

# EECS 545: Machine Learning

## Lecture 4. Classification

Honglak Lee

1/22/2025



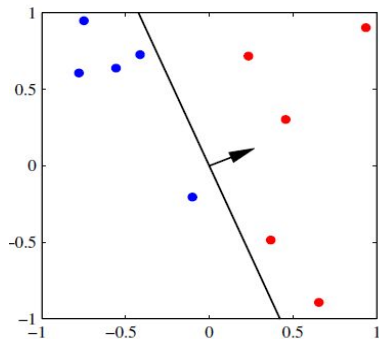
# Outline

- Logistic regression
- Newton's method
- K-nearest neighbors (KNN)

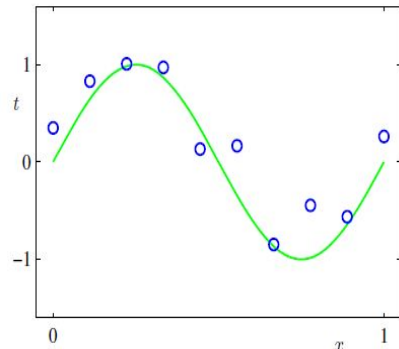
# Supervised learning: classification

# Supervised learning

- Goal:
  - Given data  $X$  in feature space and labels  $Y$
  - Learn to predict  $Y$  from  $X$
- Labels could be discrete or continuous
  - Discrete-valued labels: classification (today's topic)
  - Continuous-valued labels: regression



classification



regression

# Classification problem

- The task of classification:
  - Given an input vector  $\mathbf{x}$ , assign it to one of  $K$  distinct classes  $C_k$  where  $k = 1, \dots, K$
- Representing the assignment:
  - For  $K = 2$ :
    - $y = 1$  means that  $\mathbf{x}$  is in  $C_1$
    - $y = 0$  means that  $\mathbf{x}$  is in  $C_2$ .
    - (Sometimes,  $y = -1$  can be used depending on algorithms)
- For  $K > 2$ :
  - Use 1-of- $K$  coding
  - e.g.,  $\mathbf{y} = (0, 1, 0, 0, 0)^T$  means that  $\mathbf{x}$  is in  $C_2$ .
    - (This works for  $K = 2$  as well)

# Classification problem

- Training: train a classifier  $h(\mathbf{x})$  from training data
  - Training data  $\{ (x^{(1)}, y^{(1)}) , (x^{(2)}, y^{(2)}) , \dots , (x^{(N)}, y^{(N)}) \}$
- Testing (evaluation):
  - testing data:  $h \left( x_{\text{test}}^{(1)} \right) , h \left( x_{\text{test}}^{(2)} \right) , \dots , h \left( x_{\text{test}}^{(N')} \right)$
  - The learning algorithm produces predictions
  - 0-1 loss: Classification error  $= \frac{1}{N'} \sum_{j=1}^{N'} \mathbb{I} \left[ h \left( x_{\text{test}}^{(j)} \right) \neq y_{\text{test}}^{(j)} \right]$

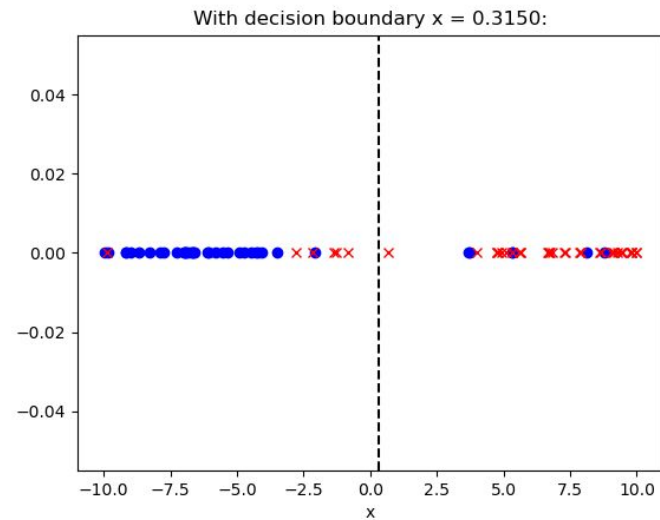
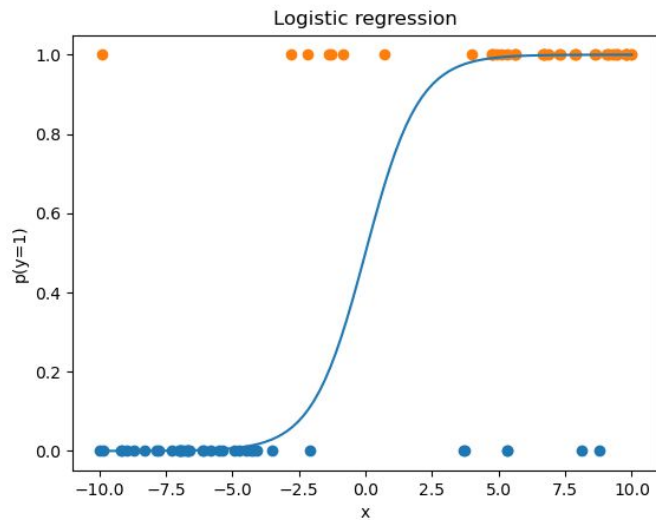
# Logistic regression

