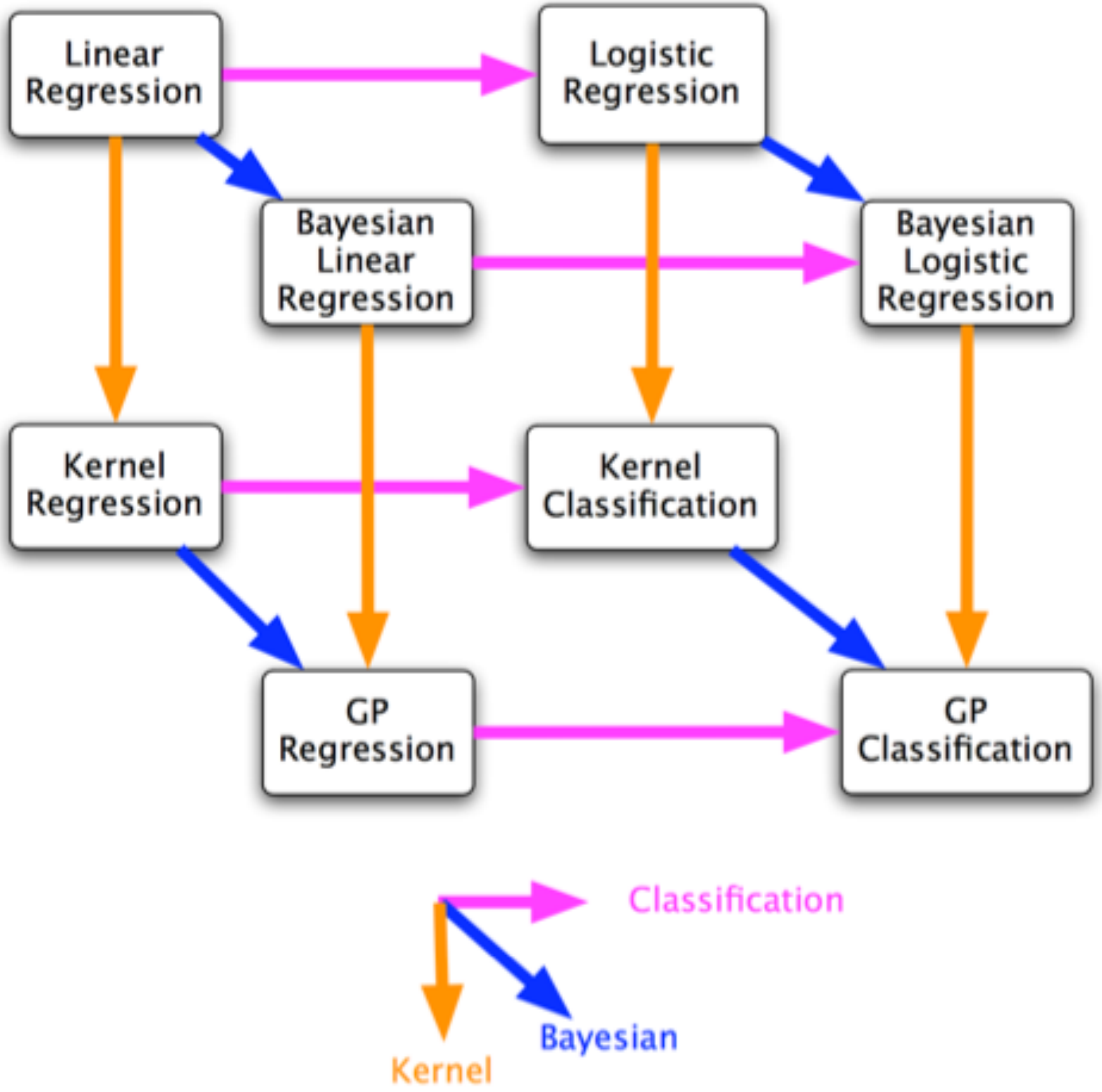
- So we want to model $p(x|\theta)$ where θ are some parameters of the model. In normal regression we took the path of linear regression $f = ax+b$, generalized linear regression of arbitrary basis functions, Bayesian generalized linear regression (prior), kernels and GP.
- Since we also want to enforce probabilistic interpretation we map f into an interval between 0 and 1. There are many sigmoid functions to do this:
- Logit or logistic function: $log[p/(1-p)] = f$
- Probit: error function or cumulative normal $p = \Phi(f)$ (Gaussian integrates to 1)

Linear Logistic Regression

- For linear logistic regression with logit we have $log[p/(1-p)] = f=wx + w_0$ giving $p(Y=1|x) = 1/(1+e^{-(wx+w_0)})$
- Notice similarity to two state statistical mechanics, with occupation numbers given by Boltzmann factors and $wx + w_0$ playing the role of $-E/kT$ (Boltzmann machine)
- To classify we say Y = 1 if p > 0.5 and Y = 0 if p < 0.5: $Y = 1$ if $wx + w_0 > 0$ and Y = 0 if $wx + w_0 < 0$
- This is called a linear classifier or *perceptron*: the decision boundary is a *N*-1 dimensional plane if *x* is *N*-dimensional
- Class probability depends on the distance from this plane: if far the probabilities are close to 0 or 1 (well classified), if close they are close to 0.5

Linear Logistic Regression

- Now that we specified the model we perform regression on parameters w based on a set of training data (x_i, y_i). Probability is p if $y_i = 1$ or (1-p) if $y_i = 0$. So the likelihood is
  $L(x,y|w,w_0) = \Pi_{i=1}^{N} p(x_i)^{y_i}[1-p(x_i)]^{1-y_i}$
- Taking the log and using $p = 1/(1+e^{-(wx+w_0)})$ this is converted to
  $logL = \Sigma p(x_i,y_i|w,w_0) = \Sigma_i \log(1-p_i) + \Sigma_i y_i \log[p_i/(1-p_i)]$
  $= \Sigma \log(1+e^{wx_i+w_0}) + \Sigma y_i(wx_i+w_0)$
- This iss called *cross-entropy*
- To find MLE we maximize log$L$ wrt w, w_0: unlike linear regression this does not have analytic linear algebra solution, so we have to use NL optimization
- It is however *convex optimization* (local=global minimum)
- Generalizations are now the same as for regression case: we can add Bayesian regularization prior, kernelize, GP…

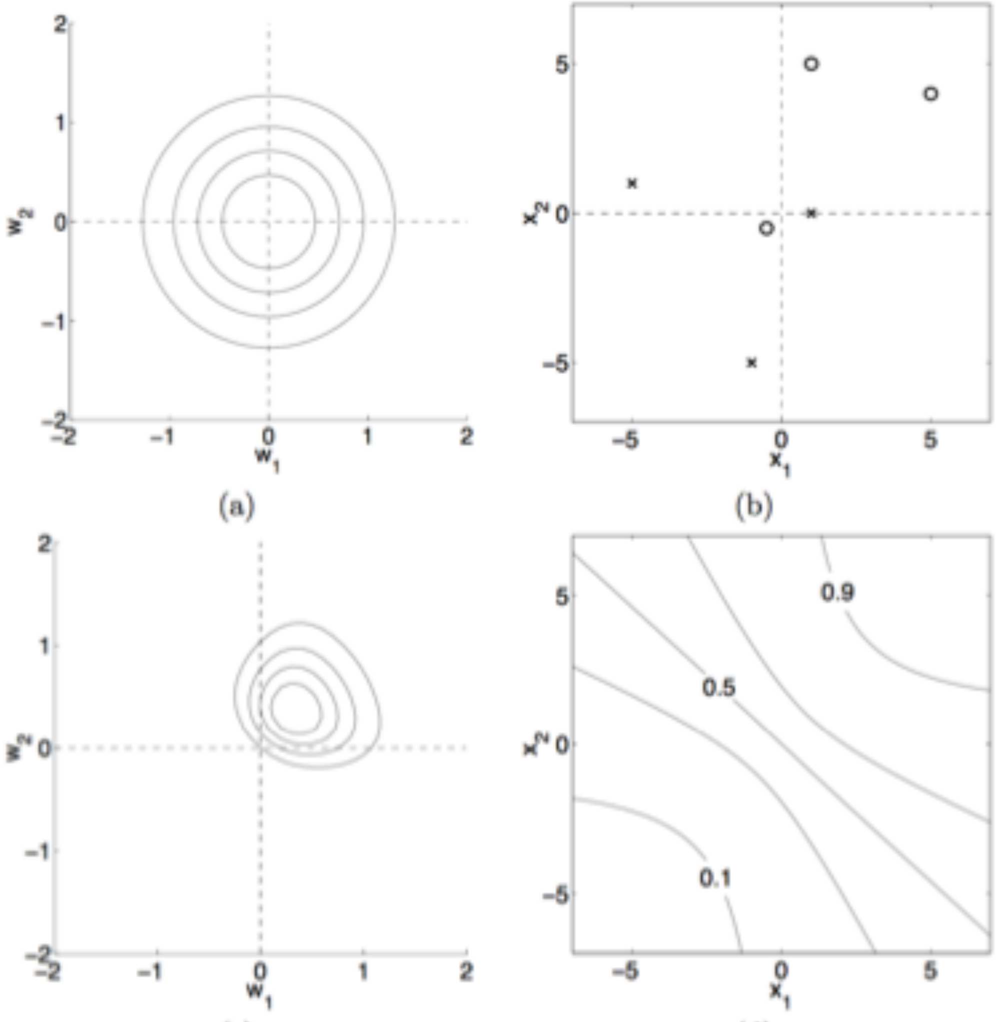

Linear Regression
Logistic Regression
Bayesian Linear Regression
Bayesian Logistic Regression
Kernel Regression
Kernel Classification
GP Regression
GP Classification
Classification
Bayesian
Kernel

