

# Probabilistic modeling

Subhransu Maji

CMPSCI 689: Machine Learning

3 March 2015

5 March 2015

## Overview

- ◆ So far the **models** and **algorithms** you have learned about are relatively disconnected
- ◆ **Probabilistic modeling** framework unites the two
- ◆ Learning can be viewed as **statistical inference**
- ◆ Two kinds of **data models**
  - **Generative**
  - **Conditional**
- ◆ Two kinds of **probability models**
  - **Parametric**
  - **Non-parametric**

## Administrivia

- ◆ Mini-project 1 due Thursday, March 05
- ◆ Turn in a hard copy
  - In the next class
  - Or in CS main office reception area by 4:00pm (mention 689 hw)
- ◆ Clearly write your **name** and **student id** in the front page
- ◆ Late submissions:
  - At most 48 hours at 50% deduction (by 4:00pm March 07)
  - More than 48 hours get zero
  - Submit a pdf via email to the TA: [xiaojian@cs.umass.edu](mailto:xiaojian@cs.umass.edu)

## Classification by density estimation

- ◆ The data is generated according to a **distribution**  $D$
- $$(\mathbf{x}, y) \sim D(\mathbf{x}, y)$$
- ◆ Suppose you had access to  $D$ , then **classification** becomes simple:

$$\hat{y} = \arg \max_y D(\hat{\mathbf{x}}, y)$$

- ◆ This is the **Bayes optimal classifier** which achieves the **smallest expected loss** among all classifiers

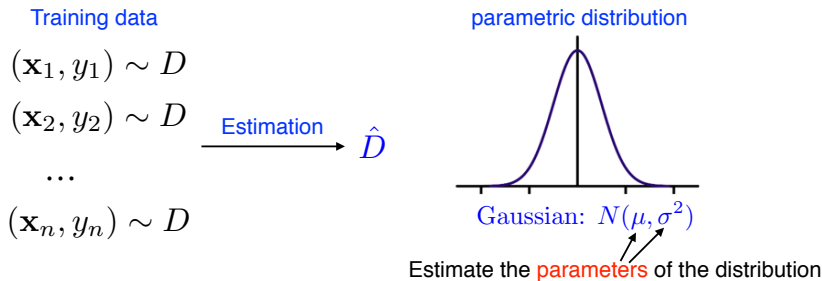
$$\epsilon(\hat{y}) = \mathbb{E}_{(\mathbf{x}, y) \sim D} [\ell(y, \hat{y})] : \text{expected loss of a predictor}$$

$$y \in \{0, 1\} \quad \ell(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$$

- ◆ Unfortunately, we don't have access to the **distribution**

# Classification by density estimation

- ◆ This suggests that one way to learn a **classifier** is to estimate  $D$



- ◆ We will assume that each point is **independently** generated from  $D$ 
  - ▶ A new point **doesn't** depend on previous points
  - ▶ Commonly referred to as the **i.i.d** assumption or **independently and identically distributed** assumption

# Statistical estimation

- ◆ **Coin toss:** observed sequence  $\{H, T, H, H\}$
- ◆ **Probability of H:**  $\beta$
- ◆ What is the value of  $\beta$  that best explains the observed data?
- ◆ **Maximum likelihood principle (MLE):** pick parameters of the distribution that maximize the likelihood of the observed data
- ◆ **Likelihood of data:**



$$p_\beta(\text{data}) = p_\beta(H, T, H, H)$$

$$= p_\beta(H)p_\beta(T)p_\beta(H)p_\beta(H) \quad \text{i.i.d data}$$

$$= \beta \times (1 - \beta) \times \beta \times \beta$$

$$= \beta^3(1 - \beta)$$

- ◆ **Maximize likelihood:**

$$\frac{dp_\beta(\text{data})}{d\beta} = \frac{d\beta^3(1 - \beta)}{d\beta} = 3\beta^2(1 - \beta) + \beta^3(-1) = 0 \implies \beta = \frac{3}{4}$$