# Probabilistic discriminative models

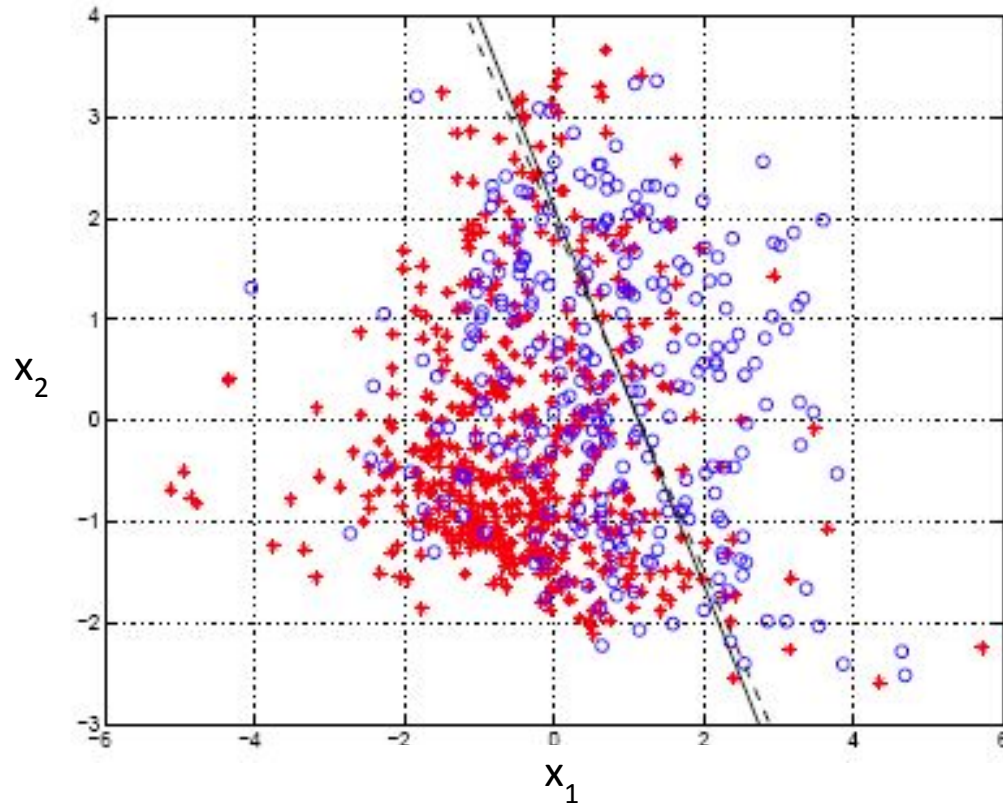
- Model decision boundary as a function of input  $\mathbf{x}$ 
  - Learn  $P(C_k | \mathbf{x})$  over data (e.g., maximum likelihood)
  - Directly predict class labels from inputs
- Next class: we will cover probabilistic generative models
  - Learn  $P(C_k, \mathbf{x})$  over data (maximum likelihood) and then use Bayes' rule to predict  $P(C_k | \mathbf{x})$



# Example (1-dim. case)



# Example (2-dim. case)



# Logistic regression

- Models the class posterior using a sigmoid applied to a linear function of the feature vector:

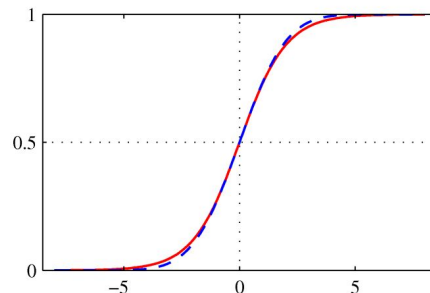
$$p(C_1|\phi) = h(\phi) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}))$$

- We can solve the parameter  $\mathbf{w}$  by maximizing the likelihood of the training data

