Introduction to Machine Learning

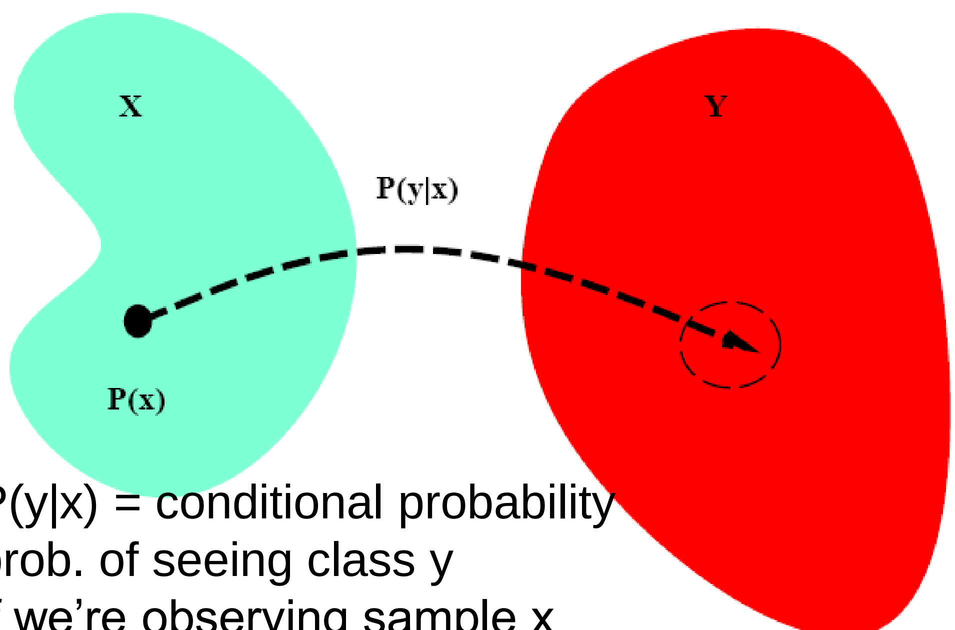
Lecture 5

Instructor:
Dr. Tom Arodz

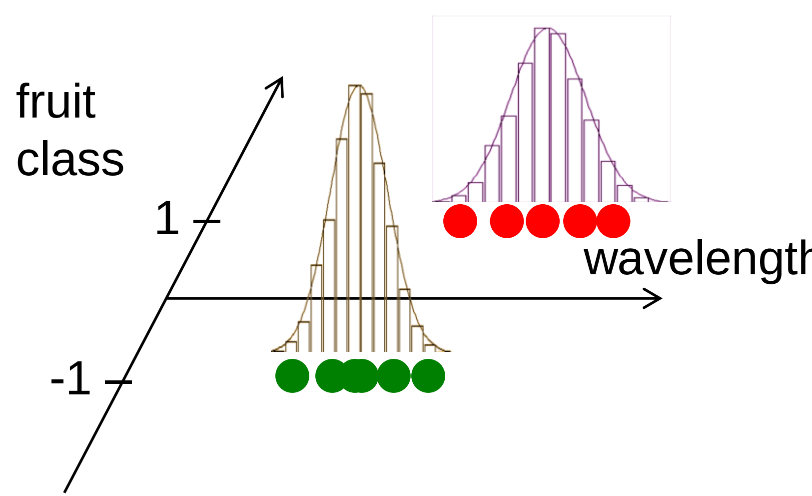
Recap: Probabilistic setting

- Training examples $\mathbf{z} = (\mathbf{x}, y)$ come from space $\mathcal{Z} = \mathcal{X} \times \{-1, +1\}$
- Over that space, we have a joint probability distribution
- Examples are randomly sampled (we call them "samples") from that distribution and have probability P(z) = P(x,y)
- We can factor it using conditional probability, in two ways
  - P(z)= P(x,y)=P(y|x)P(x)
  - =P(x|y)P(y)

Bayes's Theorem:
$P(y_i \mid x) = P(x \mid y_i)P(y_i) / P(x)$



P(y|x) = conditional probability
prob. of seeing class y
if we're observing sample x



Recap: Probabilistic classification

- It is often "easy" to know this factorization: $P(x,y)=P(x|y)P(y)$
  - The distributions $P(x|y_i)$ for each class y_i (i.e., the distributions over feature vectors x)
  - The probabilities $P(y_i)$ for each class y_i (i.e., single numbers)
- How do we make decisions given this information?
  - $p(y_i \mid x) = p(x \mid y_i)\, p(y_i) \,/\, p(x)$
  - $p(y_i \mid x) = p(x \mid y_i)\, p(y_i) \,/\, \Sigma_i\, p(x \mid y_i)\, P(y_i)$

    $P(A)=\Sigma_i P(A|B_i)P(B_i)$

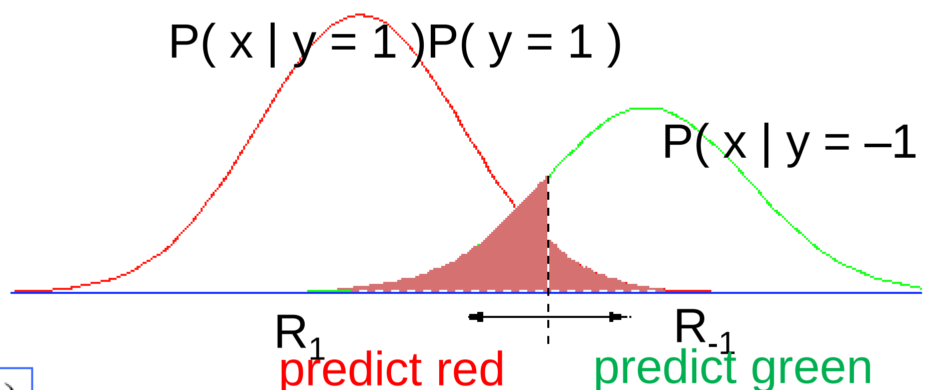
    - If we are interested only in predicting "which class" (i.e., which $p(y_i|x)$ is highest) and not in the probability of each class, then we can ignore $p(x)$
    - $p(y_i \mid x) \approx p(x \mid y_i)\, p(y_i)$
- **If the distributions** $P(x|y)$ **and** $P(y)$ **we use are "correct", this is the best way to make predictions**
  - **It's called "Optimal Bayes Classifier"**
  - **It has lowest possible "error rate" for that distrib.**

Optimal Bayes classifier

- Classifier with the lowest possible error for a given distribution over (X,y)
  - **1D example**: classifier is a single threshold
  - dividing feature space into
  - regions R_i in which we predict class y_i