# Log-likelihood

- ◆ It is convenient to maximize the **logarithm** of the **likelihood** instead
- ◆ **Log-likelihood** of the **observed data:**

$$\log p_\beta(\text{data}) = \log p_\beta(H, T, H, H)$$

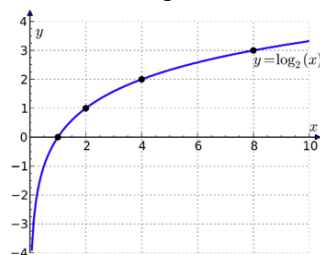
$$= \log p_\beta(H) + \log p_\beta(T) + \log p_\beta(H) + \log p_\beta(H)$$

$$= \log \beta + \log(1 - \beta) + \log \beta + \log \beta$$

$$= 3 \log \beta + \log(1 - \beta)$$

- ◆ Maximizing the **log-likelihood** is equivalent to maximizing **likelihood**

- ▶ Log is a concave monotonic function
- ▶ Products become sums
- ▶ Numerically stable



# Log-likelihood

- ◆ **Log-likelihood** of observing **H**-many heads and **T**-many tails:

$$\log p_\beta(\text{data}) = H \log \beta + T \log(1 - \beta)$$

- ◆ Maximizing the **log-likelihood:**

$$\frac{d[H \log \beta + T \log(1 - \beta)]}{d\beta} = \frac{H}{\beta} - \frac{T}{1 - \beta} = 0$$

$$\implies \beta = \frac{H}{H + T}$$

## Rolling a die

- ◆ Suppose you are rolling a **k-sided die** with parameters:  $\theta_1, \theta_2, \dots, \theta_k$
- ◆ You observe:  $x_1, x_2, \dots, x_k$
- ◆ **Log-likelihood** of the data:

$$\log p(\text{data}) = \sum_k x_k \log \theta_k$$

- ◆ Maximizing the **log-likelihood** by setting the derivative to zero:

$$\frac{d \log p(\text{data})}{d\theta_k} = \frac{x_k}{\theta_k} = 0 \implies \theta_k = \infty$$

- ◆ We need **additional constraints**:

$$\sum_k \theta_k = 1$$



## Lagrangian multipliers

- ◆ **Constrained optimization**:

$$\max_{\theta_1, \theta_2, \dots, \theta_k} \sum_k x_k \log \theta_k$$

$$\text{subject to: } \sum_k \theta_k = 1$$

- ◆ **Unconstrained optimization**:

$$\min_{\lambda} \max_{\{\theta_1, \theta_2, \dots, \theta_k\}} \sum_k x_k \log \theta_k + \lambda \left( 1 - \sum_k \theta_k \right)$$

▸ At optimality:  $\frac{x_k}{\theta_k} = \lambda \implies \theta_k = \frac{x_k}{\lambda}$

$$\lambda = \sum_k x_k \quad \theta_k = \frac{x_k}{\sum_k x_k}$$

## Naive Bayes

- ◆ Consider the **binary prediction** problem
- ◆ Let the data be distributed according to a probability distribution:

$$p_{\theta}(y, \mathbf{x}) = p_{\theta}(y, x_1, x_2, \dots, x_D)$$

- ◆ We can simplify this using the **chain rule** of probability:

$$\begin{aligned} p_{\theta}(y, \mathbf{x}) &= p_{\theta}(y) p_{\theta}(x_1|y) p_{\theta}(x_2|x_1, y) \dots p_{\theta}(x_D|x_1, x_2, \dots, x_{D-1}, y) \\ &= p_{\theta}(y) \prod_{d=1}^D p_{\theta}(x_d|x_1, x_2, \dots, x_{d-1}, y) \end{aligned}$$

- ◆ **Naive Bayes** assumption:

$$p_{\theta}(x_d|x_{d'}, y) = p_{\theta}(x_d|y), \forall d' \neq d$$

- ◆ E.g., The words “free” and “money” are independent given **spam**

## Naive Bayes

- ◆ **Naive Bayes** assumption:

$$p_{\theta}(x_d|x_{d'}, y) = p_{\theta}(x_d|y), \forall d' \neq d$$

- ◆ We can simplify the joint probability distribution as:

$$\begin{aligned} p_{\theta}(y, \mathbf{x}) &= p_{\theta}(y) \prod_{d=1}^D p_{\theta}(x_d|x_1, x_2, \dots, x_{d-1}, y) \\ &= p_{\theta}(y) \prod_{d=1}^D p_{\theta}(x_d|y) \quad // \text{ simpler distribution} \end{aligned}$$