# Sigmoid and logit functions

- The *logistic sigmoid* function is:

$$\sigma(a) = \frac{1}{1 + \exp(-a)} = \frac{\exp(a)}{1 + \exp(a)}$$



- Its inverse is the *logit* function (aka log odds ratio):

$$a = \ln \left( \frac{\sigma}{1 - \sigma} \right)$$

- Generalizes to *normalized exponential*, or *softmax*

$$p_i = \frac{\exp(q_i)}{\sum_j \exp(q_j)}$$

# Class-conditional probability (for a single example)

- Depending on the label  $y$ , the conditional probability of  $y$  given  $\mathbf{x}$  is defined as:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}))$$

$$P(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^\top \phi(\mathbf{x}))$$

- Therefore we can write both cases compactly as:

$$P(y|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}))^y (1 - \sigma(\mathbf{w}^\top \phi(\mathbf{x})))^{1-y}$$

# Likelihood function (of logistic regression)

- The likelihood of Data  $\{(\phi(\mathbf{x}^{(n)}), y^{(n)})\}$ , where  $y^{(n)} \in \{0, 1\}$

$$P(D|\mathbf{w}) = \prod_{i=1}^N P(\mathbf{x}^{(i)}, y^{(i)}|\mathbf{w}) \quad \text{IID (Independent Identical Distribution)}$$

Definition of  
conditional  
probability



$$= \prod_{i=1}^N P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) \underbrace{P(\mathbf{x}^{(i)}|\mathbf{w})}_{=P(\mathbf{x}^{(i)})}$$

P(x) does not depend on w

$$\propto \prod_{i=1}^N P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) \longrightarrow P(\mathbf{y}|\mathbf{X}, \mathbf{w})$$

Compact notation: Technically speaking, this is  
(conditional) likelihood of y given X

# Logistic regression

- For a data set  $\{(\phi(\mathbf{x}^{(n)}), y^{(n)})\}$ , where  $y^{(n)} \in \{0, 1\}$  the likelihood function is

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^N (h^{(n)})^{y^{(n)}} (1 - h^{(n)})^{1-y^{(n)}}$$

**note:**  $h(\mathbf{x})$  is the hypothesis function,  $\sigma(\mathbf{x})$  is the specific hypothesis for logistic regression

where

$$h^{(n)} = p(C_1|\phi(\mathbf{x}^{(n)})) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}))$$

- Define a loss function  $E(\mathbf{w}) = -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})$ 
  - Minimizing  $E(\mathbf{w})$  maximizes likelihood

# Derivation

- $\log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^N y^{(n)} \log h^{(n)} + (1 - y^{(n)}) \log(1 - h^{(n)})$
- Gradient (matrix calculus)

$$\begin{aligned} & \nabla_{\mathbf{w}} \log P(\mathbf{y}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{w}) \\ &= \sum_{n=1}^N \nabla_{\mathbf{w}} \left( y^{(n)} \log h(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - y^{(n)}) \log(1 - h(\mathbf{x}^{(n)}, \mathbf{w})) \right) \end{aligned}$$



# Derivation


- $\log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^N y^{(n)} \log h^{(n)} + (1 - y^{(n)}) \log(1 - h^{(n)})$
- **Gradient (matrix calculus)**  
$$\nabla_{\mathbf{w}} \log P(\mathbf{y}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{w})$$
$$= \sum_{n=1}^N \nabla_{\mathbf{w}} \left( y^{(n)} \log h(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - y^{(n)}) \log(1 - h(\mathbf{x}^{(n)}, \mathbf{w})) \right)$$

$h(\mathbf{x}^{(n)}, \mathbf{w}) \triangleq \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \triangleq \sigma^{(n)}$

# Derivation

- $\log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^N y^{(n)} \log h^{(n)} + (1 - y^{(n)}) \log(1 - h^{(n)})$
- Gradient (matrix calculus)**  $h(\mathbf{x}^{(n)}, \mathbf{w}) \triangleq \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \triangleq \sigma^{(n)}$

$$\begin{aligned} & \nabla_{\mathbf{w}} \log P(\mathbf{y}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{w}) \\ &= \sum_{n=1}^N \nabla_{\mathbf{w}} \left( y^{(n)} \log h(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - y^{(n)}) \log(1 - h(\mathbf{x}^{(n)}, \mathbf{w})) \right) \\ &= \sum_{n=1}^N \left( y^{(n)} \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{\sigma^{(n)}} - (1 - y^{(n)}) \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{1 - \sigma^{(n)}} \right) \nabla_{\mathbf{w}} (\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \end{aligned}$$


$$\frac{\partial}{\partial s} \sigma(s) = \frac{\partial}{\partial s} \left( \frac{1}{1 + \exp(-s)} \right) = \sigma(s)(1 - \sigma(s))$$