Bayesian Logistic Regression

- As before: we add a prior on coefficients $w = (w_0, w_1, \ldots w_M)$ where M is dimension of x
- Example: L_2 regularization
- log posterior = log likelihood + log prior
  $\log p(w|x,y) = \Sigma_i p(x_i, y_i|w) - w\Sigma^{-1}w/2 + c$
- Find MAP for w, find posterior (Laplace…)
- Predict $p(y_*=1)$ at a new point x_* by marginalizing over w

$$p(y_* = +1|\mathbf{x}_*, \mathcal{D}) = \int p(y_* = +1|\mathbf{w}, \mathbf{x}_*) p(\mathbf{w}|\mathcal{D})\, d\mathbf{w}$$

Example (Rassmussen & Williams)



(a) (b)

(c) (d)

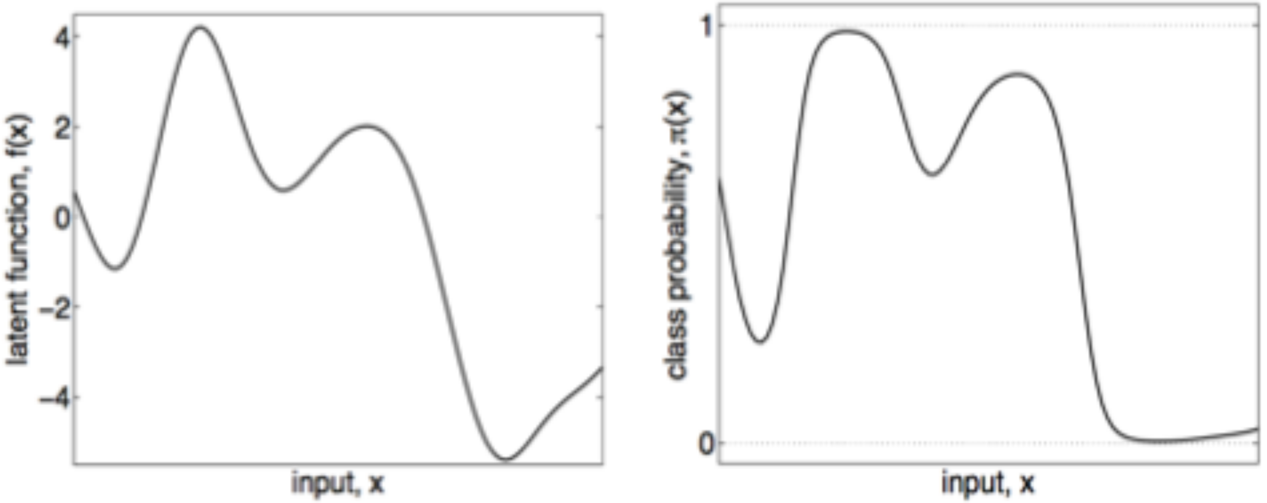
Figure 3.1: Linear logistic regression: Panel (a) shows contours of the prior distribution $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, I)$. Panel (b) shows the dataset, with circles indicating class +1 and crosses denoting class −1. Panel (c) shows contours of the posterior distribution $p(\mathbf{w}|\mathcal{D})$. Panel (d) shows contours of the predictive distribution $p(y_* = +1|\mathbf{x}_*)$.

Gaussian Process for Classification

- Natural extension of linear Bayesian logistic regression

$$f_i = f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + w_0$$

- In GP, *f* is a latent (nuisance) function with Gaussian prior distribution and no specified basis
- We squash it to determine class probability *p(f)*
- We implicitly assume no noise in training data



Gaussian Process for Classification

- 1) Evaluate function *f* which models how the likelihood of one class over the other varies across *X*. If we assume 0 noise:

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_*^{\mathrm{T}} \\ K_* & K_{**} \end{bmatrix}\right)$$

$$f_*|\mathbf{f} \sim \mathcal{N}(K_* K^{-1}\mathbf{f},\ K_{**} - K_* K^{-1} K_*^{\mathrm{T}})$$

- 2) Squash *f* into into (0,1) interval using sigmoid *p(f)*

$$\pi(f) = \mathrm{prob}(y = 1|f)$$

$$\boxed{\text{data},\ x_*} \xrightarrow{\text{GP}} \boxed{\text{latent function},\ f_*|x_*} \xrightarrow{\text{sigmoid}} \boxed{\text{class probability},\ \pi(f_*)}.$$

Gaussian Process for Classification

- 1) is almost the same as the regression case

$$p(f_*|\mathbf{f}) = \mathcal{N}(K_* K^{-1}\hat{\mathbf{f}},\ K_{**} - K_*(K')^{-1}K_*^{\mathrm{T}})$$

- 2) $\overline{\pi}_* = \int \pi(f_*) p(f_*|\mathbf{f}) df_*$

  This integral is analytic if we use error function:

$$\overline{\pi}_* = \Phi\left(\frac{\overline{f}_*}{\sqrt{1 + \mathrm{var}(f_*)}}\right)$$

- In 1) we have to do $p(f_*|X, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|X, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|X, \mathbf{y})\, d\mathbf{f}$,

- We will use MAP + Laplace: this will specify *K’* and *f*~

Training the GP in Classifier

- Classification: we have no noise, gaussian prior on $f = N(0,K)$, i.e. $\log p(f|x) = -f^T K^{-1} f/2$

$$p(\mathbf{f}|\mathbf{x},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})} \qquad p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i) \qquad \pi(f) = \mathrm{prob}(y=1|f)$$

$$\begin{aligned}\Psi(\mathbf{f}) &\triangleq \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|X) \\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi.\end{aligned}$$

$$\begin{aligned}\nabla\Psi(\mathbf{f}) &= \nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1}\mathbf{f}, \\ \nabla\nabla\Psi(\mathbf{f}) &= \nabla\nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1} \;=\; -W - K^{-1}\end{aligned}$$

- *$\log p(f|x) = -f^T K^{-1} f/2$*

$$p(\mathbf{f}|\mathbf{x},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})} \qquad p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i) \qquad \pi(f) = \text{prob}(y=1|f)$$

$$\begin{aligned}\Psi(\mathbf{f}) &\triangleq \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|X)\\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi.\end{aligned}$$

$$\begin{aligned}\nabla\Psi(\mathbf{f}) &= \nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1}\mathbf{f},\\ \nabla\nabla\Psi(\mathbf{f}) &= \nabla\nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1} = -W - K^{-1}\end{aligned}$$

- We seek to optimize $log\ p(f_i|x_i,y_i) = \Psi$ wrt f_i:

MLE solution is $\hat{\mathbf{f}} = K\nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$

- Solved using NL optimization (Newton's method…)
- f can fluctuate and we can use Laplace approximation to estimate its variance

$$(K^{-1} + W)^{-1}, \text{ with } W = -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$$

Training the GP in Classifier

$$p(\mathbf{f}|\mathbf{x},\mathbf{y}) \sim q(\mathbf{f}|\mathbf{x},\mathbf{y}) = \mathcal{N}(\hat{\mathbf{f}}, (K^{-1}+W)^{-1})$$

- So we need to add W to the posterior variance, giving $K' = K + W^{-1}$

$$p(f_*|\mathbf{f}) = \mathcal{N}(K_* K^{-1}\hat{\mathbf{f}},\ K_{**} - K_*(K')^{-1}K_*^{\mathrm{T}})$$

- Finally, we train for θ: $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x})d\mathbf{f}$.
- Using Laplace

$$p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = -\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}K^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log(|K|\cdot|K^{-1}+W|)$$

- We optimize this wrt θ

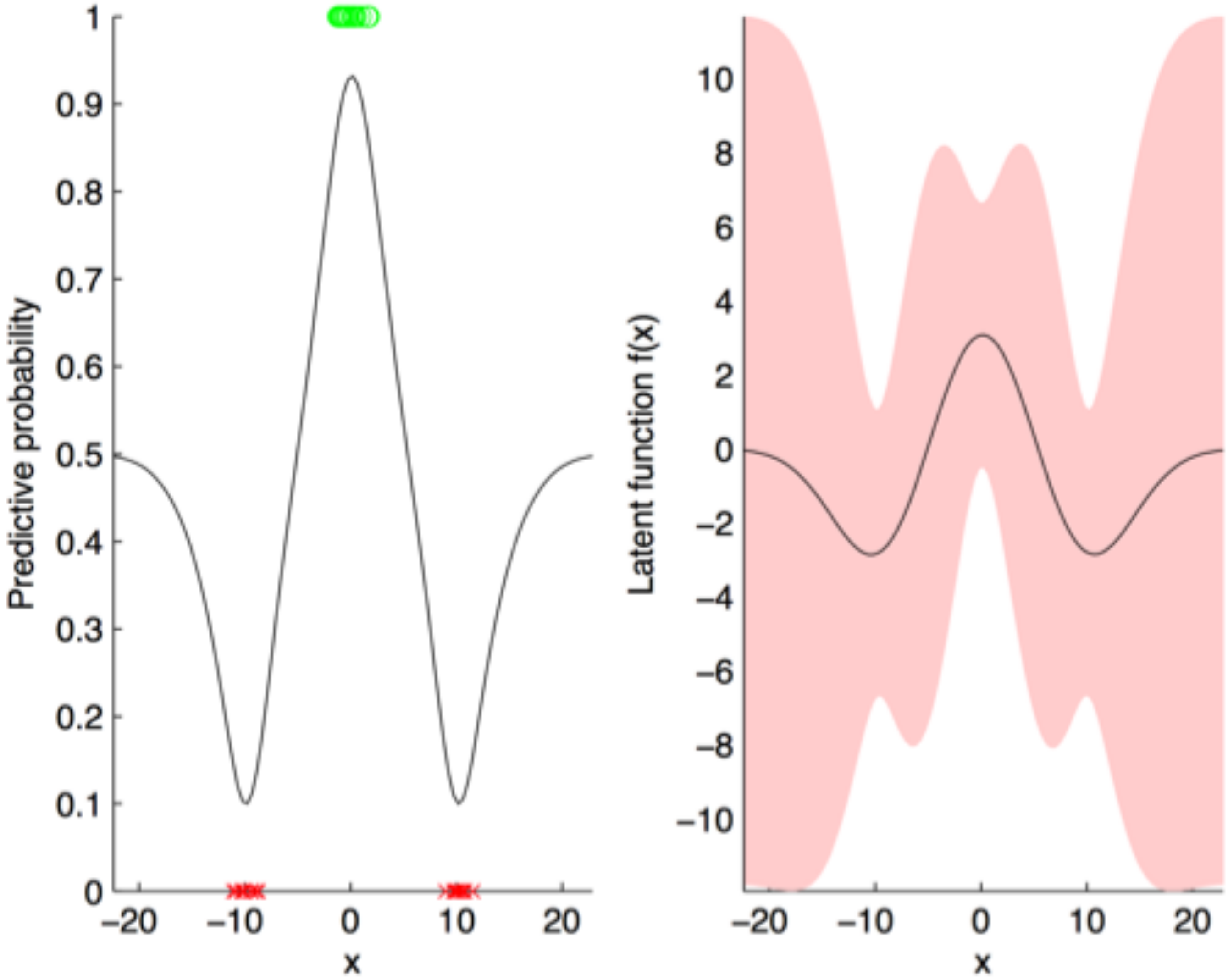
$$\boldsymbol{\theta} = (v_0, v_1, r_1, \ldots, r_d, \alpha)$$