- ◆ At this point we can start parametrizing the distribution

## Naive Bayes: a simple case

- Case: **binary labels** and **binary features**

$$\left. \begin{aligned} p_{\theta}(y) &= \text{Bernoulli}(\theta_0) \\ p_{\theta}(x_d|y=1) &= \text{Bernoulli}(\theta_d^+) \\ p_{\theta}(x_d|y=-1) &= \text{Bernoulli}(\theta_d^-) \end{aligned} \right\} \text{1+2D parameters}$$

- Probability of the data:

$$\begin{aligned} p_{\theta}(y, \mathbf{x}) &= p_{\theta}(y) \prod_{d=1}^D p_{\theta}(x_d|y) \\ &= \theta_0^{[y=+1]} (1 - \theta_0)^{[y=-1]} \\ &\dots \times \prod_{d=1}^D \theta_d^{+[x_d=1, y=+1]} (1 - \theta_d^+)^{[x_d=0, y=+1]} \quad // \text{label +1} \\ &\dots \times \prod_{d=1}^D \theta_d^{-[x_d=1, y=-1]} (1 - \theta_d^-)^{[x_d=0, y=-1]} \quad // \text{label -1} \end{aligned}$$

## Naive Bayes: parameter estimation

- Given data we can estimate the **parameters** by maximizing **data likelihood**
- Similar to the coin toss example the **maximum likelihood** estimates are:

$$\hat{\theta}_0 = \frac{\sum_n [y_n = +1]}{N} \quad // \text{fraction of the data with label as +1}$$

$$\hat{\theta}_d^+ = \frac{\sum_n [x_{d,n} = 1, y_n = +1]}{\sum_n [y_n = +1]} \quad // \text{fraction of the instances with 1 among +1}$$

$$\hat{\theta}_d^- = \frac{\sum_n [x_{d,n} = 1, y_n = -1]}{\sum_n [y_n = -1]} \quad // \text{fraction of the instances with 1 among -1}$$

- Other cases:

- Nominal features:** Multinomial distribution (like rolling a die)
- Continuous features:** Gaussian distribution

inductive bias

## Naive Bayes: prediction

- To make **predictions** compute the **posterior distribution**:

$$\hat{y} = \arg \max_y p_{\theta}(y|\mathbf{x}) \quad // \text{Bayes optimal prediction}$$

$$= \arg \max_y \frac{p_{\theta}(y, \mathbf{x})}{p_{\theta}(\mathbf{x})} \quad // \text{Bayes rule}$$

$$= \arg \max_y p_{\theta}(y, \mathbf{x})$$

- For **binary labels** we can also compute the **likelihood ratio**:

$$\text{LR} = \frac{p_{\theta}(+1, \mathbf{x})}{p_{\theta}(-1, \mathbf{x})} \quad \hat{y} = \begin{cases} +1 & \text{LR} \geq 1 \\ -1 & \text{otherwise} \end{cases}$$

- Or the **log likelihood ratio**:

$$\text{LLR} = \log(p_{\theta}(+1, \mathbf{x})) - \log(p_{\theta}(-1, \mathbf{x})) \quad \hat{y} = \begin{cases} +1 & \text{LLR} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

## Naive Bayes: decision boundary

$$\begin{aligned} \text{LLR} &= \log(p_{\theta}(+1, \mathbf{x})) - \log(p_{\theta}(-1, \mathbf{x})) \\ &= \log\left(\theta_0 \prod_{d=1}^D \theta_d^{+[x_d=1]} (1 - \theta_d^+)^{[x_d=0]}\right) - \log\left((1 - \theta_0) \prod_{d=1}^D \theta_d^{-[x_d=1]} (1 - \theta_d^-)^{[x_d=0]}\right) \\ &= \log \theta_0 - \log(1 - \theta_0) + \sum_{d=1}^D [x_d = 1] (\log \theta_d^+ - \log \theta_d^-) \\ &\dots + \sum_{d=1}^D [x_d = 0] (\log(1 - \theta_d^+) - \log(1 - \theta_d^-)) \\ &= \log\left(\frac{\theta_0}{1 - \theta_0}\right) + \sum_{d=1}^D [x_d = 1] \log\left(\frac{\theta_d^+}{\theta_d^-}\right) + \sum_{d=1}^D [x_d = 0] \log\left(\frac{1 - \theta_d^+}{1 - \theta_d^-}\right) \\ &= \log\left(\frac{\theta_0}{1 - \theta_0}\right) + \sum_{d=1}^D x_d \log\left(\frac{\theta_d^+}{\theta_d^-}\right) + \sum_{d=1}^D (1 - x_d) \log\left(\frac{1 - \theta_d^+}{1 - \theta_d^-}\right) \\ &= \log\left(\frac{\theta_0}{1 - \theta_0}\right) + \sum_{d=1}^D x_d \left(\log\left(\frac{\theta_d^+}{\theta_d^-}\right) - \log\left(\frac{1 - \theta_d^+}{1 - \theta_d^-}\right)\right) + \sum_{d=1}^D \log\left(\frac{1 - \theta_d^+}{1 - \theta_d^-}\right) \\ &= \mathbf{w}^T \mathbf{x} + b \end{aligned}$$