# Derivation

- $\log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^N y^{(n)} \log h^{(n)} + (1 - y^{(n)}) \log(1 - h^{(n)})$
- **Gradient (matrix calculus)**  $h(\mathbf{x}^{(n)}, \mathbf{w}) \triangleq \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \triangleq \sigma^{(n)}$

$$\begin{aligned} & \nabla_{\mathbf{w}} \log P(\mathbf{y}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{w}) \\ &= \sum_{n=1}^N \nabla_{\mathbf{w}} \left( y^{(n)} \log h(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - y^{(n)}) \log(1 - h(\mathbf{x}^{(n)}, \mathbf{w})) \right) \\ &= \sum_{n=1}^N \left( y^{(n)} \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{\sigma^{(n)}} - (1 - y^{(n)}) \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{1 - \sigma^{(n)}} \right) \nabla_{\mathbf{w}} (\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \\ &= \sum_{n=1}^N \left( y^{(n)}(1 - \sigma^{(n)}) - (1 - y^{(n)})\sigma^{(n)} \right) \nabla_{\mathbf{w}} (\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \end{aligned}$$

# Derivation

- $\log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^N y^{(n)} \log h^{(n)} + (1 - y^{(n)}) \log(1 - h^{(n)})$
- **Gradient (matrix calculus)**  $h(\mathbf{x}^{(n)}, \mathbf{w}) \triangleq \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \triangleq \sigma^{(n)}$

$$\begin{aligned} & \nabla_{\mathbf{w}} \log P(\mathbf{y}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{w}) \\ &= \sum_{n=1}^N \nabla_{\mathbf{w}} \left( y^{(n)} \log h(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - y^{(n)}) \log(1 - h(\mathbf{x}^{(n)}, \mathbf{w})) \right) \\ &= \sum_{n=1}^N \left( y^{(n)} \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{\sigma^{(n)}} - (1 - y^{(n)}) \frac{\sigma^{(n)}(1 - \sigma^{(n)})}{1 - \sigma^{(n)}} \right) \nabla_{\mathbf{w}} (\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \\ &= \sum_{n=1}^N \left( y^{(n)}(1 - \sigma^{(n)}) - (1 - y^{(n)})\sigma^{(n)} \right) \nabla_{\mathbf{w}} (\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) \\ &= \sum_{n=1}^N \left( y^{(n)} - \sigma^{(n)} \right) \phi(\mathbf{x}^{(n)}) \end{aligned}$$

# Logistic regression: gradient descent

Note: Vectorized form

- Taking the gradient of  $E(\mathbf{w})$  gives us  $\nabla E(\mathbf{w}) = \Phi^T (\mathbf{h} - \mathbf{y})$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (h^{(n)} - y^{(n)}) \phi(\mathbf{x}^{(n)})$$

- Recall  $h^{(n)} = p(C_1 | \phi(\mathbf{x}^{(n)})) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}))$
- This is essentially the same gradient expression that appeared in linear regression with least-squares.
- Note the error term between model prediction and target value:
  - Logistic regression:  $h^{(n)} - y^{(n)} = \sigma(\mathbf{w}^\top \phi(\mathbf{x}^{(n)})) - y^{(n)}$
  - Cf. Linear regression:  $h^{(n)} - y^{(n)} = \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}$

# Newton's method

- Goal: Minimizing a general function  $E(\mathbf{w})$  (one-dimensional case)

- Approach: solve for

$$f(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$$

- So, how to solve this problem?