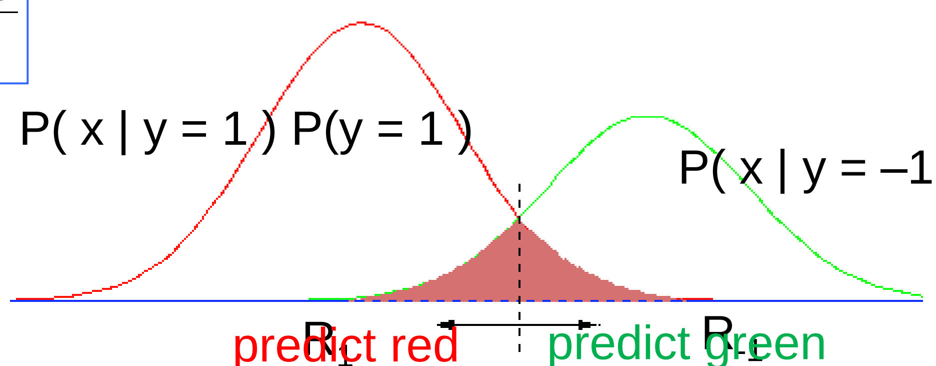
What do the pink areas represent?

Which decision threshold is better?



Bayes Theorem:

$$P(y_i \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y_i) P(y_i)}{p(\mathrm{x})}$$



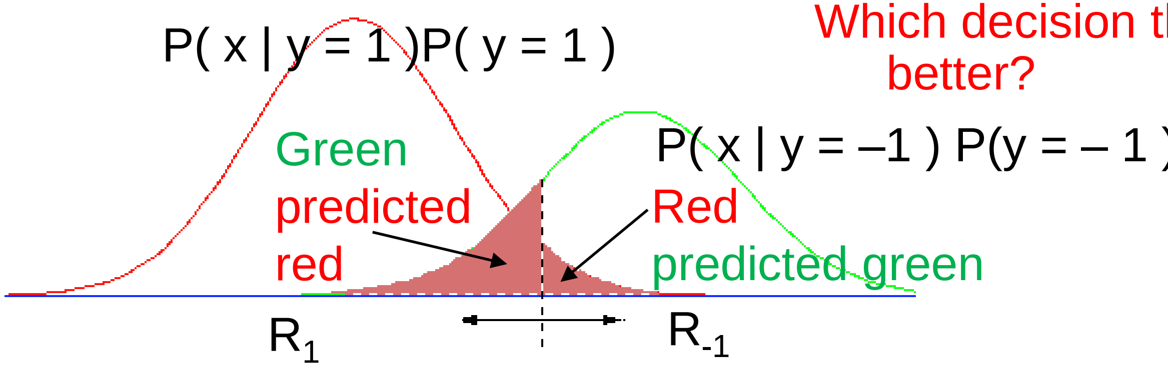
Optimal Bayes classifier

- Classifier with the lowest possible error for a given distribution over (X,y)
  - **1D example**: classifier is a single threshold
  - dividing feature space into
  - regions R_i in which we predict class y_i

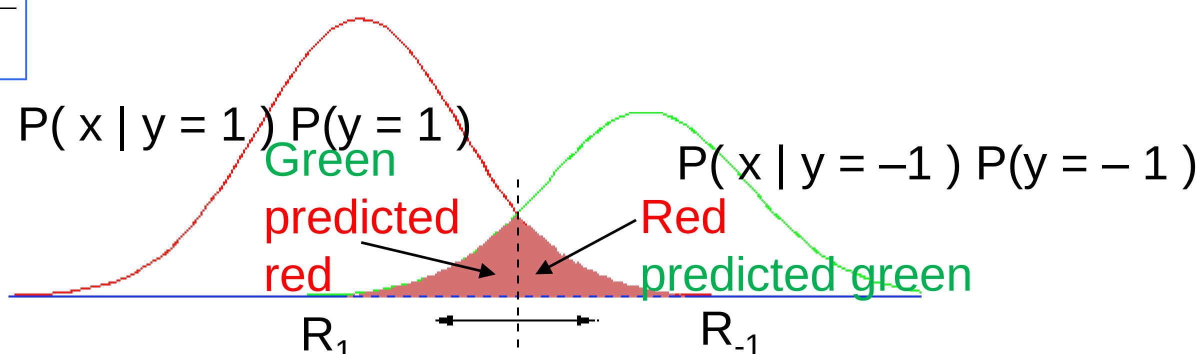
What do the pink areas represent?

Expected error
$= \mathbf{E}_\mathbf{x}(P(y_{wrong}|x))$
$= \int P(y_{wrong}|x)\, P(x)\, dx$
$= \int P(x|y_{wrong})\, P(y_{wrong})\, dx$

Which decision threshold is better?



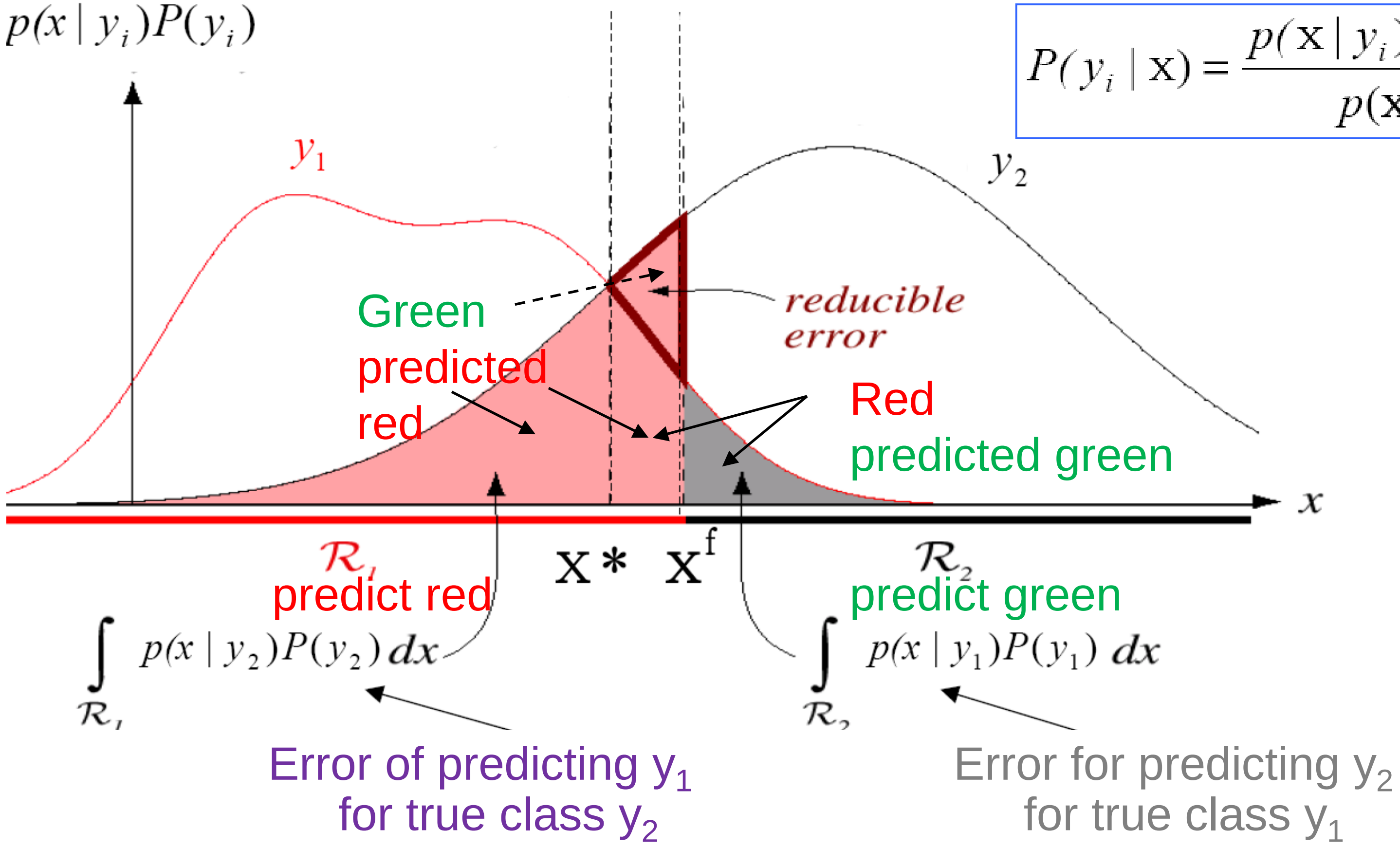
$$P(y_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y_i) P(y_i)}{p(\mathbf{x})}$$