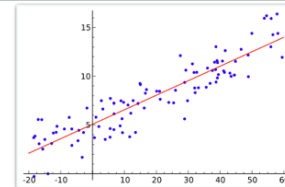
Naive Bayes classifier has a linear decision boundary!

# Generative and conditional models

- ◆ **Generative models:**
  - ▶ Model the **joint distribution**  $p(\mathbf{x}, y)$
  - ▶ Use Bayes rule to compute the label posterior
  - ▶ Need to make simplifying assumptions (e.g. **Naive bayes**)
- ◆ **In most cases we are given  $\mathbf{x}$  and are only interested in the labels  $y$**
- ◆ **Conditional models:**
  - ▶ Model the distribution  $p(y | \mathbf{x})$
  - ▶ Saves some modeling effort
  - ▶ Can assume a simpler parametrization of the distribution  $p(y | \mathbf{x})$
  - ▶ Most of ML we did so far directly aimed at predicting  $y$  from  $\mathbf{x}$

# Conditional models: regression

- ◆ Assume that  $y$  has a **linear relationship** with  $\mathbf{x}$
- ◆ **Generative story** of the dataset:
  - ▶ For  $i = 1$  to  $N$ ,
    - Compute:  $t_n = \mathbf{w}^T \mathbf{x}_n$
    - Compute:  $\epsilon_n = N(0, \sigma^2)$
    - Compute:  $y_n = t_n + \epsilon_n$



- ◆ This can be written as:  $y_n \sim N(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$ , and

$$p(y_n | \mathbf{x}_n) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(y_n - \mathbf{w}^T \mathbf{x}_n)^2}{2\sigma^2}\right)$$

- ◆ The **log-likelihood** of the dataset is:

$$\log(\mathcal{D}) = \sum_n -\frac{(y_n - \mathbf{w}^T \mathbf{x}_n)^2}{2\sigma^2} + \text{constants}$$

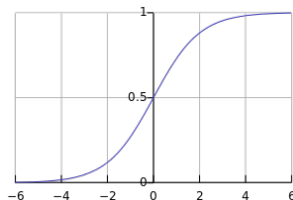
Maximizing log-likelihood is equivalent to minimizing squared error

# Conditional models: classification

- ◆ The **sigmoid function**:

$$\sigma(z) = \frac{1}{1 + \exp[-z]}$$

- ▶ Maps  $-\infty \rightarrow 0, \infty \rightarrow 1$
- ▶  $\sigma(-z) = 1 - \sigma(z)$ ,
- ▶  $d\sigma/dz = \sigma(z)(1 - \sigma(z))$



- ◆ **Generative story** of the data:

- ▶ For  $i = 1$  to  $N$ ,
  - Compute:  $t_n = \sigma(\mathbf{w}^T \mathbf{x}_n)$
  - Compute:  $z_n = \text{Bernoulli}(t_n)$
  - Compute:  $y_n = 2z_n - 1$

# Conditional models: classification

- ◆ The **log-likelihood** of the dataset is:

$$\begin{aligned} \log(\mathcal{D}) &= \sum_n [y_n = +1] \log \sigma(\mathbf{w}^T \mathbf{x}_n) + [y_n = -1] \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \\ &= \sum_n [y_n = +1] \log \sigma(\mathbf{w}^T \mathbf{x}_n) + [y_n = -1] \log(\sigma(-\mathbf{w}^T \mathbf{x}_n)) \\ &= \sum_n \log \sigma(y_n \mathbf{w}^T \mathbf{x}_n) \\ &= \sum_n -\log(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)) \\ &= \sum_n -\ell^{(\log)}(y_n, \mathbf{w}^T \mathbf{x}_n) \end{aligned}$$