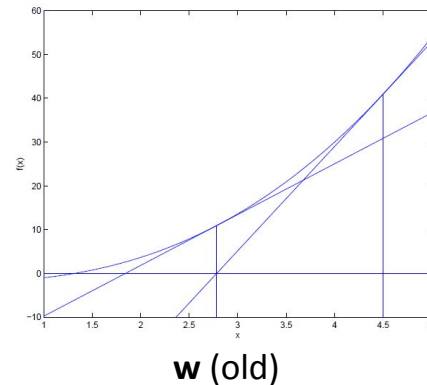
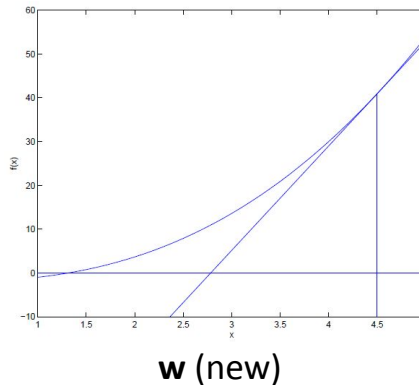
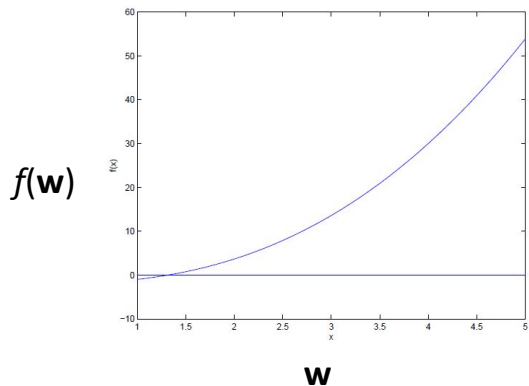
- Newton's method (aka Newton-Raphson method)

- Repeat until convergence:

$$\mathbf{w} := \mathbf{w} - \frac{f(\mathbf{w})}{f'(\mathbf{w})}$$

# Newton's method

- Interactively solve until we get  $f(w) = 0$ .



- Geometric intuition:

$$w := w - \frac{f(w)}{f'(w)}$$

Current value

"Slope"

# Newton's method

- Now we want to minimize  $E(\mathbf{w})$ 
  - Convert  $E'(\mathbf{w}) = f(\mathbf{w})$
  - Repeat until convergence

$$\mathbf{w} := \mathbf{w} - \frac{E'(\mathbf{w})}{E''(\mathbf{w})}$$

Newton update  
when  $w$  is a scalar



# Newton's method

- Now we want to minimize  $E(\mathbf{w})$ 
  - Convert  $E'(\mathbf{w}) = f(\mathbf{w})$
  - Repeat until convergence

$$\mathbf{w} := \mathbf{w} - \frac{E'(\mathbf{w})}{E''(\mathbf{w})}$$

Newton update  
when  $w$  is a scalar

- This method can be extended to the multivariate case:

$$\mathbf{w} := \mathbf{w} - H^{-1} \nabla_{\mathbf{w}} E$$

Newton update  
when  $w$  is a vector

where  $\mathbf{H}$  is a Hessian matrix evaluated at  $\mathbf{w}$

$$H_{ij}(\mathbf{w}) = \frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}_i \partial \mathbf{w}_j}$$

- Note: for linear regression, the Hessian is  $\Phi^T \Phi$

# Derivation of Newton's method

Taylor expansion of  $E(\mathbf{w})$  at  $\mathbf{w}_0$  up to 2nd order:

$$\begin{aligned} E(\mathbf{w}) \approx & E(\mathbf{w}_0) + \nabla E(\mathbf{w}_0)^T (\mathbf{w} - \mathbf{w}_0) \\ & + \frac{1}{2} (\mathbf{w} - \mathbf{w}_0)^T H(\mathbf{w}_0) (\mathbf{w} - \mathbf{w}_0) \end{aligned}$$

Find a closed-form solution that optimizes the quadratic approximation above

$$\nabla \left[ E(\mathbf{w}_0) + \nabla E(\mathbf{w}_0)^T (\mathbf{w} - \mathbf{w}_0) + \frac{1}{2} (\mathbf{w} - \mathbf{w}_0)^T H(\mathbf{w}_0) (\mathbf{w} - \mathbf{w}_0) \right] = \mathbf{0}$$

Using matrix calculus trick similar to Linear regression, we get

$$\mathbf{w} = \mathbf{w}_0 - H(\mathbf{w}_0)^{-1} \nabla E(\mathbf{w}_0)$$

# Iteratively Reweighted Least Squares (IRLS)

- Recall: for linear regression, least-squares has a closed-form solution:

$$\mathbf{w}_{\text{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top y$$

- This generalizes to weighted-least-squares with an  $N \times N$  diagonal weight matrix  $\mathbf{R}$ .

$$\mathbf{w}_{\text{WLS}} = (\Phi^\top \mathbf{R} \Phi)^{-1} \Phi^\top \mathbf{R} y$$

- For logistic regression, however,  $h(\mathbf{x}, \mathbf{w})$  is non-linear, and there is no closed-form solution. So **we need to iterate (i.e. repeatedly apply Newton steps) to get the optimal solution, which is called IRLS.**