Optimal Bayes classifier

- Classifier with the lowest possible error for a given distribution (decision threshold x* (the *optimal Bayes classifier*) is better then any other threshold x^f)

$$P(y_i \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y_i)P(y_i)}{p(\mathrm{x})}$$

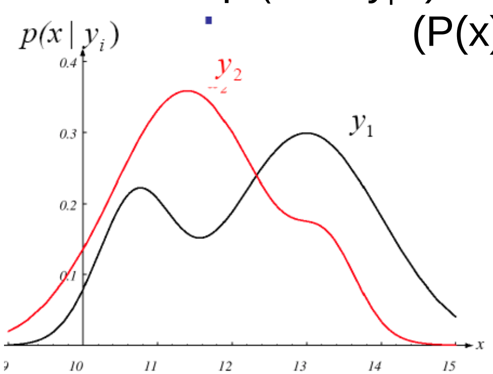


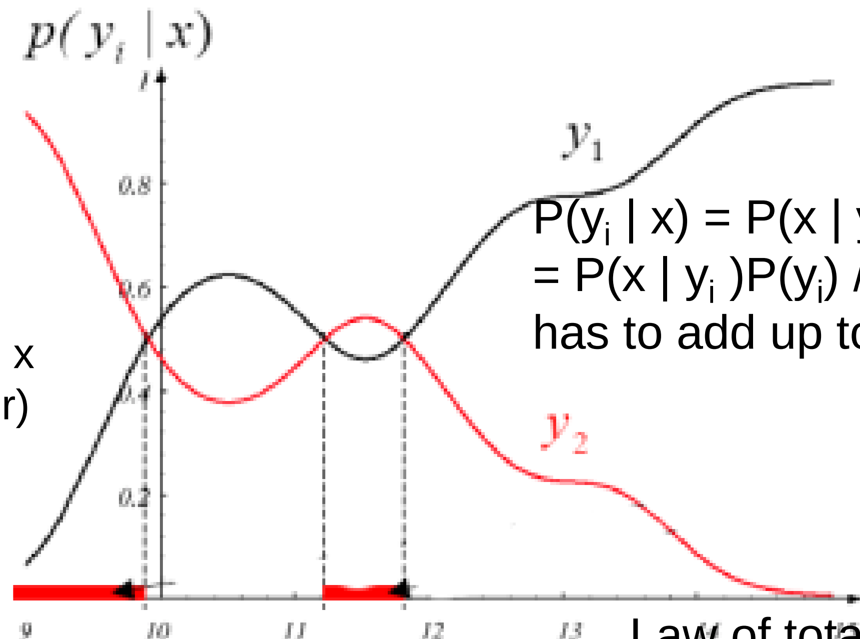
Optimal Bayes classifier

- Probabilistic decision making
  - If we know the probability P(x|y) and P(y)
    (or, we know P(x,y)) then we can make
    best possible predictions (lowest expected future error rate)
    - By using optimal Bayes classifier:
    - Predict class y_i with highest $P(y_i \mid x) = P(x \mid y_i)P(y_i) / P(x)$

Which to use:
$P(x \mid y_i)P(y_i)$
or
$P(x \mid y_i)$

Don't add up (over y_i's) to 1 for each x
(P(x) may differ)





$P(y_i \mid x) = P(x \mid y_i)P(y_i) / P(x)$
$= P(x \mid y_i)P(y_i) / \sum_i P(x \mid y_i)P(y_i)$
has to add up to 1 for each x

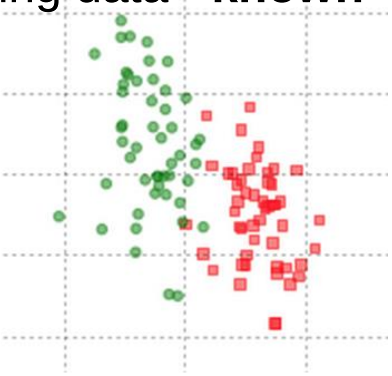
Law of total probability:
$P(x) = \sum_i P(x \mid y_i)P(y_i)$

Back to reality

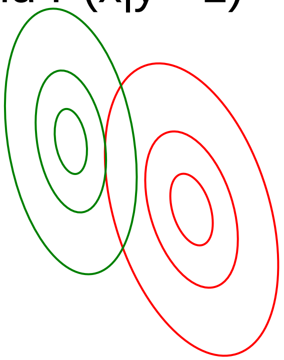
- We can make optimal predictions using P(y|x)
  or P(x|y) and P(y)

- But most of the time we don't know these distributions!

Training data - **known**



P(x|y=1) and P(x|y=-1) - **unknown**



ML: Typical assumptions

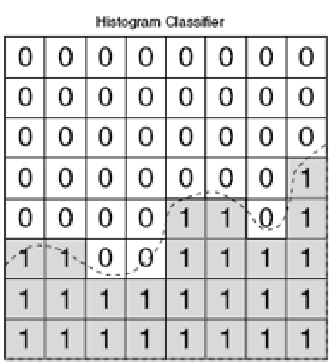
- The modeled phenomenon is poorly understood / too complex to simulate
- The features are somewhat informative but not perfectly correlated with the class
- The association between *regions of feature space* and the *class variable* is fixed
- The association between features and class we can learn is likely to be accurate only for objects similar to our training set

- The association between features and class is given by a probability distribution [over the *feature space* x *class*]
  - The distribution is fixed, but it is unknown to us!
  - We only observe training samples randomly sampled from it

Back to reality

- We can do predictions using P(y|x)
  - But we don't know the distribution!

