

# COMP 875 Project/Survey Proposals

- Due Tuesday, September 29
- Submission: e-mail me a PDF document (about two pages)

# COMP 875 Project/Survey Proposals

- Due Tuesday, September 29
- Submission: e-mail me a PDF document (about two pages)
- For survey paper:
  - Topic description, your goals, questions to be answered
  - Outline of proposed paper
  - Reference list

- Due Tuesday, September 29
- Submission: e-mail me a PDF document (about two pages)
- For survey paper:
  - Topic description, your goals, questions to be answered
  - Outline of proposed paper
  - Reference list
- For project:
  - Problem definition, hypothesis (if applicable)
  - Outline of proposed work
  - Resources: code, data available on the Internet
  - Potential problems, issues

- The kernel trick

- The kernel trick
- Types of kernels

- The kernel trick
- Types of kernels
- Multi-class SVMs

- Learning  $\approx$  inferring (fitting) model parameters based on training data.

- Learning  $\approx$  inferring (fitting) model parameters based on training data.
- Is SVM classifier

$$\text{sign} \left( w_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

parametric?

- Learning  $\approx$  inferring (fitting) model parameters based on training data.
- Is SVM classifier

$$\text{sign} \left( w_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

parametric?

- The Lagrange multipliers  $\alpha$  are kind of parameters.
- However, we also need to keep around some (or possibly all!) of the training data.

- Learning  $\approx$  inferring (fitting) model parameters based on training data.
- Is SVM classifier

$$\text{sign} \left( w_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

parametric?

- The Lagrange multipliers  $\alpha$  are kind of parameters.
- However, we also need to keep around some (or possibly all!) of the training data.
- In **nonparametric methods** the training examples are explicitly used as parameters.

- Learning  $\approx$  inferring (fitting) model parameters based on training data.
- Is SVM classifier

$$\text{sign} \left( w_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

parametric?

- The Lagrange multipliers  $\alpha$  are kind of parameters.
- However, we also need to keep around some (or possibly all!) of the training data.
- In **nonparametric methods** the training examples are explicitly used as parameters.

- Generic description:
  - Memorize training  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ .
  - Given test  $\mathbf{x}$  predict

$$\hat{y} = f(\mathbf{x}, \mathbf{x}_1, y_1, \dots, \mathbf{x}_N, y_N).$$

- This is **lazy learning**: (almost) all the work is done at test time.
- The function  $f$  is typically expressed in terms of similarity of  $\mathbf{x}$  to the training examples.

- Generic description:
  - Memorize training  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ .
  - Given test  $\mathbf{x}$  predict

$$\hat{y} = f(\mathbf{x}, \mathbf{x}_1, y_1, \dots, \mathbf{x}_N, y_N).$$

- This is **lazy learning**: (almost) all the work is done at test time.
- The function  $f$  is typically expressed in terms of similarity of  $\mathbf{x}$  to the training examples.
- **What nonparametric method have we seen already?**

# Nearest Neighbor Classification

- **Asymptotic performance:** As the amount of data approaches infinity, the nearest neighbor classifier is guaranteed to yield an error rate no worse than twice the Bayes error rate

# Nearest Neighbor Classification

- **Asymptotic performance:** As the amount of data approaches infinity, the nearest neighbor classifier is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

# Nearest Neighbor Classification

- **Asymptotic performance:** As the amount of data approaches infinity, the nearest neighbor classifier is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).
- The k-NN classifier is guaranteed to approach the Bayes error rate, provided k increases suitably as a function of the number of data points.

# Nearest Neighbor Classification

- **Asymptotic performance:** As the amount of data approaches infinity, the nearest neighbor classifier is guaranteed to yield an error rate no worse than twice the Bayes error rate (**the minimum achievable error rate given the distribution of the data**).
- The k-NN classifier is guaranteed to approach the Bayes error rate, provided k increases suitably as a function of the number of data points.
- In theory, the rate of convergence to the bounds can be arbitrarily slow, though in practice, k-NN classifiers can work very well.

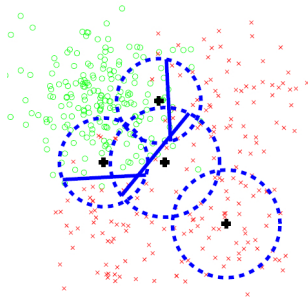
# Nearest Neighbor Classification

- **Asymptotic performance:** As the amount of data approaches infinity, the nearest neighbor classifier is guaranteed to yield an error rate no worse than twice the Bayes error rate (**the minimum achievable error rate given the distribution of the data**).
- The k-NN classifier is guaranteed to approach the Bayes error rate, provided k increases suitably as a function of the number of data points.
- In theory, the rate of convergence to the bounds can be arbitrarily slow, though in practice, k-NN classifiers can work very well.
  - **Two active research areas:** learning distance functions, accelerating nearest-neighbor search.