# Iterative solution

- Apply Newton-Raphson method to iterate to a solution  $\mathbf{w}$  for  $\nabla E(\mathbf{w}) = 0$
- This involves least-squares with weights  $\mathbf{R}$ :

$$R_{nn} = h^{(n)}(1 - h^{(n)})$$

- Since  $\mathbf{R}$  depends on  $\mathbf{w}$  (and vice versa), we get *iterative reweighted least squares* (IRLS)

where  $\mathbf{w}^{(\text{new})} = (\Phi^\top \mathbf{R} \Phi)^{-1} \Phi^\top \mathbf{R} \mathbf{z}$

$$\mathbf{z} = \Phi \mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{h} - \mathbf{y})$$

# Iterative solution: Derivation

Applying Newton's method:

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - H(\mathbf{w}^{(\text{old})})^{-1} \nabla E(\mathbf{w}^{(\text{old})})$$

Gradient and Hessians from Logistic Regression loss function:

$$\nabla E(\mathbf{w}) = \Phi^T (\mathbf{h} - \mathbf{y}) \quad \text{and} \quad H(\mathbf{w}) = \nabla^2 E(\mathbf{w}) = \Phi^T R \Phi$$

where  $R_{nn} = h^{(n)} (1 - h^{(n)})$

Putting together:  $\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - [\Phi^T R \Phi]^{-1} \Phi^T (\mathbf{h} - \mathbf{y})$

If we define  $\mathbf{z}$  as  $\mathbf{z} = \Phi \mathbf{w}^{(\text{old})} - R^{-1} (\mathbf{h} - \mathbf{y})$

We can also write the solution as:  $\mathbf{w}^{(\text{new})} = (\Phi^T R \Phi)^{-1} \Phi^T R \mathbf{z}$

# K-nearest neighbor classification

# K-nearest neighbors

- Training method:
  - Save the training examples (no sophisticated learning)
- At prediction (testing) time:
  - Given a test (query) example  $\mathbf{x}$ , find the  $K$  training examples that are *closest* to  $\mathbf{x}$ .

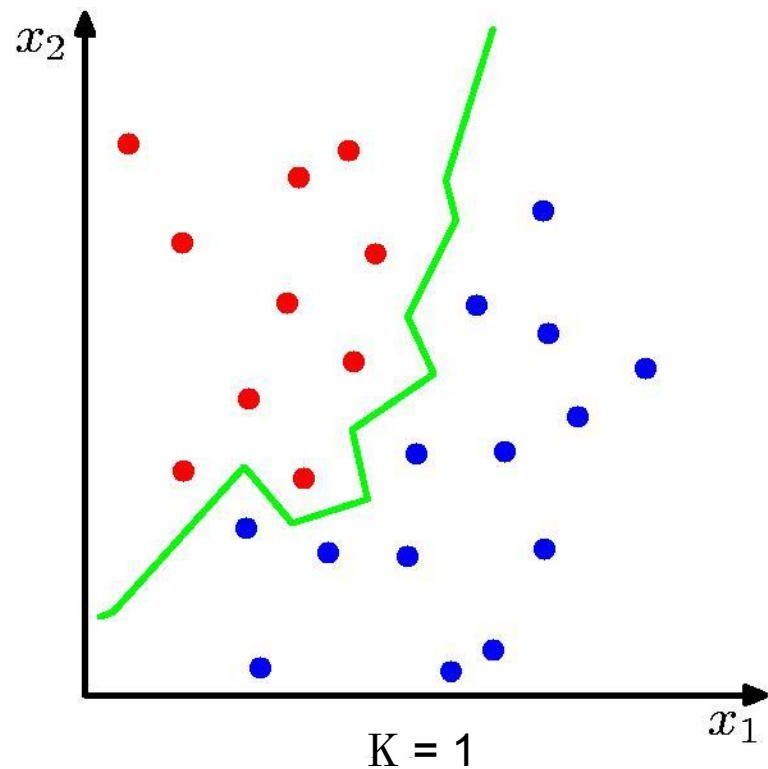
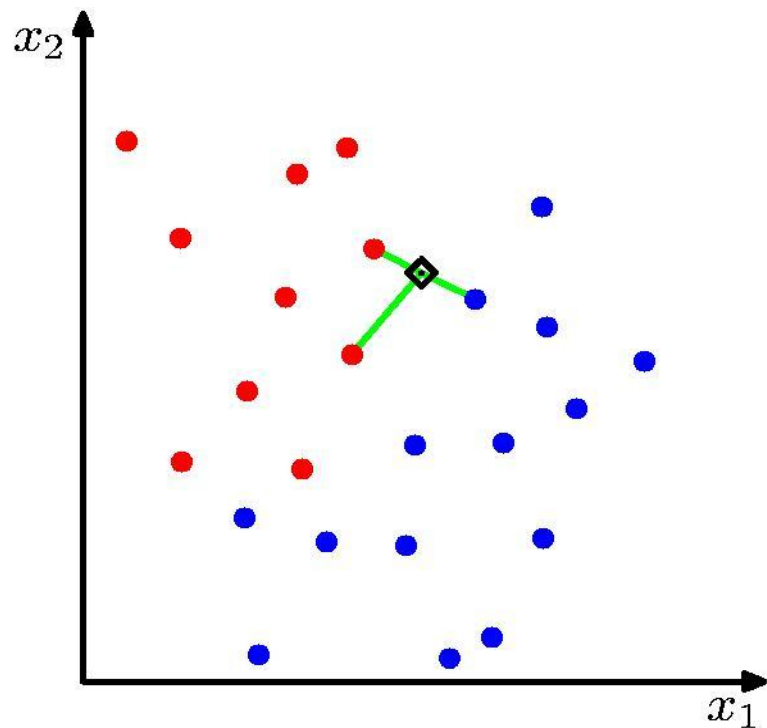
$$\text{KNN}(\mathbf{x}) = \{ (\mathbf{x}^{(1)'}, y^{(1)'}) , (\mathbf{x}^{(2)'}, y^{(2)'}) , ..., (\mathbf{x}^{(K)'}, y^{(K)'}) \}$$