Example

Input data: crosses and circles

Solid line is the answer $\pi(x)$

Left panel: latent function f



Extending regression to more than 2 classes

- We can give logit a probabilistic interpretation: probability of correct label assigned to y_i given x_i and $w(c)$ for each class c
- We know how to handle the normalization via partition function (as in statistical mechanics)
- Here we will assign logistic regression parameters to each class: $w(c)$ and use softmax function:
- $P(Y=c|X) = e^{w(c)x}/[\Sigma_c e^{w(c)x}]$
- Note that we can always set the parameters for one class to 0, as we have done for 2 classes
- Short-hand notation: we have set $x_0 = 1$ and defined new vector $w = (w_0, w_1, \ldots)$, this absorbs the bias into x_0

Multiclass Softmax Classifier

$$p(y = \mathcal{C}_c | \mathbf{x}, W) = \frac{\exp(\mathbf{x}^\top \mathbf{w}_c)}{\sum_{c'} \exp(\mathbf{x}^\top \mathbf{w}_{c'})}, \qquad L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

- Multiclass classification (MLE), Bayesian with a prior (MAP)
- Can be very large numbers: use $\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = \frac{Ce^{f_{y_i}}}{C\sum_j e^{f_j}} = \frac{e^{f_{y_i}+\log C}}{\sum_j e^{f_j+\log C}}$
  $\log C = -\max_j f_j$
- Information theoretical interpretation of why use softmax: it is cross-entropy

$$H(p,q) = -\sum_x p(x)\log q(x)$$

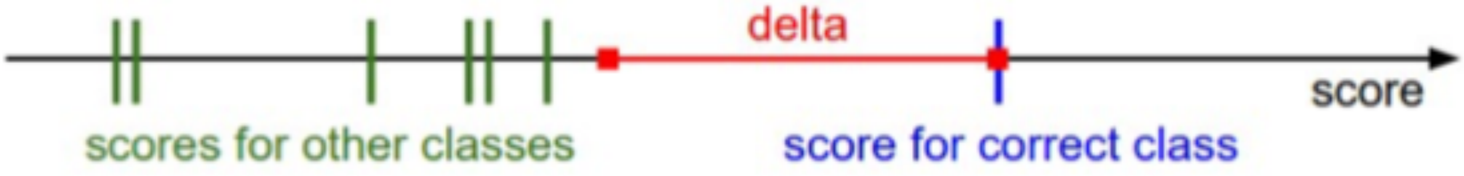
$$p = [0, \ldots 1, \ldots, 0]$$

$$H(p,q) = H(p) + D_{KL}(p||q)$$

- True distribution *p* has entropy *H(p) = 0*
- Softmax q: $q = e^{f_{y_i}} / \sum_j e^{f_j}$
- Minimizing softmax function corresponds to minimizing KL divergence between true *p* and *q*

Multiclass Support Vector Machines (SVM)

- A class *j* has a score $s_j = f(x_i, W)_j$ where $f(x_i, W) = Wx_i$
- This is also a linear function of *x*
- Hinge loss: SVM method wants to have score for correct class to be higher by some margin Δ:



$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

- Example: s = [13,-7,11], true class is 0, Δ = 10. Then
- L_i = max(0,-7-13+10) + max(0,11-13+10) = 8
- We induced penalty (loss) of 8 because 3rd class was only 2 less than the true class, and we required 10 for 0 loss

Regularization: adding (Bayesian) priors on weights

- Regularization

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

- SVM

$$L = \frac{1}{N}\sum_i \sum_{j\neq y_i} [\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$$

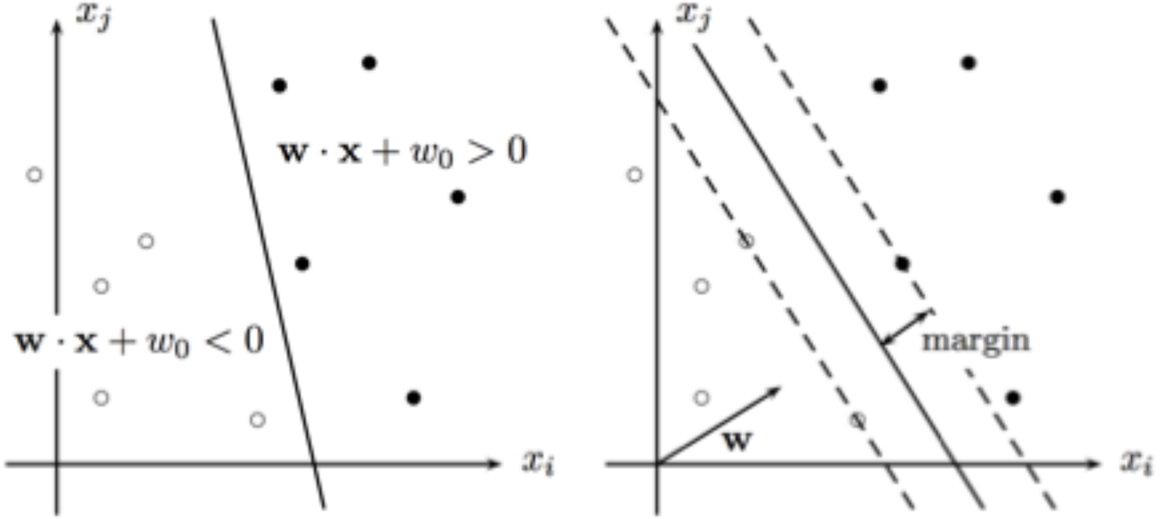
- Δ and λ are degenerate, so we only need to choose one (usually λ, setting Δ to 1)
- 2 class equivalent expression

$$\frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n}(1 - y_i f_i)_+$$

- $(x)_+$ stands for $\max(0, x)$

SVM: Hyperplanes

- The loss optimization can be derived from constrained optimization problem (quadratic programming)
- This can be converted into unconstrained problem using Lagrange multipliers $L = \frac{1}{N} \sum_i \sum_{j \neq y_i} [\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$
- From the derivatives w.r.t. $w = 0$ we get: $\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i$
- Final solution is linear in x: hyperplane (support vectors)


x_j
$\mathbf{w} \cdot \mathbf{x} + w_0 > 0$
$\mathbf{w} \cdot \mathbf{x} + w_0 < 0$
x_i
x_j
margin
$\mathbf{w}$
x_i

SVM Prediction and Kernel Trick

- Prediction at a new input x_*

$$\mathrm{sgn}(\mathbf{w}\cdot\mathbf{x}_* + w_0) \;=\; \mathrm{sgn}\left(\sum_{i=1}^{n}\lambda_i y_i(\mathbf{x}_i\cdot\mathbf{x}_*) + w_0\right)$$

- We see that the prediction is in the form of a dot product $x_i \cdot x_*$ so by basis kernel duality we can lift this into a kernel

$$\mathbf{w} \;=\; \textstyle\sum_i \alpha_i \mathbf{x}_i \qquad \alpha_i = \lambda_i y_i$$

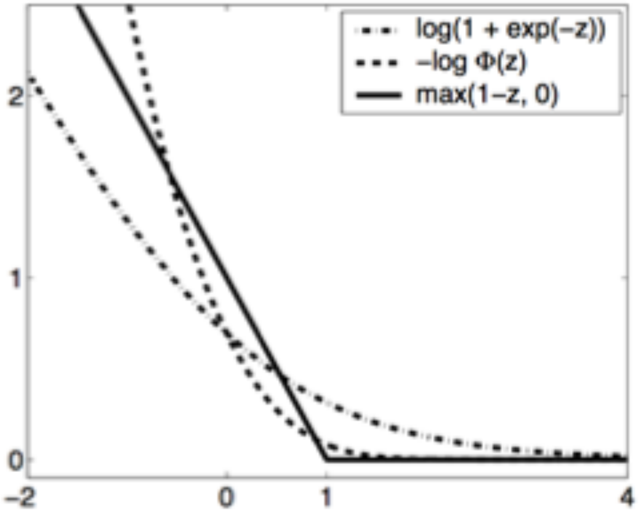
$$|\mathbf{w}|^2 = \textstyle\sum_{i,j}\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j). \qquad |\mathbf{w}|^2 \;=\; \boldsymbol{\alpha}^\top K\boldsymbol{\alpha} \;=\; \mathbf{f}^\top K^{-1}\mathbf{f},$$

$$K\boldsymbol{\alpha} \;=\; \mathbf{f} \qquad \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} + C\sum_{i=1}^{n}(1 - y_i f_i)_+$$

- So we can view SVM as a Gaussian process (where kernel corresponds to correlation *f*.)