# Example of a “lazy learning” method: SVM-KNN

For each test point:

- 1 Find  $k$  nearest neighbors. Can use either kernel function or a faster “approximate” distance.
- 2 If all  $k$  neighbors are from the same class, we’re done.
- 3 Otherwise, train an SVM on the  $k$  examples and use it to classify the test point.



H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition, CVPR 2006.

- The problem of probability density estimation: Suppose we have observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  drawn from some unknown probability density function  $p(\mathbf{x})$ . Estimate  $p(\mathbf{x})$ .
  - What kind of learning problem is this?

- The problem of probability density estimation: Suppose we have observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  drawn from some unknown probability density function  $p(\mathbf{x})$ . Estimate  $p(\mathbf{x})$ .
  - What kind of learning problem is this?
- Parametric estimation: assume a parametric form  $p(\mathbf{x}; \theta)$  and estimate  $\theta$  using, e.g., **maximum likelihood (ML)** or **maximum a posteriori (MAP)**:

- The problem of probability density estimation: Suppose we have observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  drawn from some unknown probability density function  $p(\mathbf{x})$ . Estimate  $p(\mathbf{x})$ .
  - What kind of learning problem is this?
- Parametric estimation: assume a parametric form  $p(\mathbf{x}; \theta)$  and estimate  $\theta$  using, e.g., **maximum likelihood (ML)** or **maximum a posteriori (MAP)**:

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} p(\mathbf{x}; \theta).$$

- The problem of probability density estimation: Suppose we have observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  drawn from some unknown probability density function  $p(\mathbf{x})$ . Estimate  $p(\mathbf{x})$ .
  - What kind of learning problem is this?
- Parametric estimation: assume a parametric form  $p(\mathbf{x}; \theta)$  and estimate  $\theta$  using, e.g., **maximum likelihood** (ML) or **maximum a posteriori** (MAP):

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} p(\mathbf{x}; \theta).$$

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} p(\mathbf{x}; \theta)p(\theta).$$

- The problem of probability density estimation: Suppose we have observations  $\mathbf{x}_1, \dots, \mathbf{x}_N$  drawn from some unknown probability density function  $p(\mathbf{x})$ . Estimate  $p(\mathbf{x})$ .
  - What kind of learning problem is this?
- Parametric estimation: assume a parametric form  $p(\mathbf{x}; \theta)$  and estimate  $\theta$  using, e.g., **maximum likelihood (ML)** or **maximum a posteriori (MAP)**:

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} p(\mathbf{x}; \theta).$$

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} p(\mathbf{x}; \theta)p(\theta).$$

- The idea behind nonparametric estimation: directly evaluate how dense is the vicinity of  $\mathbf{x}$ .

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .
- Assume that region  $R$  is sufficiently small so that  $p(\mathbf{x})$  is approximately constant within it. Then  $P \approx$

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .
- Assume that region  $R$  is sufficiently small so that  $p(\mathbf{x})$  is approximately constant within it. Then  $P \approx p(\mathbf{x})V$ , where  $V$  is the volume of  $R$ .

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .
- Assume that region  $R$  is sufficiently small so that  $p(\mathbf{x})$  is approximately constant within it. Then  $P \approx p(\mathbf{x})V$ , where  $V$  is the volume of  $R$ .
- This gives us the density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .
- Assume that region  $R$  is sufficiently small so that  $p(\mathbf{x})$  is approximately constant within it. Then  $P \approx p(\mathbf{x})V$ , where  $V$  is the volume of  $R$ .
- This gives us the density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

- Assumptions:  $R$  is small enough so that the density is almost constant inside it, yet large enough so that  $k/N$  gives us a good estimate of  $P$ .