- Predict the most frequent class among all  $y$ 's from  $\text{KNN}(\mathbf{x})$ .

$$h(\mathbf{x}) = \arg \max_y \sum_{(\mathbf{x}', y') \in \text{KNN}(\mathbf{x})} \mathbb{I}[y' = y] \quad \text{“majority vote”}$$

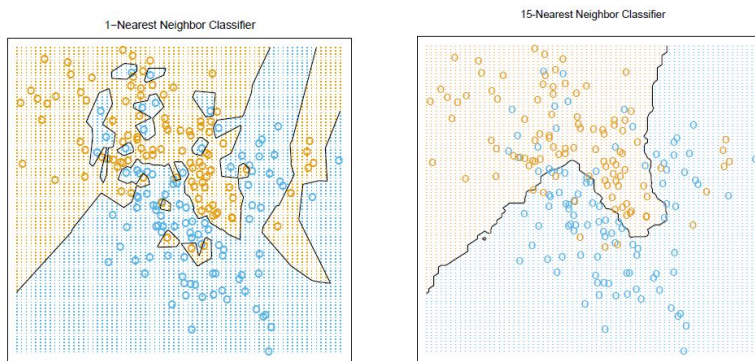
- Note: this function can be applied to regression!

# K-nearest neighbors for classification





# K-nearest neighbors for classification



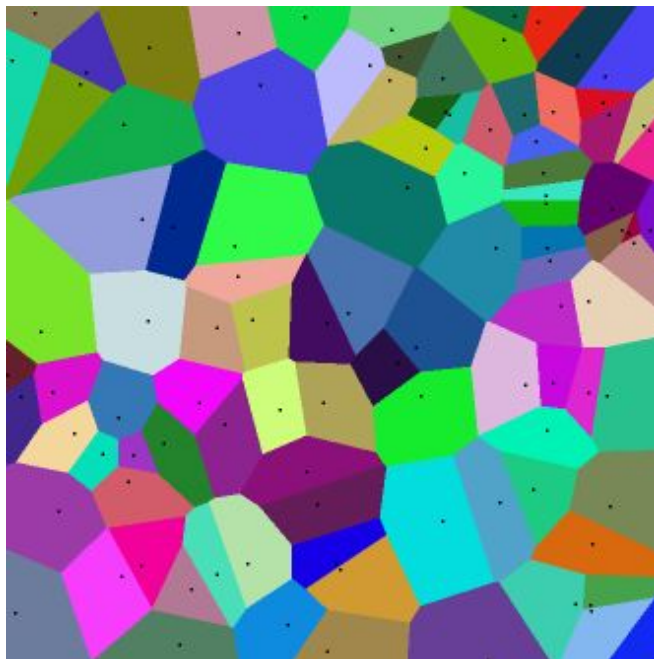
- Larger  $K$  leads to a smoother decision boundary (bias-variance trade-off)
- Classification performance generally improves as  $N$  (training set size) increases
- For  $N \Rightarrow \infty$ , the error rate of the 1-nearest-neighbor classifier is never more than twice the optimal error (obtained from the true conditional class distributions). See ESL CH 13.3.