SVM vs. GP

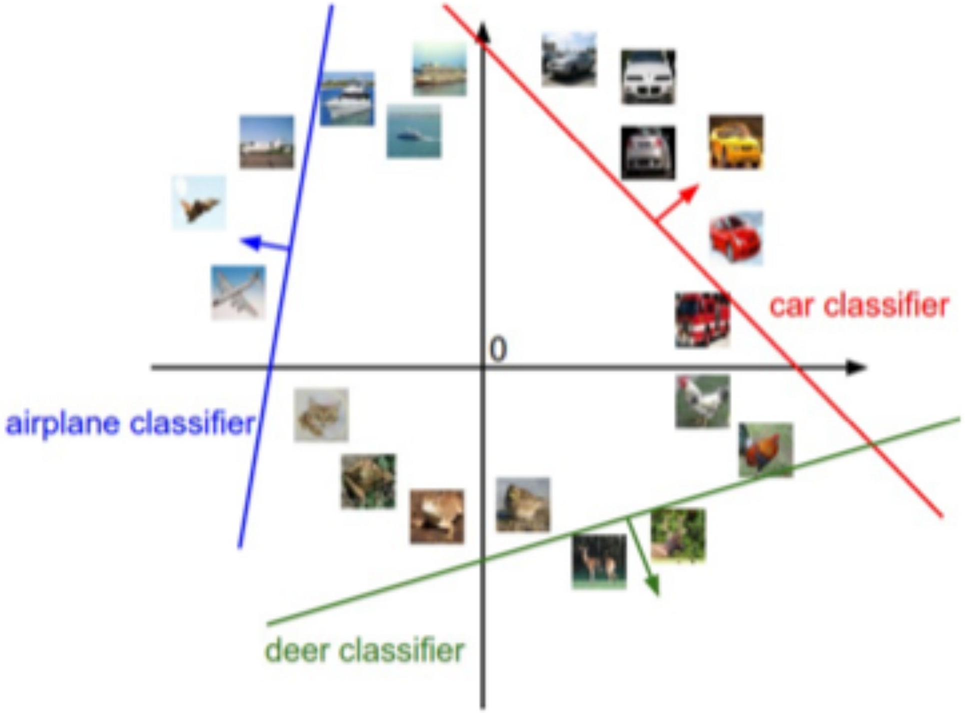
- SVM: $\frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} + C\sum_{i=1}^{n}(1 - y_i f_i)_+$
- GP: $\frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \sum_{i=1}^{n}\log p(y_i|f_i)$
- Qualitatively similar



- However, with GP one does not have regularization *C* as a free parameter, one learns kernel from the data, one has an estimate of uncertainty… So GP is better in principle, but they are often similar in practice

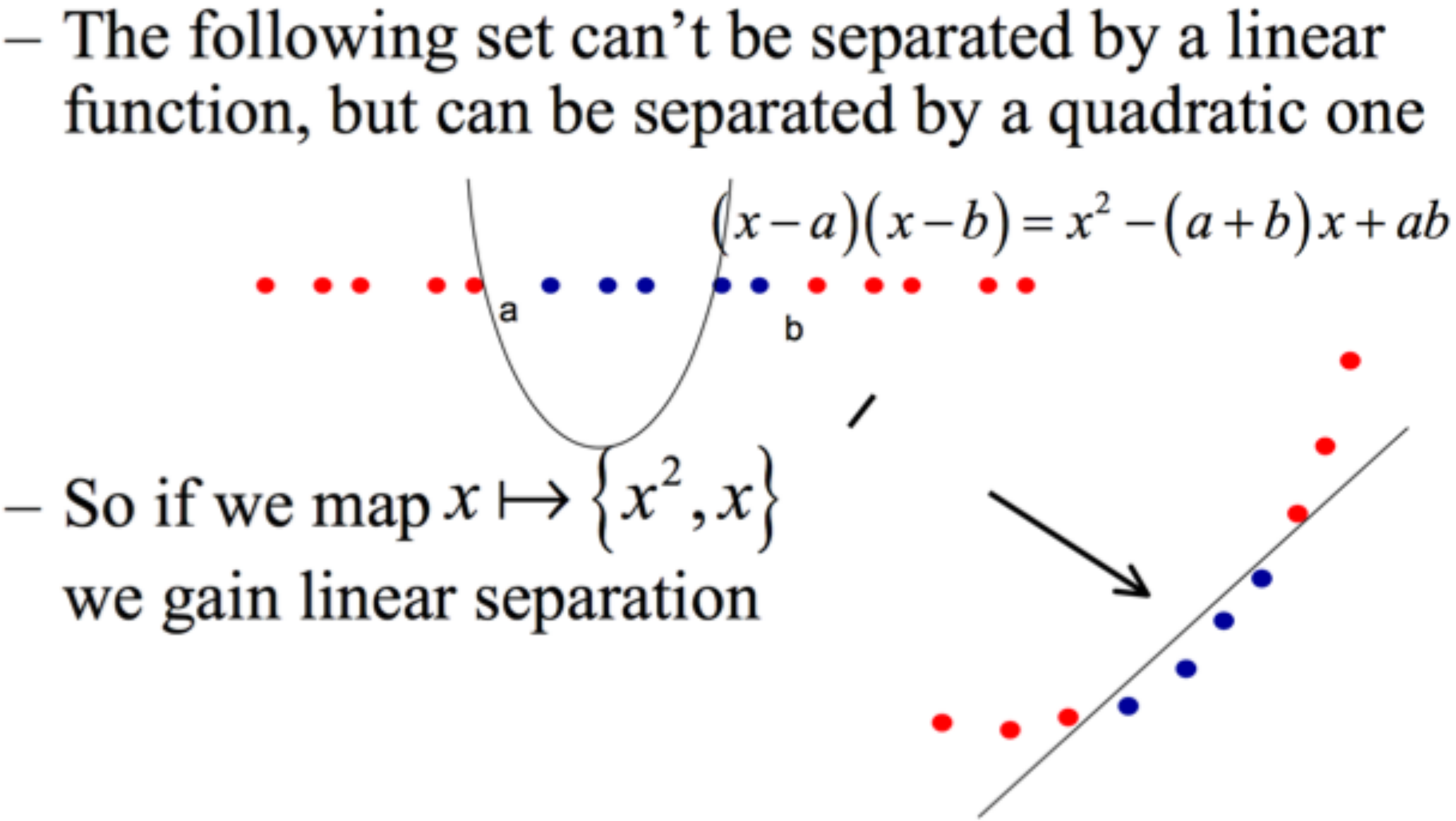
Visualizing SVM

- We have data in a very high dimension of image (e.g. CIFAR 10 32x32x3=3072 dim)
- We define the score as a weighted sum over these 3072 values: a linear function in 3072-dim space
- We find support vectors: in 2d this may look like this



Nonlinear SVM

- So far we only used basis element linear in x
- We can map the data to higher dim



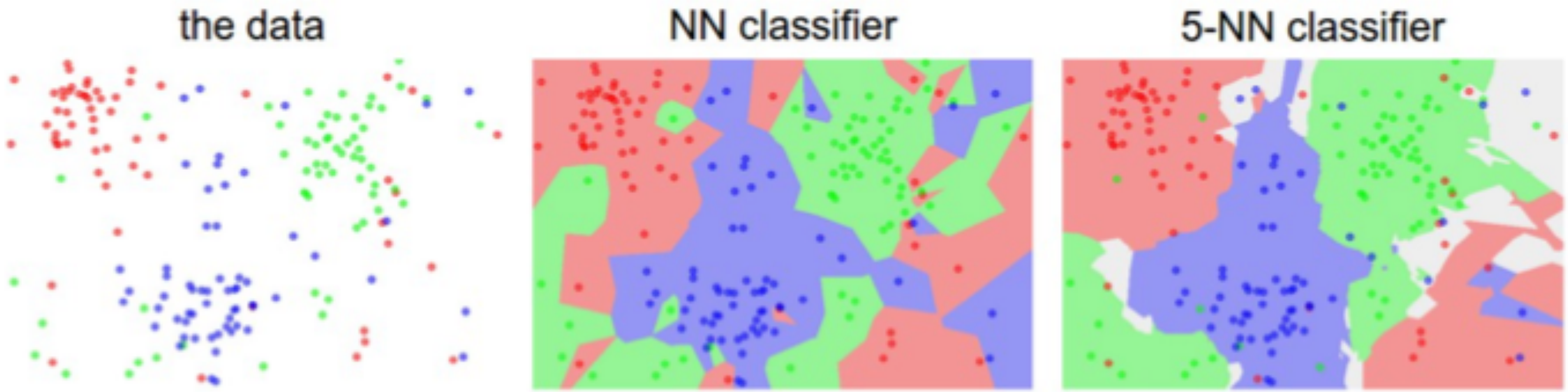
– The following set can’t be separated by a linear function, but can be separated by a quadratic one

$$(x-a)(x-b)=x^2-(a+b)x+ab$$

– So if we map $x \mapsto \{x^2, x\}$ we gain linear separation

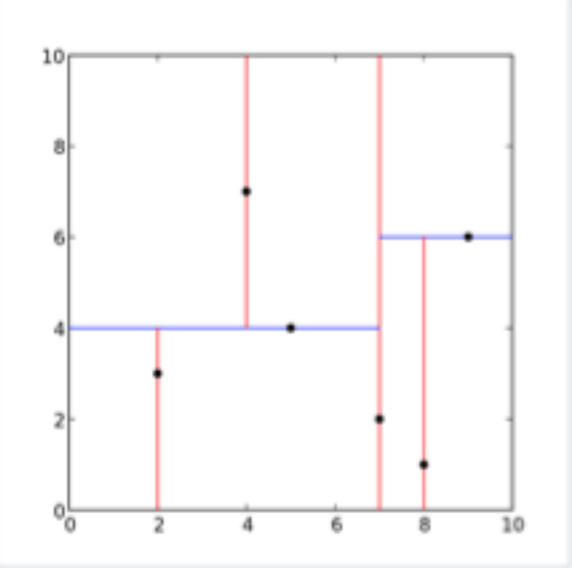
K-nearest Neighbor (K-NN) Classifier

- An even simpler classification: find nearest *k* neighbors and find highest class fraction among them. If *k* = 1 no degeneracies, but noisy

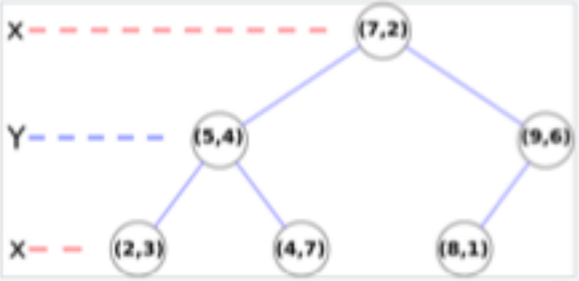


K-D Trees

- Used to partition *D* dim. distribution of *N* points into a tree. Splits by half and cycles dimensions. Cost *O(NlogN)*. Used to find NN in *logN*



k-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2) .