# Nonparametric density estimation

- Given:  $N$  observations drawn from unknown density  $p(\mathbf{x})$ .
- Given a small region  $R$  containing  $\mathbf{x}$ , the probability of each observation ending up inside  $R$  is  $P = \int_R p(\mathbf{x})d\mathbf{x}$ .
- The expected number of points that end up inside  $R$  is  $k \approx NP$ .
- Assume that region  $R$  is sufficiently small so that  $p(\mathbf{x})$  is approximately constant within it. Then  $P \approx p(\mathbf{x})V$ , where  $V$  is the volume of  $R$ .
- This gives us the density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

- Assumptions:  $R$  is small enough so that the density is almost constant inside it, yet large enough so that  $k/N$  gives us a good estimate of  $P$ .
- **Curse of dimensionality:** the number of samples required for a good local estimate grows exponentially as a function of data dimension.

# Nearest-neighbor density estimation

## Density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

We have two options:

- Fix  $V$  and determine  $k$  from data.

# Nearest-neighbor density estimation

## Density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

We have two options:

- Fix  $V$  and determine  $k$  from data.
  - Put a sphere of volume  $V$  centered at  $\mathbf{x}$  and count how many data points fall within the sphere.

# Nearest-neighbor density estimation

## Density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

We have two options:

- Fix  $V$  and determine  $k$  from data.
  - Put a sphere of volume  $V$  centered at  $\mathbf{x}$  and count how many data points fall within the sphere.
- Fix  $k$  and determine  $V$  from data –  $k$ -NN density estimate.

# Nearest-neighbor density estimation

## Density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

We have two options:

- Fix  $V$  and determine  $k$  from data.
  - Put a sphere of volume  $V$  centered at  $\mathbf{x}$  and count how many data points fall within the sphere.
- Fix  $k$  and determine  $V$  from data –  $k$ -NN density estimate.
  - Put a sphere centered at point  $\mathbf{x}$  and let it grow until it contains  $k$  data points.

## Density estimate

$$\hat{p}(\mathbf{x}) = \frac{k}{NV}.$$

We have two options:

- Fix  $V$  and determine  $k$  from data.
  - Put a sphere of volume  $V$  centered at  $\mathbf{x}$  and count how many data points fall within the sphere.
- Fix  $k$  and determine  $V$  from data –  $k$ -NN density estimate.
  - Put a sphere centered at point  $\mathbf{x}$  and let it grow until it contains  $k$  data points.
- Both estimation methods converge to the true probability density in the limit  $N \rightarrow \infty$  provided  $V$  shrinks suitably with  $N$  and  $k$  grows with  $N$  (Duda and Hart, 1973).

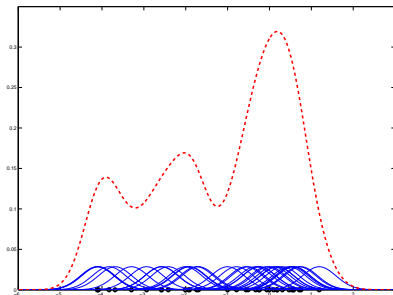
- Instead of a “hard” spherical window function, we can use a kernel function that is also a probability density.
  - e.g., Gaussian kernel  $K(\mathbf{x}, \mathbf{x}_i) = \mathcal{N}(\mathbf{x} - \mathbf{x}_i; 0, \sigma^2 \mathbf{I})$ .

- Instead of a “hard” spherical window function, we can use a kernel function that is also a probability density.
  - e.g., Gaussian kernel  $K(\mathbf{x}, \mathbf{x}_i) = \mathcal{N}(\mathbf{x} - \mathbf{x}_i; 0, \sigma^2 \mathbf{I})$ .

- Estimator:

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}, \mathbf{x}_i).$$

This is known as the **Parzen window estimator**.

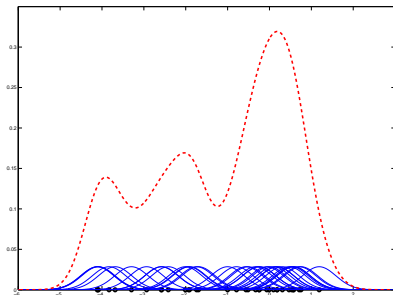


- Instead of a “hard” spherical window function, we can use a kernel function that is also a probability density.
  - e.g., Gaussian kernel  $K(\mathbf{x}, \mathbf{x}_i) = \mathcal{N}(\mathbf{x} - \mathbf{x}_i; 0, \sigma^2 \mathbf{I})$ .

- Estimator:

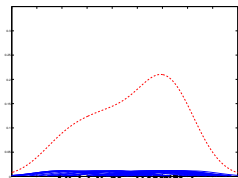
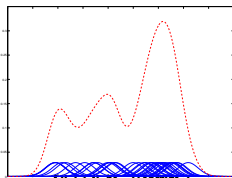
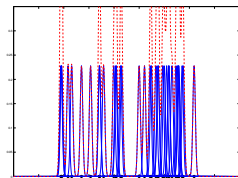
$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}, \mathbf{x}_i).$$

This is known as the **Parzen window estimator**.



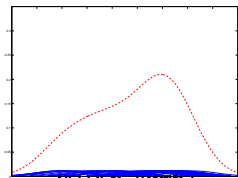
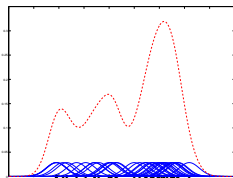
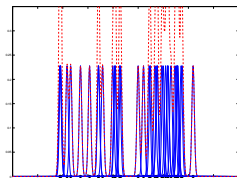
- The estimated density function is a sum of “bumps” centered at training points.

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \mathcal{N}(\mathbf{x} - \mathbf{x}_i; 0, \sigma^2 \mathbf{I})$$

 $\sigma = 1$  $\sigma = 0.4$  $\sigma = 0.05$ 

- Choice of the *kernel width*  $\sigma$  is crucial.

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \mathcal{N}(\mathbf{x} - \mathbf{x}_i; 0, \sigma^2 \mathbf{I})$$

 $\sigma = 1$  $\sigma = 0.4$  $\sigma = 0.05$ 

- Choice of the *kernel width*  $\sigma$  is crucial.
  - Bias-variance tradeoff in unsupervised learning!

The Bayes classifier:

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x} | c) p(c).$$

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x}|c)p(c).$$

- $p(c|\mathbf{x})$  is the **posterior**,  $p(\mathbf{x}|c)$  is the **likelihood** or **class-conditional distribution**,  $p(c)$  is the **prior**.

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x}|c)p(c).$$

- $p(c|\mathbf{x})$  is the **posterior**,  $p(\mathbf{x}|c)$  is the **likelihood** or **class-conditional distribution**,  $p(c)$  is the **prior**.

We can find kernel density estimates of class-conditionals  $p(\mathbf{x}|c)$  using only examples from class  $c$ :

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x}|c)p(c).$$

- $p(c|\mathbf{x})$  is the **posterior**,  $p(\mathbf{x}|c)$  is the **likelihood** or **class-conditional distribution**,  $p(c)$  is the **prior**.

We can find kernel density estimates of class-conditionals  $p(\mathbf{x}|c)$  using only examples from class  $c$ :

$$\hat{p}(\mathbf{x}|c) = \frac{1}{N_c} \sum_{y_i=c} K(\mathbf{x}, \mathbf{x}_i),$$

where  $N_c$  is the # of examples from class  $c$ .  
The estimate of prior probability for class  $c$  is

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x}|c)p(c).$$

- $p(c|\mathbf{x})$  is the **posterior**,  $p(\mathbf{x}|c)$  is the **likelihood** or **class-conditional distribution**,  $p(c)$  is the **prior**.

We can find kernel density estimates of class-conditionals  $p(\mathbf{x}|c)$  using only examples from class  $c$ :

$$\hat{p}(\mathbf{x}|c) = \frac{1}{N_c} \sum_{y_i=c} K(\mathbf{x}, \mathbf{x}_i),$$

where  $N_c$  is the # of examples from class  $c$ .

The estimate of prior probability for class  $c$  is  $\hat{p}(c) = N_c/N$ .

Therefore, the Bayes classifier is

The Bayes classifier:

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x}) \propto \operatorname{argmax}_c p(\mathbf{x}|c)p(c).$$

- $p(c|\mathbf{x})$  is the **posterior**,  $p(\mathbf{x}|c)$  is the **likelihood** or **class-conditional distribution**,  $p(c)$  is the **prior**.

We can find kernel density estimates of class-conditionals  $p(\mathbf{x}|c)$  using only examples from class  $c$ :

$$\hat{p}(\mathbf{x}|c) = \frac{1}{N_c} \sum_{y_i=c} K(\mathbf{x}, \mathbf{x}_i),$$

where  $N_c$  is the # of examples from class  $c$ .

The estimate of prior probability for class  $c$  is  $\hat{p}(c) = N_c/N$ .

Therefore, the Bayes classifier is

$$\hat{c}(\mathbf{x}) = \operatorname{argmax}_c \sum_{y_i=c} K(\mathbf{x}, \mathbf{x}_i).$$

## Aside: Discriminative vs. generative models

- **Generative models:** Learn classifiers by modeling class-conditional densities and priors.

## Aside: Discriminative vs. generative models

- **Generative models:** Learn classifiers by modeling class-conditional densities and priors.
- **Discriminative models:** Attempt to estimate posterior or decision boundary directly.
  - Example: logistic regression – connection between posterior probabilities and decision boundary:

$$\log \frac{p(Y = 1|\mathbf{x})}{p(Y = -1|\mathbf{x})} = \log \frac{p(Y = 1|\mathbf{x})}{1 - p(Y = 1|\mathbf{x})} = w_0 + \mathbf{w}^T \mathbf{x}.$$

Posterior is estimated by the sigmoid function:

$$\eta(\mathbf{x}) = \frac{1}{1 + e^{-(w_0 + \mathbf{w}^T \mathbf{x})}}.$$

## Aside: Discriminative vs. generative models

- **Generative models:** Learn classifiers by modeling class-conditional densities and priors.
- **Discriminative models:** Attempt to estimate posterior or decision boundary directly.
  - Example: logistic regression – connection between posterior probabilities and decision boundary:

$$\log \frac{p(Y = 1|\mathbf{x})}{p(Y = -1|\mathbf{x})} = \log \frac{p(Y = 1|\mathbf{x})}{1 - p(Y = 1|\mathbf{x})} = w_0 + \mathbf{w}^T \mathbf{x}.$$

Posterior is estimated by the sigmoid function:

$$\eta(\mathbf{x}) = \frac{1}{1 + e^{-(w_0 + \mathbf{w}^T \mathbf{x})}}.$$

- Other discriminative models (classifiers) do not have probabilistic motivation and do not return estimates of posterior probabilities (e.g., SVMs).

# Nonparametric Regression

- *k*-NN regression: Let  $\mathbf{x}_1, \dots, \mathbf{x}_k$  be the neighbors of  $\mathbf{x}$ , and  $y_1, \dots, y_k$  their labels. Predict  $\hat{y} = \frac{1}{k} \sum_{i=1}^k y_i$ .

# Nonparametric Regression

- *k*-NN regression: Let  $\mathbf{x}_1, \dots, \mathbf{x}_k$  be the neighbors of  $\mathbf{x}$ , and  $y_1, \dots, y_k$  their labels. Predict  $\hat{y} = \frac{1}{k} \sum_{i=1}^k y_i$ .
- Nadaraya-Watson regression (1964):

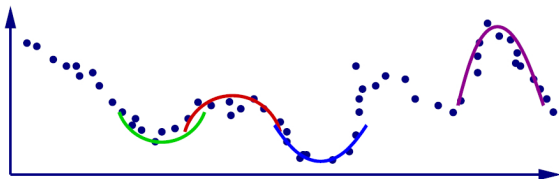
$$\hat{y} = \frac{\sum_{i=1}^n y_i K(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^n K(\mathbf{x}, \mathbf{x}_j)}.$$

# Nonparametric Regression

- *k*-NN regression: Let  $\mathbf{x}_1, \dots, \mathbf{x}_k$  be the neighbors of  $\mathbf{x}$ , and  $y_1, \dots, y_k$  their labels. Predict  $\hat{y} = \frac{1}{k} \sum_{i=1}^k y_i$ .
- Nadaraya-Watson regression (1964):

$$\hat{y} = \frac{\sum_{i=1}^n y_i K(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^n K(\mathbf{x}, \mathbf{x}_j)}.$$

- Locally weighted regression: Fit a (simple) parametric model to the neighbors of  $\mathbf{x}$ .



# Example: Local Linear Regression in 2D

Model:  $\hat{f}(x) = a(x) + b(x)x$ .

Training data:  $(x_1, y_1), \dots, (x_N, y_N)$ .

For each test point  $x$ , find parameters  $a(x)$ ,  $b(x)$  by minimizing weighted least-squares error:

$$\min_{a(x), b(x)} \sum_{i=1}^N K(x, x_i) [y_i - a(x) - b(x)x_i]^2$$

Solution:

$$[a(x), b(x)]^T = (\mathbf{X}^T \mathbf{K}(x) \mathbf{X})^{-1} \mathbf{X}^T \mathbf{K}(x) \mathbf{Y},$$

where  $\mathbf{X}$  is the  $N \times 2$  design matrix with  $i$ th row  $[1, x_i]$ ,  $\mathbf{Y} = [y_1, \dots, y_N]^T$ , and  $\mathbf{K}$  the  $N \times N$  diagonal matrix with  $i$ th diagonal element  $K(x, x_i)$ .