# Factors (hyperparameters) affecting KNN

- Distance metric  $D(\mathbf{x}, \mathbf{x}')$ 
  - How to define distance between two examples  $\mathbf{x}$  and  $\mathbf{x}'$ ?
- The value of  $K$ 
  - $K$  determines how much we “smooth out” the prediction

# What is the decision boundary?

Voronoi diagram: Euclidean ( $L_2$ ) distance

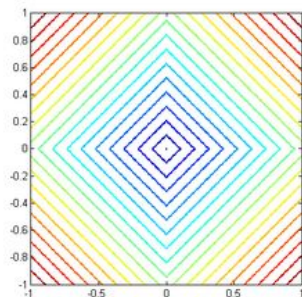


Note: Each region corresponds kNN's prediction when  $K=1$

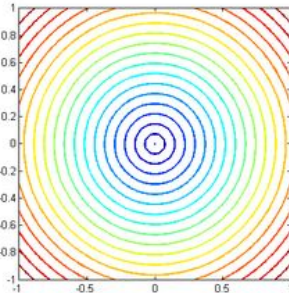
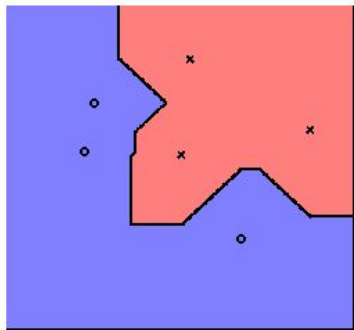
i.e. prediction is the same as the corresponding training sample's label in each region (training sample is visualized as dot).

# Dependence on distance metric ( $L^q$ norm)

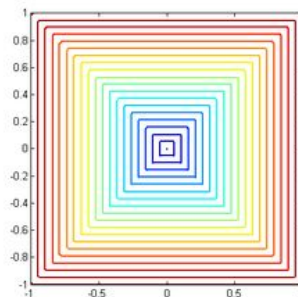
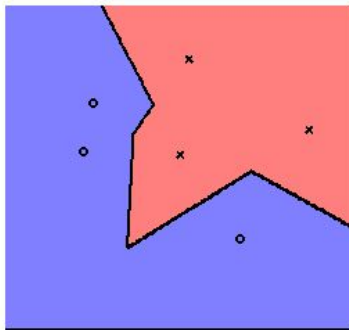
Distance between i-th and j-th example:  $\sqrt[q]{\sum_l (x_l^{(i)} - x_l^{(j)})^q}$



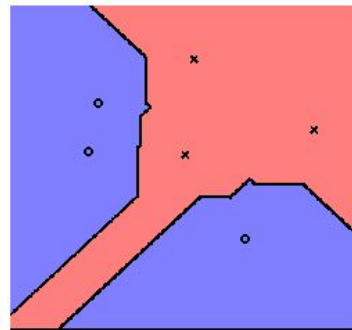
knn (K=1): l1 Distance



knn (K=1): l2 Distance



knn (K=1): linf Distance



# KNN: classification vs regression

- We can formulate KNN into regression/classification
- For classification, where the label  $y$  is categorical, we take the “majority vote” over target labels.

$$h(\mathbf{x}) = \arg \max_y \sum_{(\mathbf{x}', y') \in \text{KNN}(\mathbf{x})} \mathbb{I}[y' = y]$$

- For regression, where the label  $y$  is real-valued numbers, we take “average” over target labels.

$$h(\mathbf{x}) = \frac{1}{k} \sum_{(\mathbf{x}', y') \in \text{KNN}(\mathbf{x})} y'$$

# Advantage/disadvantages of KNN methods

- Advantage:
  - Very simple and flexible (no assumption on distribution)
  - Effective (e.g. for low dimensional inputs)
- Disadvantages:
  - Expensive: need to remember (store) and search through all the training data for every prediction
  - Curse of dimensionality: in high dimensions, all points are far
  - Not robust to irrelevant features: if  $\mathbf{x}$  has irrelevant/noisy features, then distance function does not reflect similarity between examples

# Concept check

- How are labels represented in multiclass classification problems?
- What is the motivation for using Newton's method for optimization in logistic regression?
- What does increasing  $K$  do for the results from KNN?

# Any feedback (about lecture, slide, homework, project, etc.)?

(via anonymous google form: <https://forms.gle/fpYmiBtG9Me5qbP37>)



Change Log of lecture slides:

<https://docs.google.com/document/d/e/2PACX-1vSSIHjkllypK7rKFSR1-5GYXyBCEW8UPtpSfCR9AR6M1I7K9ZQEmxfFwaWaW7kLDxusthsF8WICyZJ-/pub>