The resulting k-d tree.

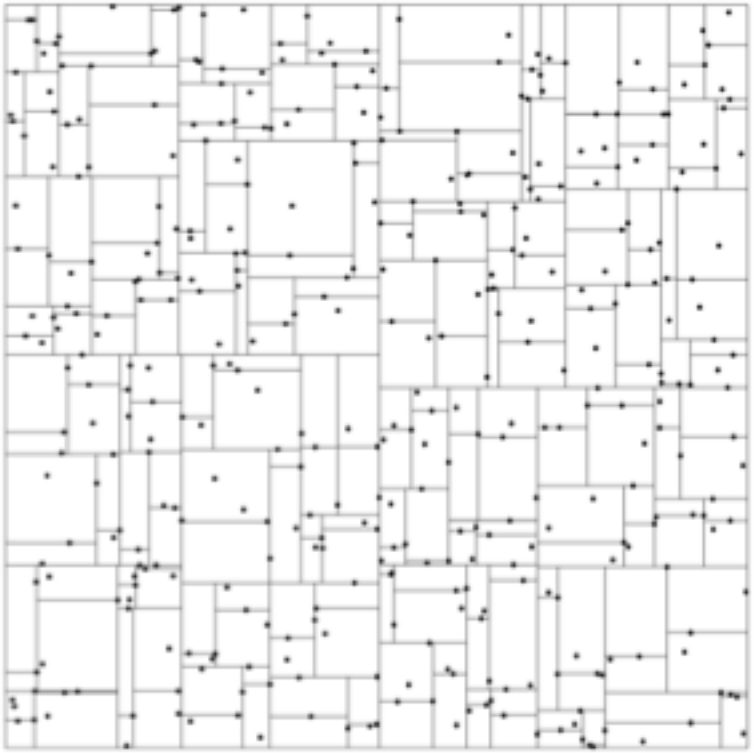
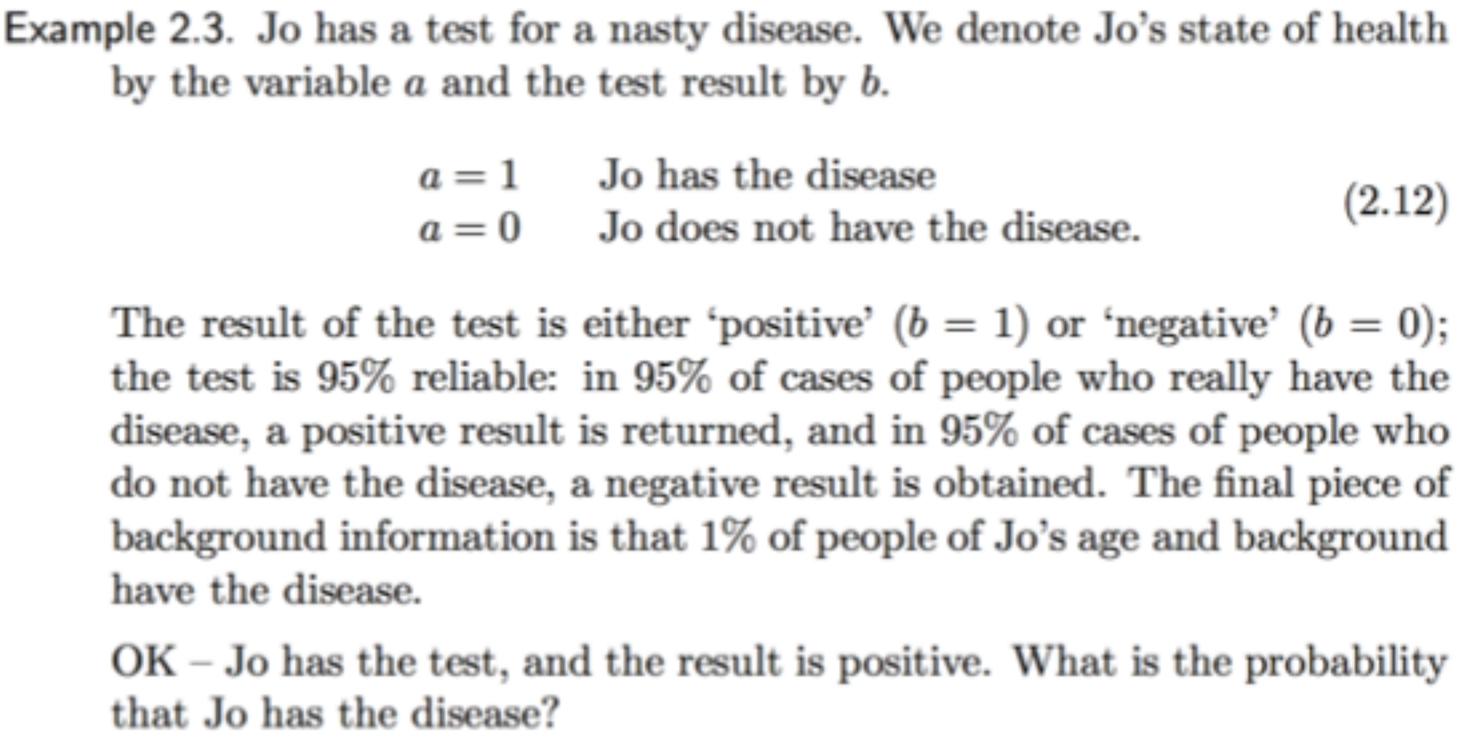


Figure 21.2.1. KD tree constructed from 1000 points in the plane. The first subdivision is visible as a full-height vertical line about halfway across the figure. The next subdivisions are horizontal lines, extending halfway across the figure. The subdivisions alternate between horizontal and vertical, and partition into (nearly) equal numbers of points at each stage. This tree terminates when there are either one or two points in a box (one of which is usually on the box boundary).

Naïve Bayes Classification

- Let’s go back to the beginning of the class when we discussed



Example 2.3. Jo has a test for a nasty disease. We denote Jo’s state of health by the variable a and the test result by b.

$$\begin{array}{ll} a=1 & \text{Jo has the disease} \\ a=0 & \text{Jo does not have the disease.} \end{array} \qquad (2.12)$$

The result of the test is either ‘positive’ ($b=1$) or ‘negative’ ($b=0$); the test is 95% reliable: in 95% of cases of people who really have the disease, a positive result is returned, and in 95% of cases of people who do not have the disease, a negative result is obtained. The final piece of background information is that 1% of people of Jo’s age and background have the disease.

OK – Jo has the test, and the result is positive. What is the probability that Jo has the disease?

Step 1: Write down all probabilities

We are given conditional probabilities

$$\begin{aligned} P(b=1\,|\,a=1) &= 0.95 & P(b=1\,|\,a=0) &= 0.05 \\ P(b=0\,|\,a=1) &= 0.05 & P(b=0\,|\,a=0) &= 0.95; \end{aligned}$$

And marginal probability of *a*

$$P(a=1) = 0.01 \qquad P(a=0) = 0.99.$$

We want $P(a = 1|b = 1)$

Step 2: Deduce joint probability *P*(*a*, *b*)

$$P(a,b) = P(a)P(b\,|\,a)$$

Step 3:

$$\begin{aligned} P(a=1\,|\,b=1) &= \frac{P(b=1\,|\,a=1)P(a=1)}{P(b=1\,|\,a=1)P(a=1) + P(b=1\,|\,a=0)P(a=0)} \\ &= \frac{0.95 \times 0.01}{0.95 \times 0.01 + 0.05 \times 0.99} \\ &= 0.16. \end{aligned}$$

Lots of false positives!