- Machine Learning Approach #1: estimate the joint distribution over (x,y) directly from samples
  - Assume feature is discrete (e.g. v.short, …, medium) or discretize it
  - Use training set to get P(x,y) for each (x,y) combination
  - Calculate P(y|x)=P(x,y)/P(x), use it for making predictions
  - We always pick class y_i with highest $P(y_i \mid x)$
  - It's called: "*Histogram classifier*"

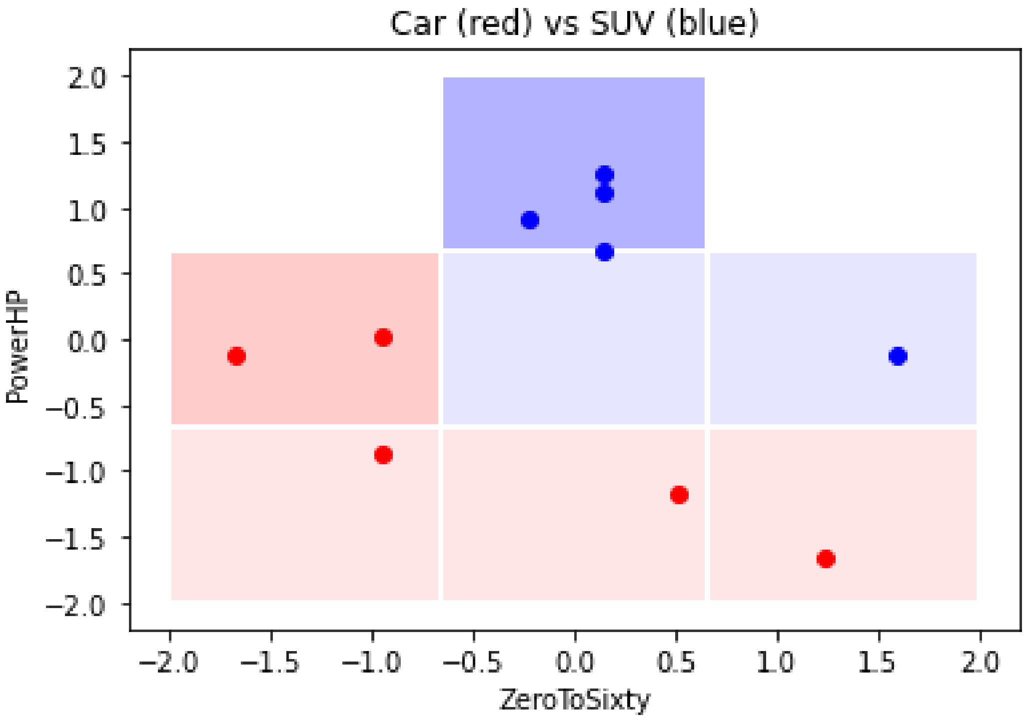


	D
P(x1=0,y=0)	0.25
P(x1=0,y=1)	0.05
P(x1=1,y=0)	0.2
P(x1=1,y=1)	0.25
P(x1=2,y=0)	0.1
P(x1=2,y=1)	0.15

$D_{Y\mid X}$		
P(y=0|x1=0)	0.833333	0.25/0.3
P(y=1|x1=0)	0.166667	0.05/0.3
P(y=0|x1=1)	0.444444	0.2/0.45
P(y=1|x1=1)	0.555556	0.25/0.45
P(y=0|x1=2)	0.4	0.1/0.25
P(y=1|x1=2)	0.6	0.15/0.25

ML Approach #1

Two classes: car or SUV (HW1 data)



- We binned feature values into 3 bins: low, medium, high
- We end up with 9 combinations
- We use training set (5 cars + 5 SUVs) to estimate 9 probability values:
  - p(car | 0-60=low, power = medium) = 100%
- We use those 9 probabilities to make predictions for new vehicles

Big problem: 9 numbers (bins/probs.) to learn/estimate, but only 10 training examples

- p(car | 0-60=low, power = medium) = 2/2 = 100%
- p(car | 0-60=medium, power = high) = 0/3 = 0%

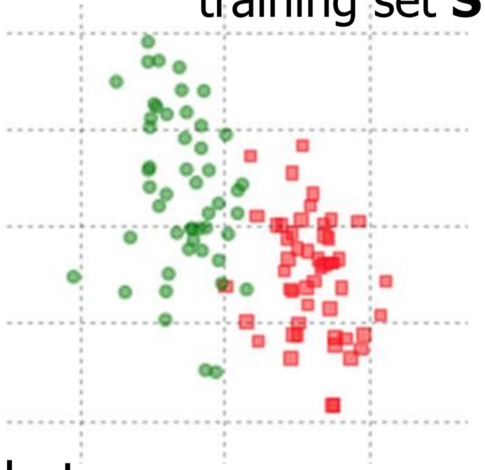
Problems with Approach #1

- Approach #1:
  - Assume feature is discrete (e.g. v.short, short, medium)
  - Use training set to get P(x,y) for each (x,y) combination
- Continuous data: we need to do binning/discretization (e.g. 150-160, 160-170, etc)
  - Loss of precision
- If we have e.g. 20 features, each with only 2 possible values, we'll have 2^{20} (~1mln) possible feature combination
  - If we have only 1000 training samples, most P(x,y) will be 0
  - We won't be able to make predictions
- Approach #1 works only if we have few (x,y) combinations!