// ignoring constants

Maximizing log-likelihood is equivalent to minimizing logistic loss

This is also called as logistic regression

# Regularization with priors

- ◆ **Coin toss:** {H,H,H,H} →  $\beta = 1$
- ◆ **Maximum likelihood estimation (MLE):**

$$\arg \max_{\theta} p(\mathcal{D}|\theta) \quad \text{likelihood}$$

- ◆ **Maximum a-posteriori estimation (MAP):**

$$\arg \max_{\theta} p(\theta|\mathcal{D}) = \arg \max_{\theta} \frac{p(\theta, \mathcal{D})}{p(\mathcal{D})} = \arg \max_{\theta} \overset{\text{prior}}{p(\theta)} \overset{\text{likelihood}}{p(\mathcal{D}|\theta)}$$

$$p(\mathcal{D}) = \int_{\theta} p(\theta, \mathcal{D}) d\theta \quad \text{data probability}$$

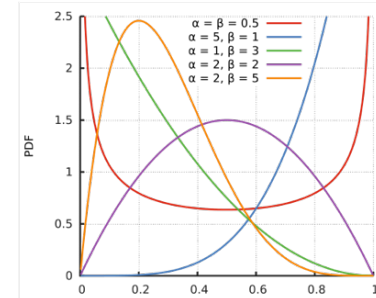
- ◆ **MAP estimation in log space:**  $\arg \max_{\theta} [\log p(\theta) + \log p(\mathcal{D}|\theta)]$

# Regularization with priors: coin toss

- ◆ **Beta distribution** as a prior on  $\beta$

$$\text{Beta}(\beta; a, b) = c\beta^{a-1}(1-\beta)^{b-1}$$

$$\text{Mode} = \frac{a-1}{a+b-2}$$



- ◆ **Posterior** over  $\beta$  given the prior and **H**-many heads and **T**-many tails:

$$p(\beta|\mathcal{D}) \propto p(\beta)p(\mathcal{D}|\beta)$$

$$\propto \beta^{a-1}(1-\beta)^{b-1}\beta^H(1-\beta)^T = \text{Beta}(a+H, b+T)$$

$$\beta_{\text{MAP}} = \frac{a+H-1}{a+H+b+T-2}$$

$$\beta_{\text{MLE}} = \frac{H}{H+T}$$

# Conjugate priors

- ◆ If the **prior** and **posterior** are in the same family, then the **prior** is **conjugate** to the **likelihood**

$$\overset{\text{posterior}}{p(\theta|\mathcal{D})} \propto \overset{\text{prior}}{p(\theta)} \overset{\text{likelihood}}{p(\mathcal{D}|\theta)}$$

- ◆ **Beta** is **conjugate** to **Bernoulli**

- ▶ **Prior:** Beta(a, b)     **Data:** {H, T}     **Posterior:** Beta(a+H, b+T)
- ▶ Interpretation of the prior: pseudo count of a-1 heads and b-1 tails

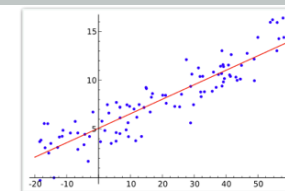
- ◆ **Dirichlet** is **conjugate** to **Multinomial**

- ▶ **Prior:** Dirichlet( $\theta; a_1, a_2, \dots, a_n$ )  $\propto \prod_k \theta_k^{a_k-1}$
- ▶ **Data:** { $k_1, k_2, \dots, k_n$ } occurrence of each side
- ▶ **Posterior:** Dirichlet( $\theta; a_1 + k_1, a_2 + k_2, \dots, a_n + k_n$ )
- ▶ Interpretation of the prior: pseudo count of  $a_i-1$  for the  $i^{\text{th}}$  side

# Regularization with priors: regression

- ◆ Assume that **y** has a **linear relationship** with **x**
- ◆ **Generative story** of the dataset:

- ▶ For  $i = 1$  to  $N$ ,
  - Compute:  $t_n = \mathbf{w}^T \mathbf{x}_n$
  - Compute:  $\epsilon_n = N(0, \sigma^2)$
  - Compute:  $y_n = t_n + \epsilon_n \longrightarrow y_n \sim N(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$



- ◆ Assume a **Gaussian prior** on the weights:

$$p(\mathbf{w}) = N(0_D, \tau^2 \mathbf{I}_D) = c \exp\left(-\sum_i \frac{w_i^2}{2\tau^2}\right) \quad c \exp\left(-\sum_i \frac{|w_i|}{b}\right)$$

Laplace prior for  $l_1$

- ◆ MAP estimate of **w**:

$$\arg \max_{\mathbf{w}} \sum_i -\frac{w_i^2}{2\tau^2} + \sum_n -\frac{(y_n - \mathbf{w}^T \mathbf{x}_n)^2}{2\sigma^2} + \text{constants}$$

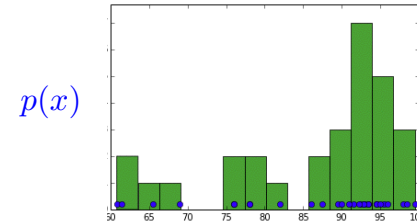
MAP is same as  $l_2$  regularized least-squares regression

# Non-parametric density models

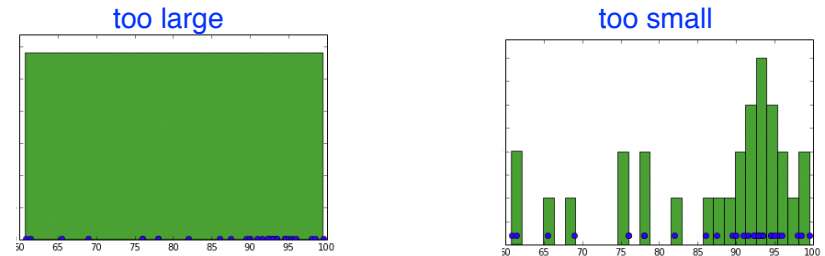
- ◆ So far we assumed that the probability distribution was **parametric**
  - Gaussian distribution, Binomial distribution, etc
  - This allowed us to estimate the data distribution by estimating the parameters of the probability distribution
- ◆ However, the data distribution can be complicated
  - For example there might be multiple modes
- ◆ **Non-parametric** density models offer a flexible alternative

# Density estimation using histograms

- ◆ This is the simplest example of a **non-parametric** density model

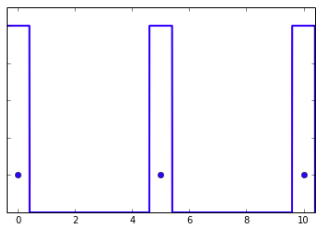


- ◆ The bin size is a hyperparameter of the model



# Kernel density estimation

- ◆ **Histograms** are sums of **delta functions** centered at each point



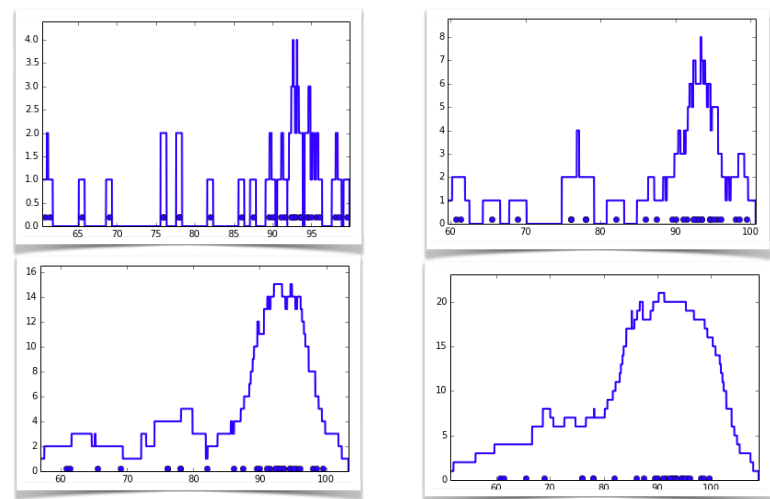
$$p(x) = \frac{1}{N} \sum_{i=1}^N K(x - x_i)$$

$$K(x - x_i) = \begin{cases} \frac{1}{b} & |x - x_i| \leq \frac{b}{2} \\ 0 & \text{otherwise} \end{cases}$$

- ◆ The hyperparameter  $b$  controls the width of the **delta function**
- ◆ The function  $K$  is called the **kernel function**
- ◆ These **density estimators** are also called as **Parzen window estimators**
- ◆ Set hyperparameters by cross-validation
  - MLE estimate is  $b=0$ . This is clearly wrong (overfitting)

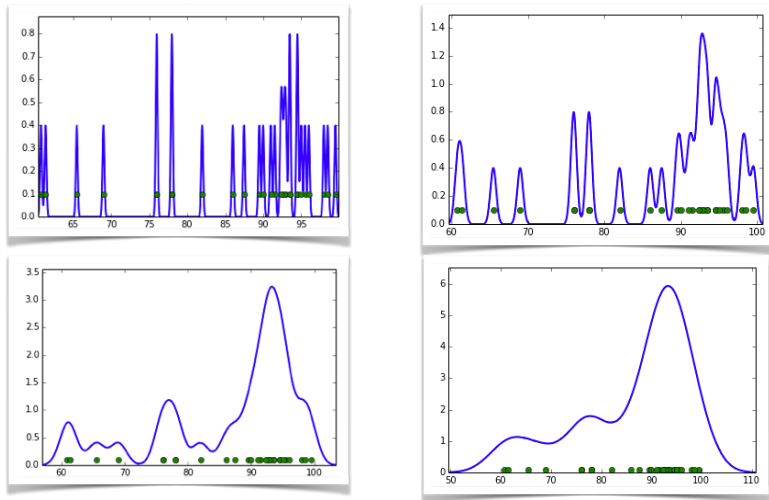
# Kernel density estimation: example

Rectangle kernel  $K(x - x_i) = \begin{cases} \frac{1}{b} & |x - x_i| \leq \frac{b}{2} \\ 0 & \text{otherwise} \end{cases}$



# Kernel density estimation: example

Gaussian kernel 
$$K(x - x_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)$$



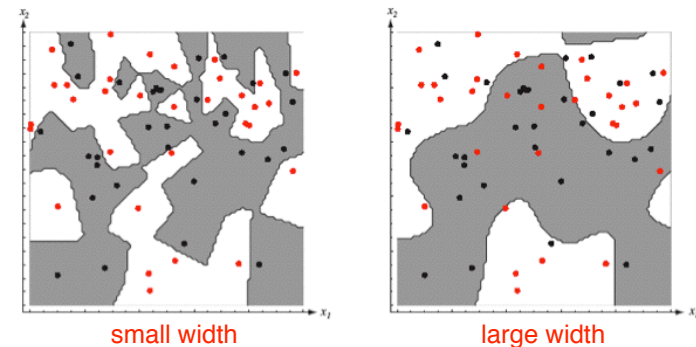
CMPSCI 689

Subhransu Maji (UMASS)

29/32

# Kernel density classifier

- ◆ Estimate  $p(\mathbf{x} | +1)$  and  $p(\mathbf{x} | -1)$  separately
- ◆ Compute likelihood ratio:  $p(+1)p(\mathbf{x} | +1) / p(-1)p(\mathbf{x} | -1)$ 
  - Predict class +1 if LR > 1



- ◆ kNN classifier is a Kernel density classifier with kernel width = distance to the  $k^{\text{th}}$  nearest neighbor

Figure from Duda et al.

CMPSCI 689

Subhransu Maji (UMASS)

30/32

# Summary

- ◆ Probabilistic modeling views learning as statistical inference
- ◆ Two ways to estimate parameters of the distribution
  - Maximum likelihood: maximize  $p(\mathcal{D}|\theta)$
  - Maximum a-posteriori: maximize  $p(\theta|\mathcal{D})$
- ◆ Two kinds of data models
  - Generative:  $p(y, \mathbf{x})$ 
    - Example: Naive Bayes, Kernel density
  - Conditional:  $p(y | \mathbf{x})$ 
    - Example: Linear and logistic regression
- ◆ Two kinds of probability models
  - Parametric: Gaussian, Bernoulli, etc
    - Learning by MLE and MAP
  - Non-parametric: kernel density estimators
    - Learning by cross validation

CMPSCI 689

Subhransu Maji (UMASS)

31/32

# Slides credit

- ◆ Figure of the logistic and linear regression are from Wikipedia
- ◆ Figure of the beta distribution is from Wikipedia
- ◆ Figures for kernel density estimation are from <http://www.mglerner.com/blog/?p=28> (the page has an interactive demo)
- ◆ Parzen window figure: "Pattern Classification", Duda, Hart & Stork
- ◆ Some slides are based on the CIML book by Hal Daume III

CMPSCI 689

Subhransu Maji (UMASS)

32/32