Naïve Bayes Example

- We use Bayes rule and assume conditional distributions of data are independent (this is the naïve part) to classify
- Example: classify customers into old and young based on whether they like/dislike 4 different radio stations. We know

  p(r1=like|young)=0.95, p(r2=like|young)=0.05,
  p(r3=like|young)=0.02, p(r4=like|young)=0.2.
  p(r1=like|old)=0.03, p(r2=like|old)=0.82,
  p(r3=like|old)=0.34, p(r4=like|old)=0.92.
- We know p(old)=0.9. We are given new customer data (like = r1, r3, dislike = r2, r4). Classify new customer:
- $p(\text{young}|r) = p(r|\text{young})p(\text{young})/S_{\text{young,old}}p(r|\text{age})p(\text{age})$ and
  $p(r|\text{age}) = \Pi_{i=1,2,3,4}\ p(r_i|\text{age})$
- $p(r|\text{young})\ p(\text{young}) = 0.95*0.95*0.02*0.8*0.1 = 0.0014$
- $p(r|\text{old})\ p(\text{old}) = 0.03*0.18*0.34*0.08*0.9 = 1.34\text{x}10^{-4}$
- $p(\text{young}|r) = 0.0014/(0.0014+1.34\text{x}10^{-4}) = 0.91 > 0.5$: classify young

Naïve Bayes Pros and Cons

- The probabilities $p(\text{r1}|\text{young})$ come from some training data: we can use ML to set $p(x_i|\text{class c})$
- Data can be an image: e.g. 32x32x3 pixels, and we use training data to obtain $p(x_i|\text{class c})$
- In this case independence assumption can be very naïve
- if an attribute x_i has no counts for class c in the training data, $p(x_i|\text{class c})=0$, then it will say that the data cannot be in that class irrespective of actual values of x
- Instead one can use Bayesian naïve Bayes to allow for uncertainty in $p(x_i|\text{class c})$, determining $p(x_i|\text{class c})$ as a distribution and averaging over it.
- For example, $p(\text{r3=like}|\text{young}) = \max(0, N(0.02,0.02))$ (properly renormalized…)

Decision Trees

- Suppose we have a set of attributes and associated decisions. We are given a set of new attributes and we need to predict the decision. We split the data into a tree based on attributes. We keep dividing the tree in terms of attributes until we have pure states

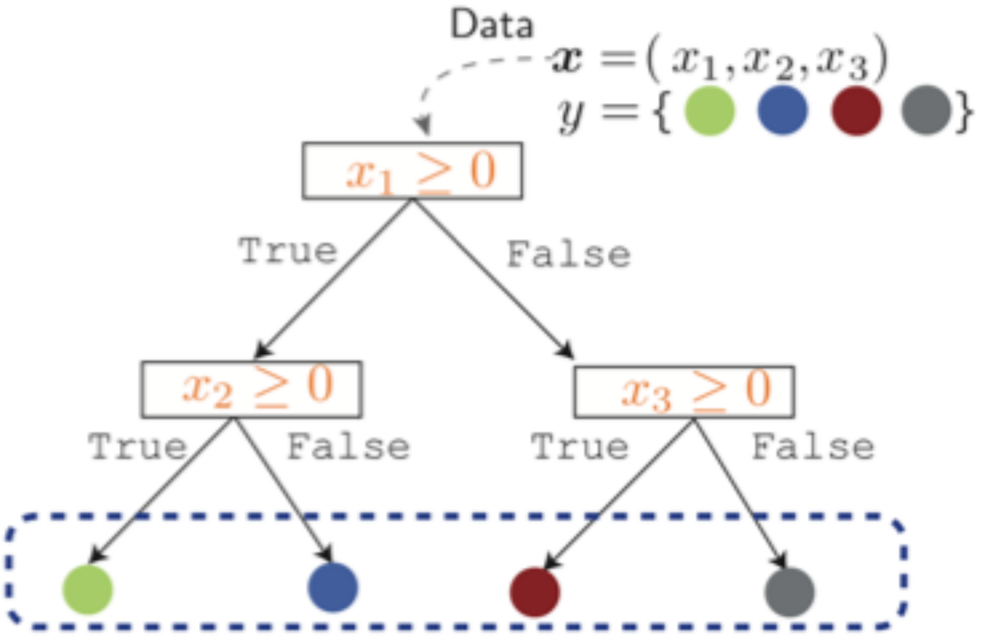
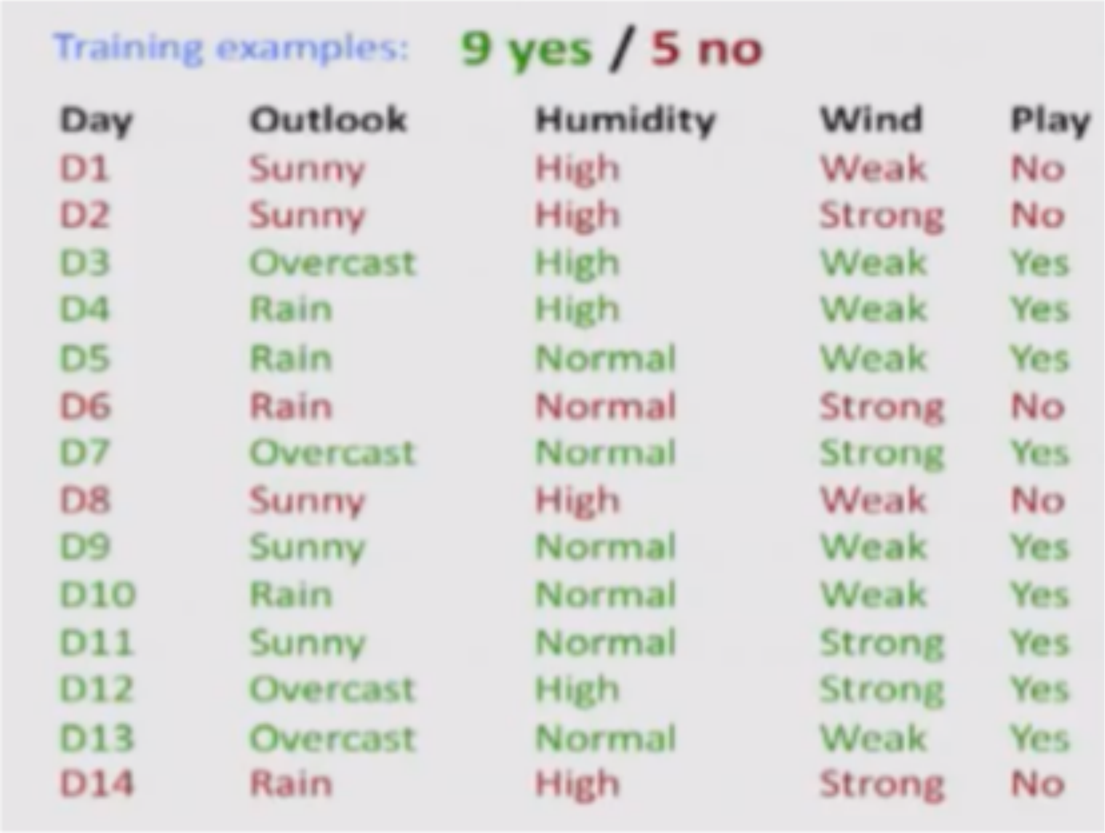


FIG. 30 Example of a decision tree. For an input observation $\boldsymbol{x}$, its label y is predicted by traversing it from the root all the way down the leaves, following branches it satisfies.

Decision Trees

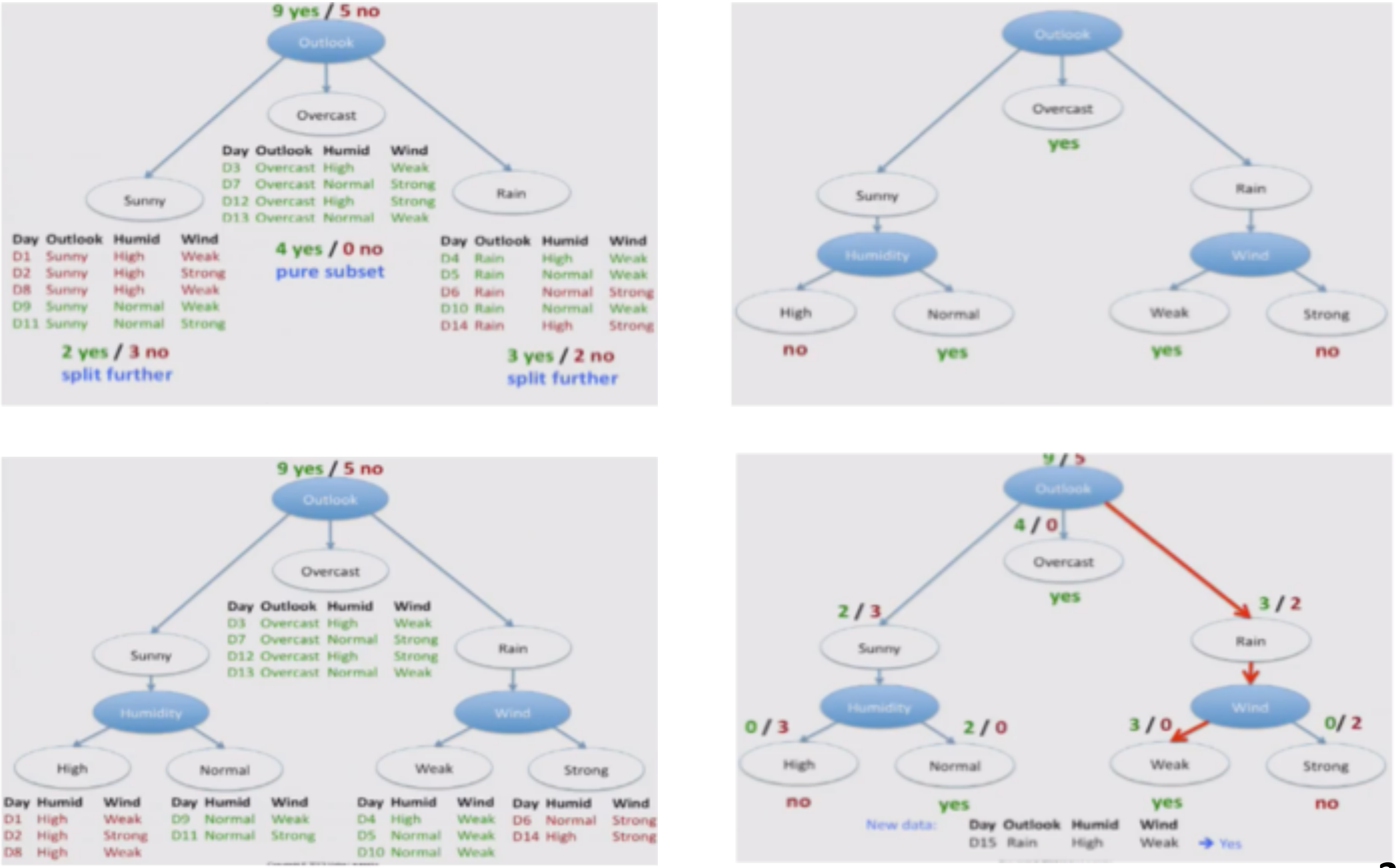
- Example:



Training examples: 9 yes / 5 no

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Decision Tree Example



Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

Day	Humid	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humid	Wind
D9	Normal	Weak
D11	Normal	Strong

Day	Humid	Wind
D4	High	Weak
D5	Normal	Weak
D10	Normal	Weak

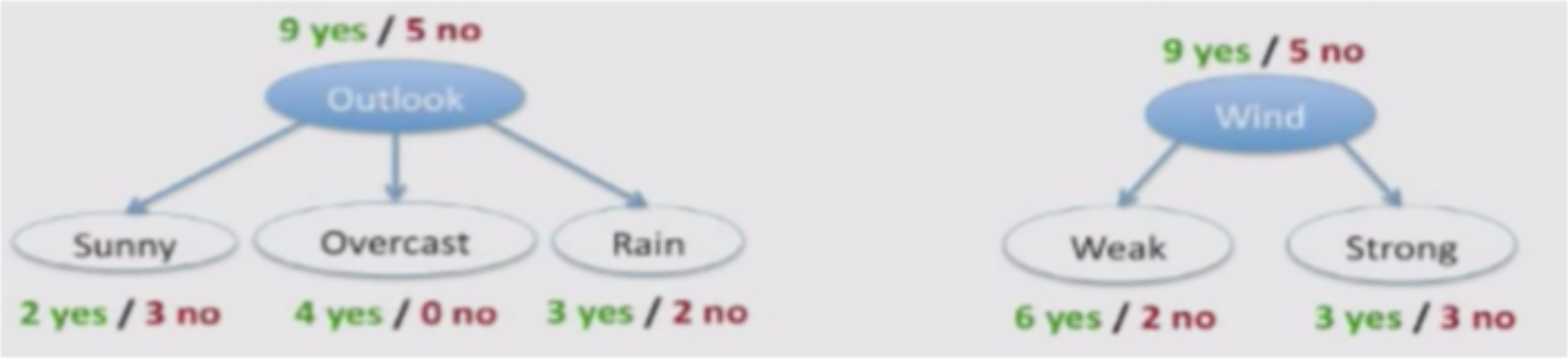
Day	Humid	Wind
D6	Normal	Strong
D14	High	Strong

New data:

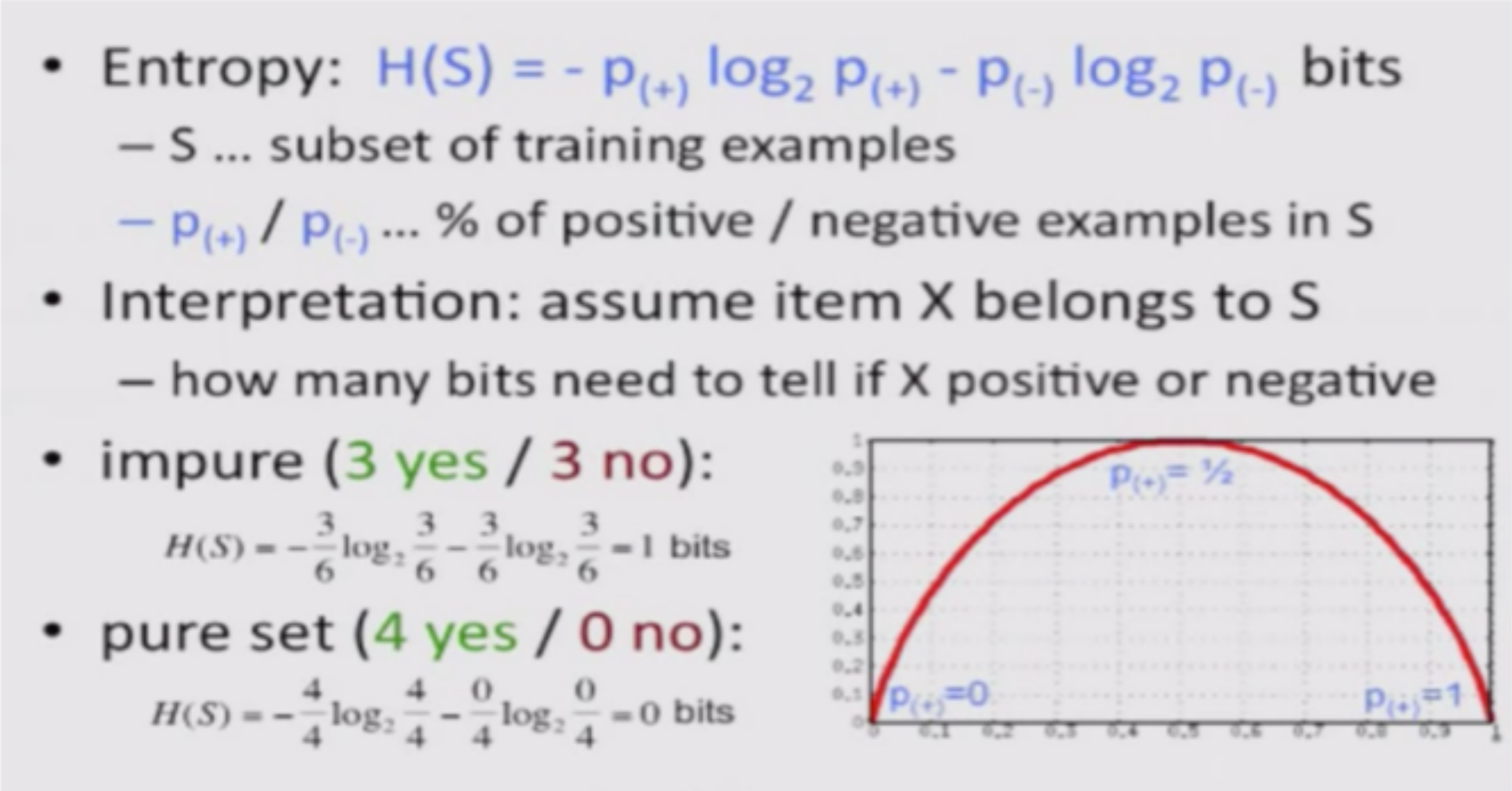
Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	→ Yes

How to choose how to split the attributes on?

- We want to measure purity of split: pure sets (4 yes/0 no) mean we are completely certain. Impure sets (3 yes/3no) very uncertain.
- After the split we want to be more certain: high purity is better
- We cannot use posterior *p*(yes|attribute) since we want to be symmetric w.r.t. yes/no



ID3/CART Algorithm: Entropy



- Entropy: $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$ bits
  - S … subset of training examples
  - $p_{(+)}$ / $p_{(-)}$ … % of positive / negative examples in S
- Interpretation: assume item X belongs to S
  - how many bits need to tell if X positive or negative
- impure (3 yes / 3 no):

$$H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1 \text{ bits}$$

- pure set (4 yes / 0 no):

$$H(S) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0 \text{ bits}$$

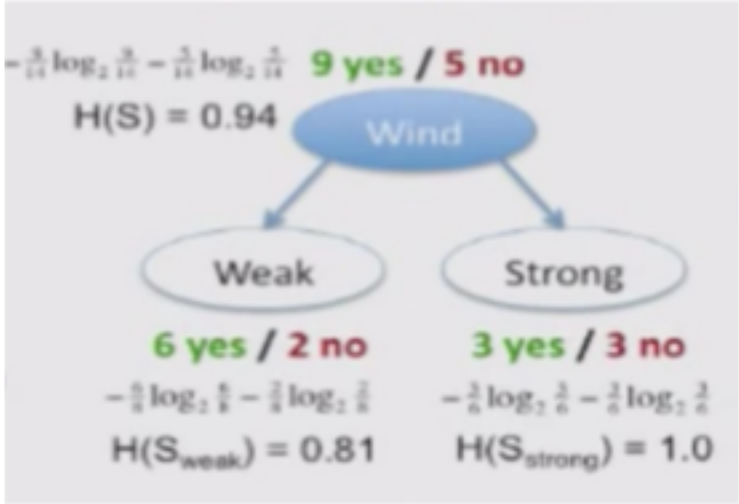
Information Gain from Mutual Information

- We want many items in pure sets.
- We want to look at the drop of entropy after split:

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

V ... possible values of A
S ... set of examples {X}
S_V ... subset where $X_A = V$

- This is same as mutual information. We have seen mutual information in the information lecture



$-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$
9 yes / 5 no
H(S) = 0.94
Wind
Weak
Strong
6 yes / 2 no
3 yes / 3 no
$-\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}$
$-\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$
$H(S_{weak}) = 0.81$
$H(S_{strong}) = 1.0$

$$I_E([6,2]) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8112781$$

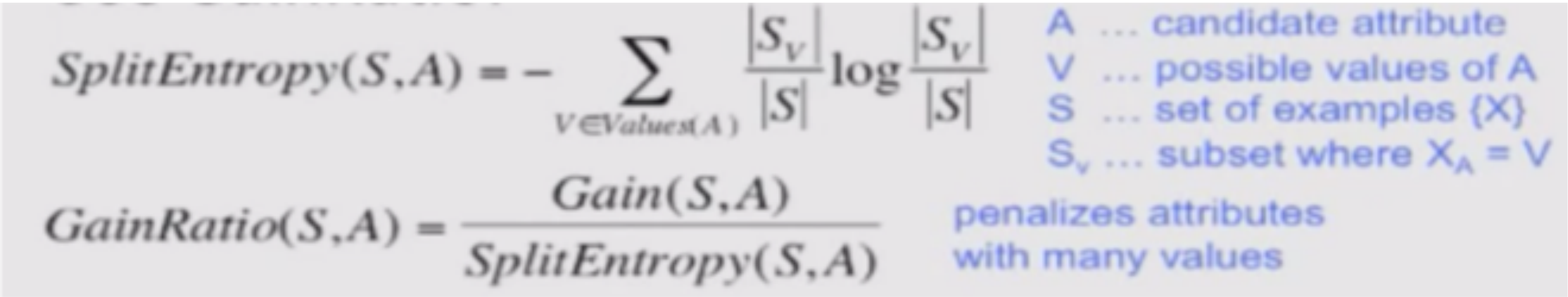
$$I_E([3,3],[6,2]) = I_E(\text{windy or not}) = \frac{6}{14}\cdot 1 + \frac{8}{14}\cdot 0.8112781 = 0.8921589$$

$$I_E([9,5]) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940286$$

$$IG(\text{windy}) = I_E([9,5]) - I_E([3,3],[6,2]) = 0.940286 - 0.8921589 = 0.0481271$$

How large trees? Are all attributes equally good?