ML Approach #2

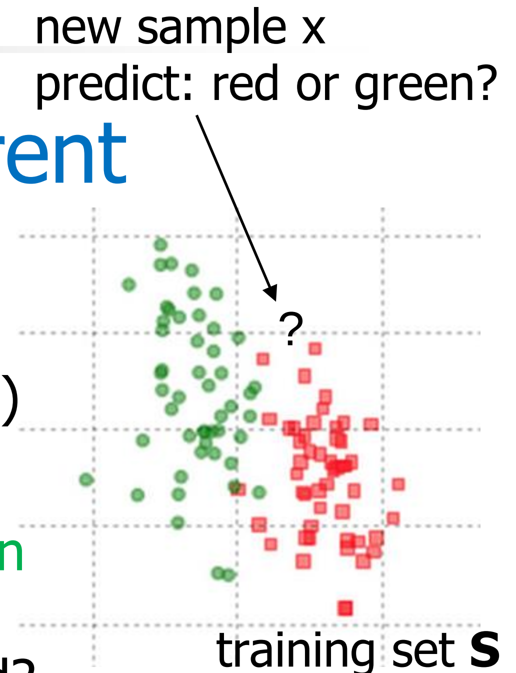
- Let's try to come up with a different approach for learning from data
  - For predictions, we want p(y|x),
    or we can also make it work if we have p(x|y)
  - We want to make use of the training data (training set **S**)
    - So, our conditional distributions will reflect that:
      - p(y|x,S)
        or
        p(x|y,S)

training set **S**



ML Approach #2

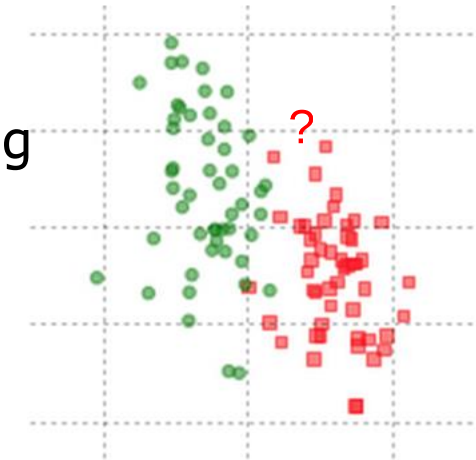
- Let's try to come up with a different approach for learning from data
  - For predictions, we want p(y|x,S),
    or we can also make it work if we have p(x|y,S)
  - If you have seen m_r red examples and m_g green examples in the training set, what's the probability that a sample x at position "?" is red? Or green?
    - i.e., p(y=red | x=…, trainingSet = S)
  - If you have seen m_r red examples in the training set, what's the probability of seeing "red ?"
    - i.e., p(x=… | y=red, trainingSet = S)

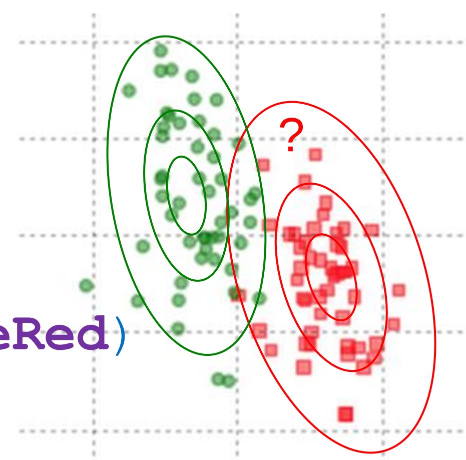


We could do histogram (est. probabilities in bins), but, most often, "too few samples, too many bins"!

ML Approach #2

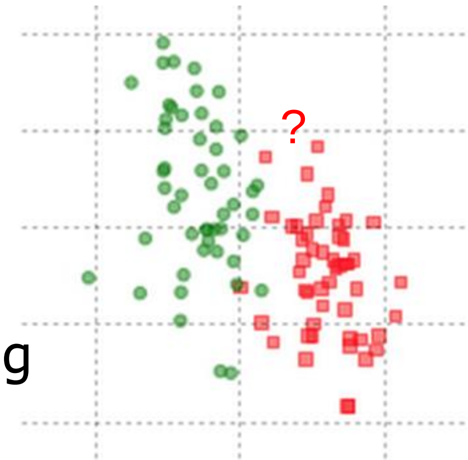
- Let's try to come up with a different approach for learning from data
  - If you have seen m_r red examples in the training set, what's the probability of seeing "red ?"
    - i.e., p(x=… | y=red, trainingSet = S)
  - **Much easier question to answer if we have a specific distribution shape (e.g. Gaussian) chosen to fit the data**
  - **Much fewer "numbers" to learn/guess!**
    - p(x=… | y=red, trainingSet = S) = `distr_x_given_classRed.pdf(x)`
    - `distr_x_given_classRed= multivariate_normal (mean=`**`meanRed`**`,cov=`**`covarianceRed`**`)`
      - no "x=…,trainingSet=S", but: red gaussian mean & cov. depend on S



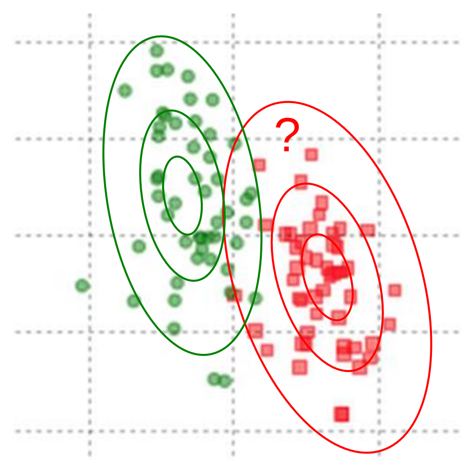


ML Approach #2

- Let's try to come up with a different approach for learning from data



  - For predictions, we want p(y|x,S) or p(x|y,S)
  - If you have seen m_r red examples in the training set, what's the probability of seeing "red ?"
    - p(x=… | y=red, trainingSet = S) = `distr_x_given_classRed.pdf(x)`

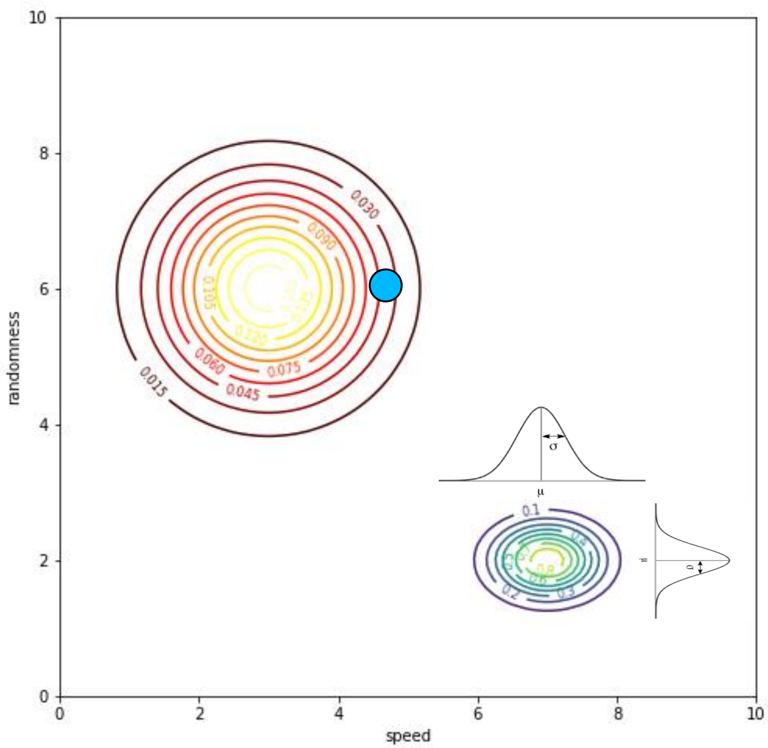


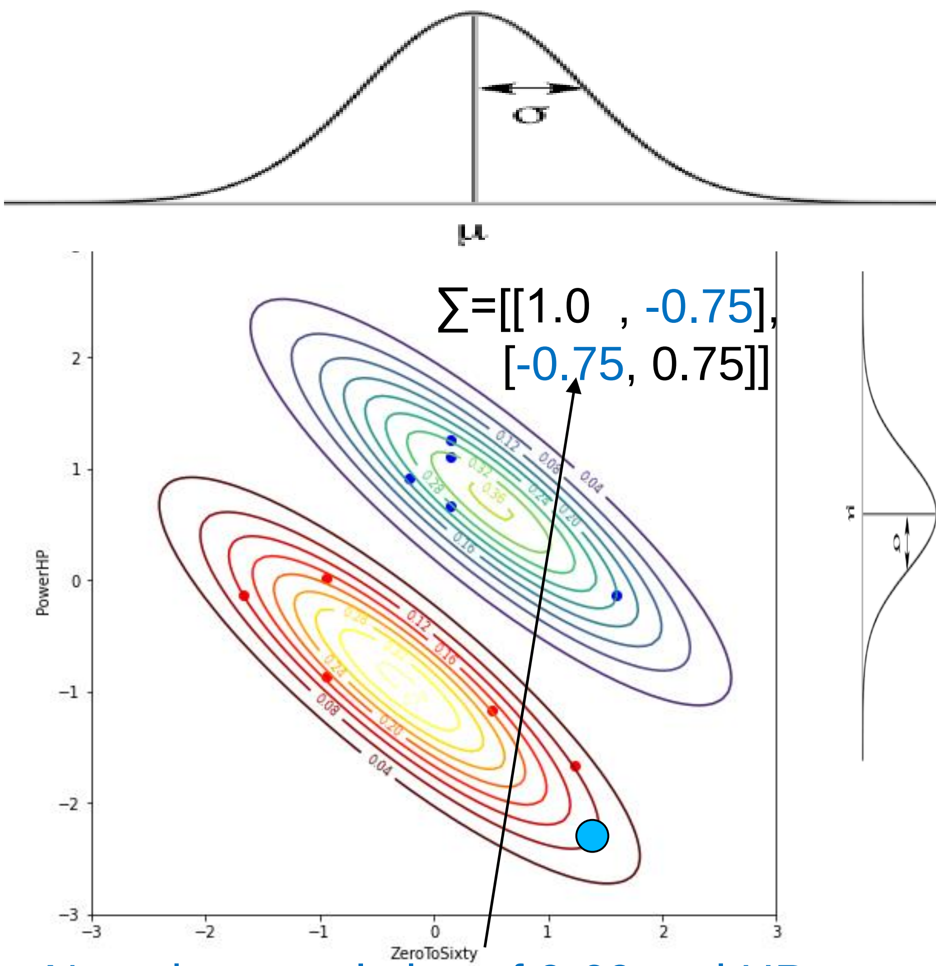
  - **We just need to figure out, based on training data S, what are the shapes of the red and green gaussian distributions**
    - **Their means (where's the center)**
    - **Their other parameters**

Normal (Gaussian) dist.

- Let's assume that conditional distribution p(x|y) for each class is multivariate normal distribution:

$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$





Negative correlation of 0-60 and HP

Normal (Gaussian) dist.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- Let's assume that conditional distribution p(x|y) for each class is multivariate normal distribution: $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

a constant to make is sum to 1 to be a proper PDF

  - Defined over d-dimensional vectors x
  - Defined by d-dimension mean μ

$$\boldsymbol{\mu} \equiv \mathcal{E}[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x}) \, d\mathbf{x}$$

$$\mu_i = \mathcal{E}[x_i]$$

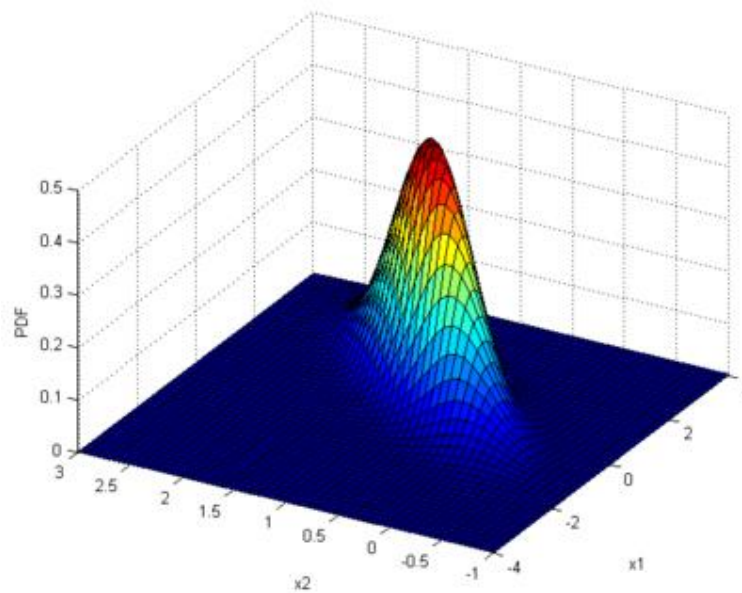
  - and *d* x *d* matrix of covariance between variables Σ

$$\boldsymbol{\Sigma} \equiv \mathcal{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t] = \int (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t p(\mathbf{x}) \, d\mathbf{x}$$

$$\sigma_{ij} = \mathcal{E}[(x_i - \mu_i)(x_j - \mu_j)]$$

$$\boldsymbol{\mu} = [x_1, x_2]^T = [1, 1]^T$$

$$\Sigma = \begin{bmatrix} 0.9 & 0.4 \\ 0.4 & 0.3 \end{bmatrix}$$

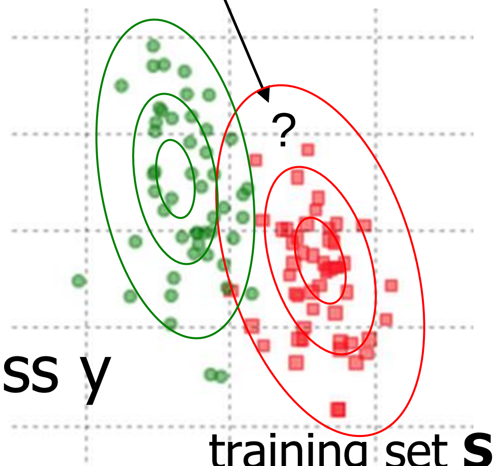


ML Approach #2

- Approach #2: fitting a distribution to each class

```
distr_x_given_classYi(params θi).pdf(x)
```

new sample x
predict: red or green?



training set **S**

- We have a **training set S**, and
  for **a new sample x** we want to predict its class y
- $\mathbf{p(y_i \mid x,S) = p(x \mid y_i,S)\, p(y_i \mid S) \,/\, p(x|S)}$
  - $P(y_i,x,S)=P(y_i|x,S)P(x,S)=P(y_i|x,S)P(x|S)P(S)$
    $P(y_i,x,S)=P(x|y_i,S)P(y_i,S)=P(x|y_i,S)P(y_i|S)P(S)$

Bayes Theorem

- $\mathbf{p(y_i \mid x,S) = p(x \mid y_i,S)\, p(y_i|S) \,/\, \Sigma_i\, p(x \mid y_i,S)\, P(y_i|S)}$
  - $\Sigma_i p(x|y_i,S)P(y_i|S) = \Sigma_i p(x|y_i,S)P(y_i|S)P(S)/P(S)$
    $=\Sigma_i p(x|y_i,S)P(y_i,S)/P(S) = 1/P(S) * \Sigma_i p(x,y_i,S)$
    $=1/P(S) * p(x,S) = P(x|S)\, P(S)/P(S)$

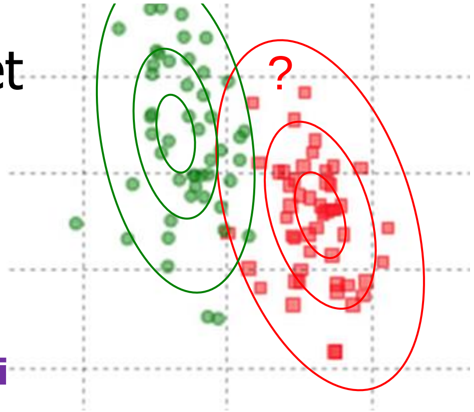
$P(A)=\Sigma_i P(A|B_i)P(B_i)$
$P(A|C)=\Sigma_i P(A|B_i C)P(B_i|C)$

ML Approach #2

$P(A)=\Sigma_i P(A|B_i)P(B_i)$
$P(A|C)=\Sigma_i P(A|B_i C)P(B_i|C)$

- $p(y_i \mid x,S) = p(x \mid y_i,S)\, p(y_i \mid S) / \Sigma_i\, p(x \mid y_i,S)\, P(y_i \mid S)$
  - To make predictions, we need:
    - $p(x \mid y_i,S)$
    - If you have seen m_r red examples in the training set, what's the probability of seeing "red ?"

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

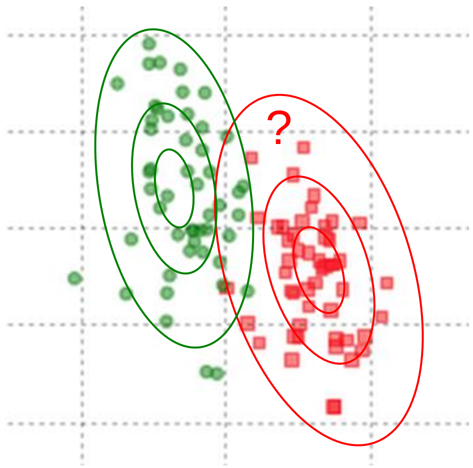


    - Easier if we fit a distribution to the training set
      - $p(x \mid y_i,S) =$ `gaussian(θi).pdf(x)`
        $=\mathcal{N}(x,\theta_i)$
        $=\mathcal{N}(x,\mu_i,\Sigma_i)$
    - **But how to choose distrib. parameters θ_i**
      - **E.g., for Gaussians where $\theta_i=(\mu_i, \Sigma_i)$, how to choose means μ_i and covariances Σ_i**
    - **Find θ_i that best fits the training data S**

ML Approach #2

$P(A)=\Sigma_i P(A|B_i)P(B_i)$
$P(A|C)=\Sigma_i P(A|B_i C)P(B_i|C)$

- To make predictions, we need:
  - $p(x \mid y_i, S)$ = `gaussian(θ_i).pdf(x)` $=\mathcal{N}(x,\theta_i)=\mathcal{N}(x,\mu_i,\Sigma_i)$



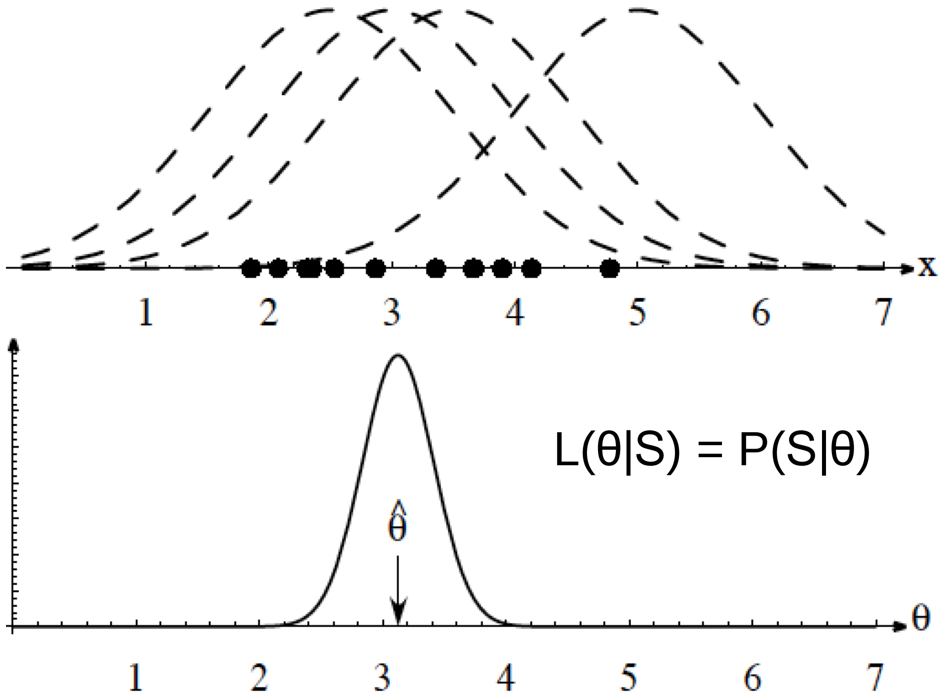
  - But how to choose θ_i?
  - Find θ_i that best fits the training data S
    - What does "best fits" mean?

- Idea: for each possible value of θ_i,
  we can calculate the *probability* $p(S_i \mid \theta_i)$
  of seeing the training data set S_i that we have
    - Then, pick the value θ_i that leads to highest probability
      $P(S_i|\theta_i) = L(\theta_i \mid S_i)$ = *likelihood* of θ_i given dataset S_i

The approach of choosing parameters that make the dataset most probable is called ***maximum likelihood estimation*** *(MLE)*

1D Gaussian example

- We assume θ_i for each class is fixed, but unknown to us
  - Some values of θ_i make the training set S_i more likely
  - Some values of θ_i make the training set S_i less likely



- Pick the value θ_i with **highest likelihood** given S,
  that is
  Pick θ_i that leads to **highest** (compared to other possible θ) **probability** p(S | θ) **of seeing** our **dataset** S

ML Approach #2

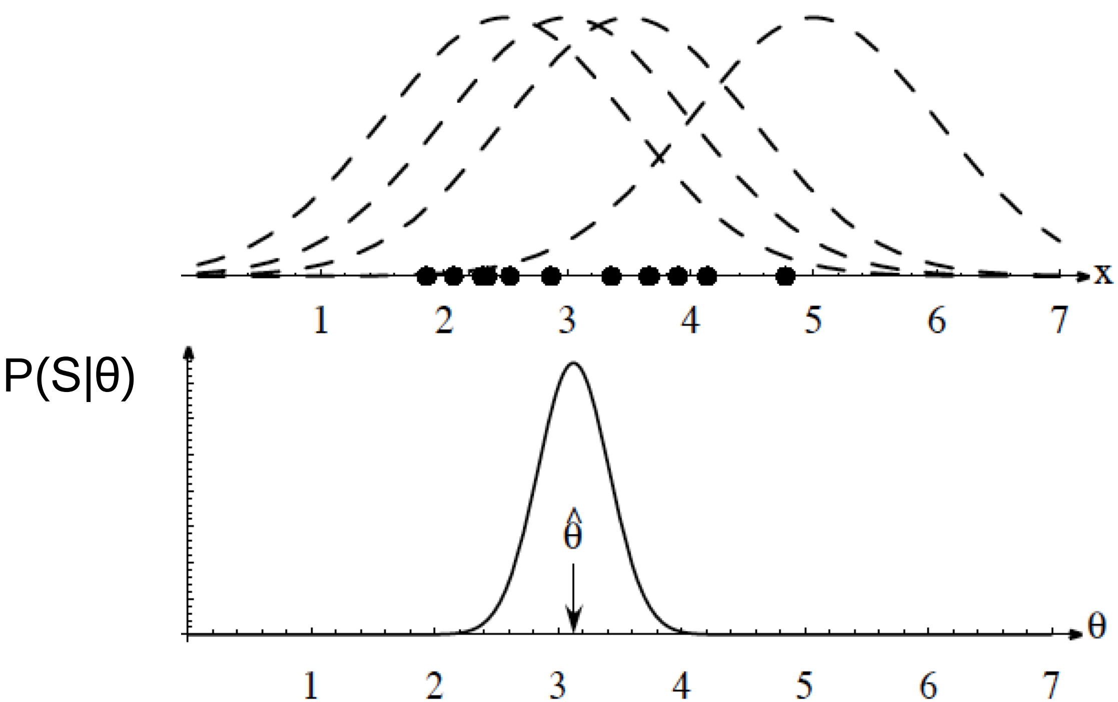
- Find: θ_i with highest $p(S_i \mid \theta_i)$
  - How to calculate $p(S_i \mid \theta_i)$
  - We have a training set S_i of m samples from class i
    - We assume each sample in class y_i in S_i is independently drawn from the same distribution
      - samples are i.i.d – independent, identically distributed
        - A,B are independent ⇔ P(A and B)=P(A) x P(B)
          - e.g. P(6 on dice 1 and 6 on dice 2) = 1/6 x 1/6
      - $P(S_i|\theta_i) = P(\{x_1,x_2,x_3,\ldots,x_m\} \mid \theta_i)$

        and if samples i.i.d. then

      - $P(S_i|\theta_i) = P(x_1|\theta_i) \times P(x_2|\theta_i) \times P(x_3|\theta_i) \times \ldots = \Pi_{k=1,..,m} P(x_k|\theta_i)$

1D Gaussian example

- Finding θ_i that maximizes probability of seeing this particular training set
  - Choose θ_i to maximize $P(S_i|\theta_i) = \Pi_k P(x_k|\theta_i)$
- We assume θ_i for each class is fixed, but unknown to us
  - Some values of θ_i make the training set S_i more likely
  - Some values of θ_i make the training set S_i less likely



- How to deal with the multiplication?
- Transform it into a sum, easier to deal with mathematically!
  - ln(exp(z))=z
    and p(z) starts with exp in Gaussians

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- Same as:
  choose θ_i to maximize
  $\ln P(S_i|\theta_i) = \Sigma_k \ln P(x_k|\theta_i)$

1D Gaussian example (grad)

- Maximize: $\ln P(S_i|\theta_i) = \Sigma_k \ln P(x_k|\theta_i)$
- How?
  - *We have the mathematical formula for the distribution P to help us!*
  - Example: we have 1 feature x, and we know $P(x|\theta)$ is Gaussian
    - The unknown parameter θ is a single number, the mean of the Gaussian (for simplicity, we only estimate mean, not std.dev in this example)
      - $P(x \mid \theta_i) = (2\pi\sigma^2)^{-.5} \exp(-(x-\theta_i)^2/2\sigma^2)$
      - $\ln P(x \mid \theta_i) = -.5 \ln 2\pi\sigma^2 - (x-\theta_i)^2/2\sigma^2$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

    - At maximum (we're doing MLE), derivative is 0:
      - $d\, \Sigma_k \ln P(x_k|\theta_i) / \; d\, \theta_i = 0$
    - $d\, \Sigma_k [-.5 \ln 2\pi\sigma^2 - (x-\theta_i)^2/2\sigma^2] / \; d\, \theta_i = 0$
    - Solve for θ_i:
      - $-\, 1/2\sigma^2 \; \Sigma_k \; d\, (x_k-\theta_i)^2 / \; d\, \theta_i = 0$
      - $\Sigma_{k=1,..,m} \; d\, (x_k^2 + \theta_i^2 - 2x_k\theta_i) / \; d\, \theta_i = 0$
      - $m 2\theta_i - \Sigma_{k=1,..,m} 2x_k = 0$
    - $\theta_i = 1/m \; \Sigma_{k=1,..,m} \; x_k$ *we just derived "average" as the estimator of mean*

Scheduling

- HW1 is due tomorrow at 5pm
  - If you see 14/14 in Gradescope, perfect
  - If less than 14pts, debug (look at what Gradescope returns) and resubmit!
- HW2 will be announced on Monday, 9/16
  - It will be due on Tuesday, 10/1, 5pm
- Study problems for Test 1 will show up in Canvas on Wed. 9/25
- On Wednesday, 10/2 we'll discuss study problems for Test 1
- On Monday, 10/7, we will have Test 1 (fully online, via Canvas)