- If you keep running the trees will grow down to 1 event per leaf: danger of overfitting
- Stop splitting when not statistically significant
- Prune branches that are not useful: pretend to remove the node+children, measure performance on validation data, remove branches that are not performing well
- Example of bad attribute: if you split by day all states are pure (1/0 or 0/1). However, this attribute will never repeat
- Penalize attributes with many values



$$SplitEntropy(S,A) = -\sum_{V\in Values(A)} \frac{|S_V|}{|S|}\log\frac{|S_V|}{|S|}$$

A ... candidate attribute

V ... possible values of A

S ... set of examples {X}

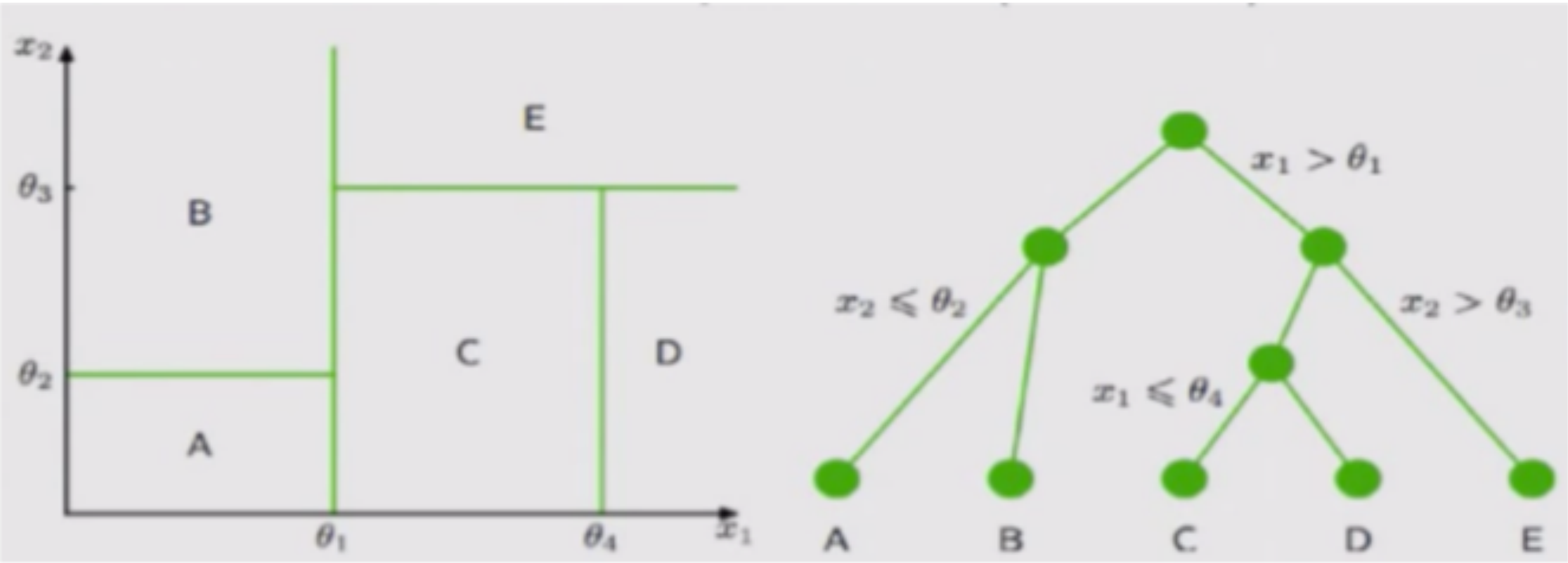
S_V ... subset where $X_A = V$

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

Continuous Attributes

- Use thresholds and partition space
- Threshold can be optimized



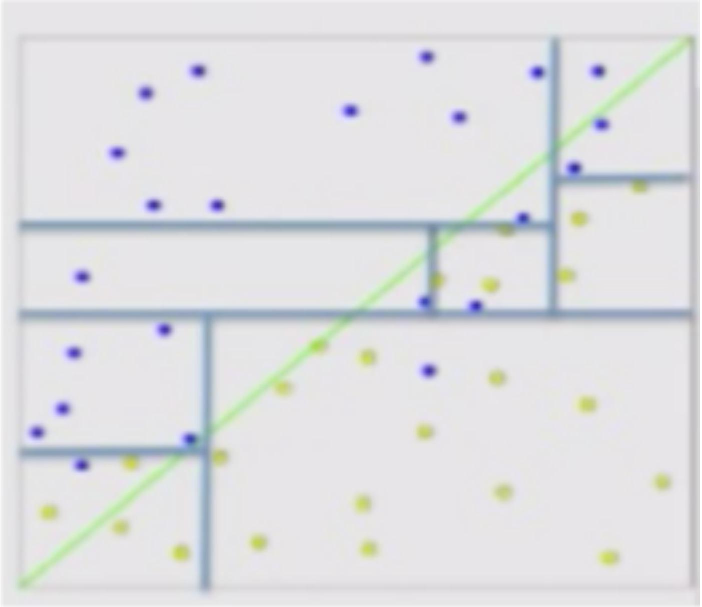
- This is similar to k-NN: the role of k-NN is replaced with the points in the same leaf

Decision Trees Generalization, Pros/Cons

- Multi-class entropy

$$H(S) = -\Sigma_c\, p_{(c)} \log_2 p_{(c)}$$

- Regression requires a different entropy definition
- Pros: interpretable, simple to handle irrelevant attributes (simply gain = 0), can be very compact, are very fast
- Cons: only axis aligned splits of data
- Greedy: looking one step ahead only



Random Forest

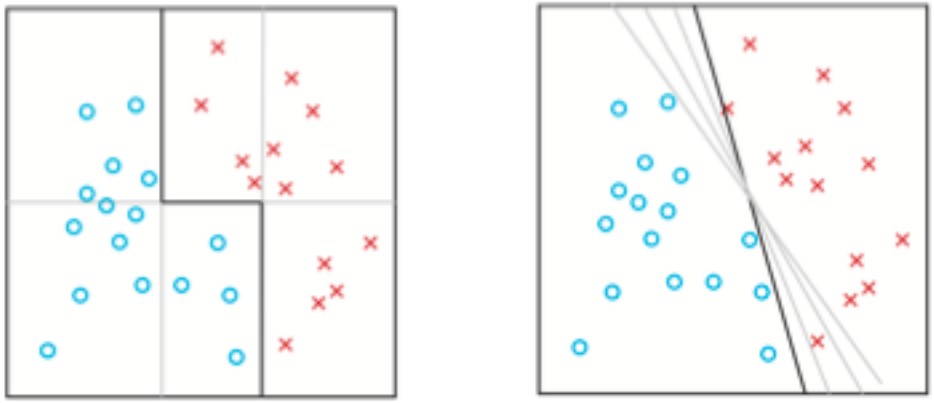
- Decision trees can have a high variance
- We grow a bunch of trees using random subsamples (bagging: subsampling the data, or bootstrap aggregation)
- Grow a tree only on a subsample of attributes d << D
- Given a new data point classify using each tree separately
- This is called feature bagging
- Use majority vote (class predicted most often) for classification (average for regression)
- Strange algorithm that uses ensembles of estimators, but works surprisingly well

Why ensemble methods work?

- If they are not fully correlated averaging over ensembles reduces the variance, while not increasing bias (or not increasing much for bootstrap aggregates)
- Statistical reason: in Bayesian statistics we always average over the models that fit the data
- Computational reason: different starting points lead to different solutions which avoids getting stuck in local minima far from the global one
- Representational reason: averaging low representational models gives us higher representation power

Louppe 2014,

Mehta etal 2018



Aggregating different linear hypotheses

Linear perceptron hypothesis

FIG. 27 Why combining models? On the left we show that by combining simple linear hypotheses (grey lines) one can achieve better and more flexible classifications (dark line), which is in stark contrast to the case in which one only uses a single perceptron hypothesis as shown on the right.

Boosting

- In bootstrapping we average over bootstrap realizations
- In Bayesian analysis we weight the models by their relative probability
- Boosting: weight different models by downweighting those that perform poorly
- Iterative procedure: AdaBoost (Freund etal 1999)
- Extension is Gradient Boosted Trees: use gradient descent down some loss function that achieves boosting
- Example: XGBoost, currently most powerful method for structured datasets, wins most Keggle competitions
- For unstructed data use Neural Networks (next lecture)

Summary

- Classification: in principle a simple extension of regression, in practice a huge field
- Here we first looked at logistic regression, its Bayesian, kernel and GP versions, SVP
- Then we looked at naïve Bayes, the simplest possible classifier we have already used in the beginning of the class
- Decision tree based classifiers: random forest, XGBoost… Uses concepts of entropy and information gain, as well as ensemble aggregation
- Our SVP discussion is a starting point of neural networks (Lecture 14)

Literature

- *Pattern Recognition and Machine Learning*, C. Bishop, Chapter 3
- *Bayesian Data Analysis,* Gelman et al. , Chapter 20-21
- Mehta etal https://arxiv.org/pdf/1803.08823.pdf
- http://www.gaussianprocess.org
- http://cs231n.github.io
- https://arxiv.org/pdf/1505.02965.pdf
- http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf