EECS 545: Machine Learning
Lecture 4. Classification

Honglak Lee

1/24/2024

UNIVERSITY OF MICHIGAN

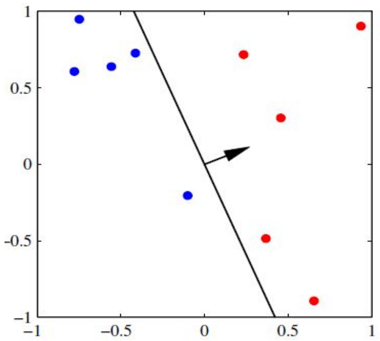
Outline

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

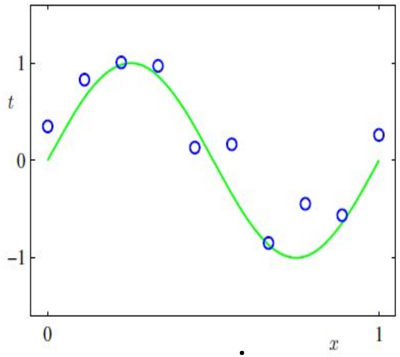
Supervised learning: classification

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 <u>(today's topic)</u>
  - Continuous-valued labels: regression



classification



regression

Classification problem

- The task of classification:
  - Given an input vector **x**, assign it to one of K distinct classes C_k where k = 1, … K
- Representing the assignment:
  - For $K = 2$:
    - y = 1 means that **x** is in C_1
    - y = 0 means that **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 **x** is in C_2.
    - (This works for $K = 2$ as well)

Classification problem

- Training: train a classifier *h*(**x**) from training data
  - Training data $\left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ \ldots \ , \left(x^{(N)}, y^{(N)}\right)\right\}$

- Testing (evaluation):
  - testing data: $h\left(x_{\text{test}}^{(1)}\right), h\left(x_{\text{test}}^{(2)}\right), \ldots, h\left(x_{\text{test}}^{(N')}\right)$
  - The learning algorithm produces predictions

  - 0-1 loss: $\text{Classification error} = \frac{1}{N'} \sum_{j=1}^{N'} \mathbb{I}\left[h\left(x_{\text{test}}^{(j)}\right) \neq y_{\text{test}}^{(j)}\right]$

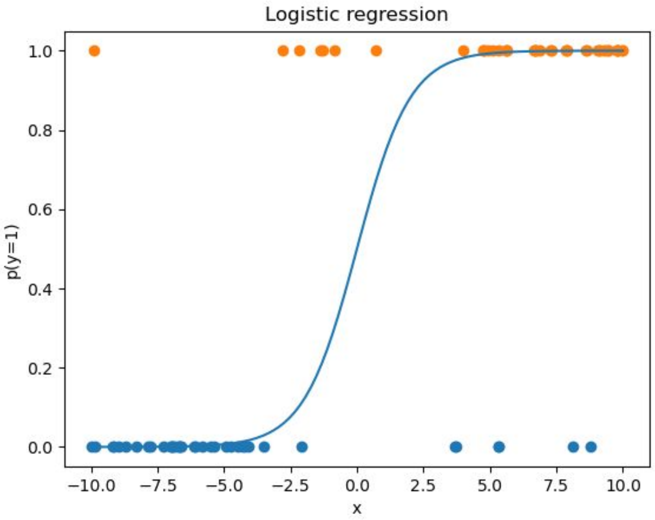
Logistic regression

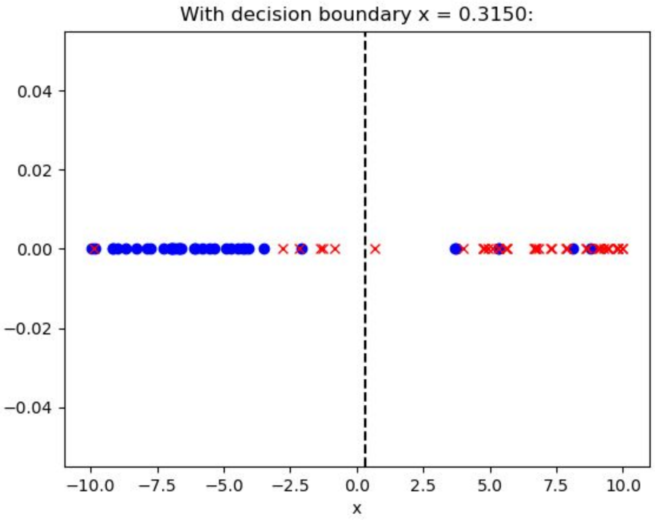
Probabilistic discriminative models

- 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

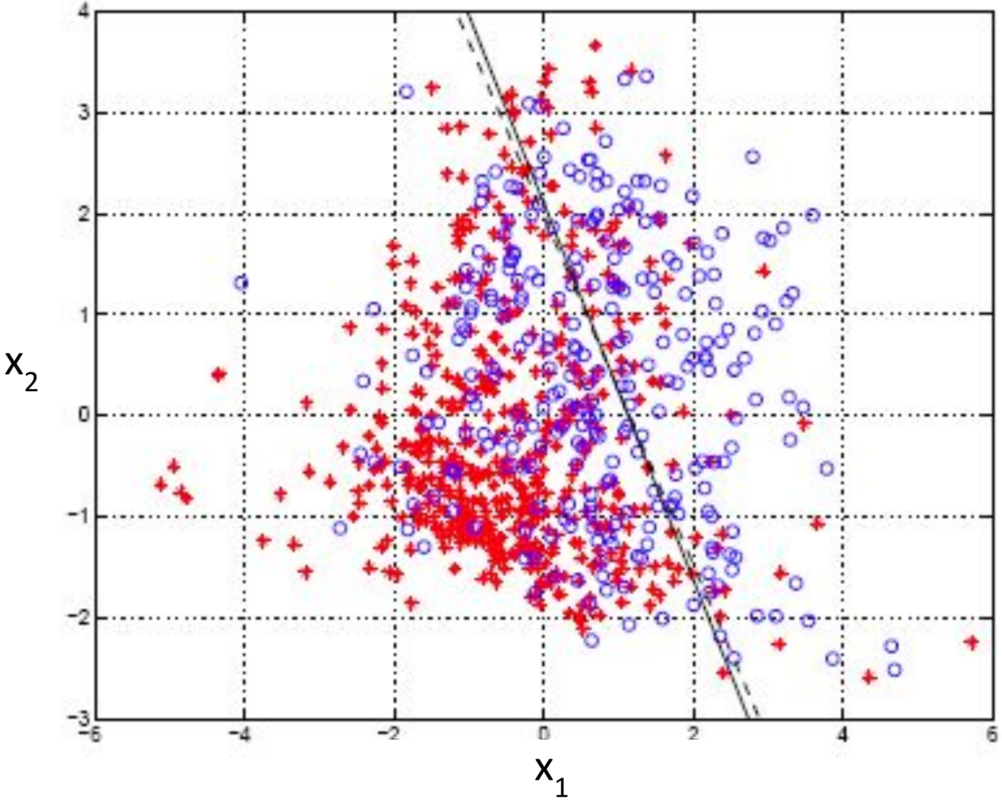
- Next class: we will cover probabilistic generative models
  - Learn $P(C_k, \mathbf{x})$ over data (maximum likelihood) and then use Bayes' rule to predict $P(C_k|\mathbf{x})$

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:

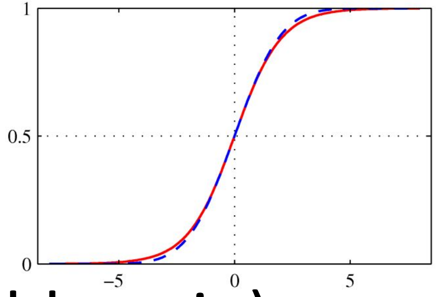
$$p(C_1|\phi) = h(\phi) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}))$$

- We can solve the parameter **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 **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 $y^{(n)} \in \{0, 1\}$

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

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 $\{(\phi(\mathbf{x}^{(n)}), y^{(n)})\}, \text{where } y^{(n)} \in \{0,1\}$ 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)

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

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)

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

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

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)

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

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

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

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

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

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)

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

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

Logistic regression: gradient descent

- Taking the gradient of E(w) gives us

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

- 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

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

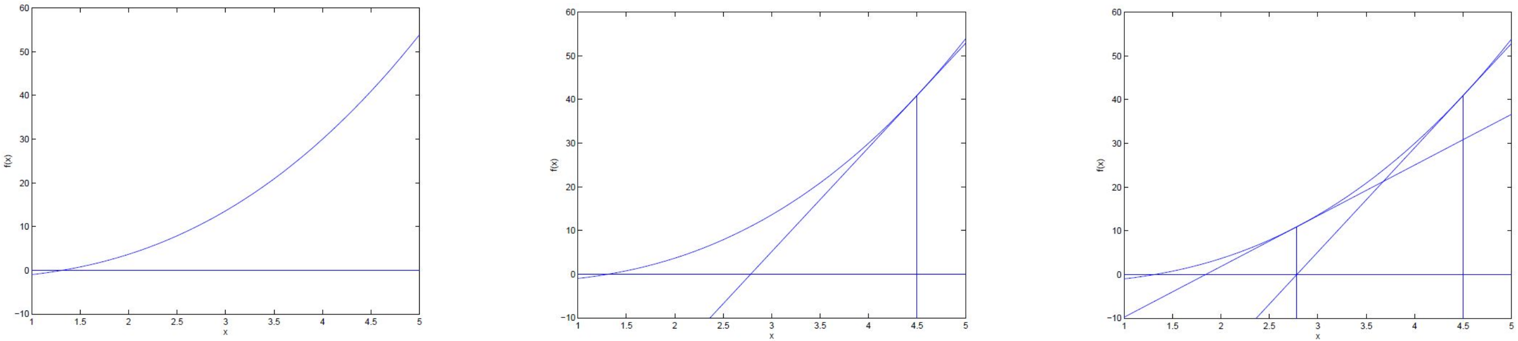
  - So, how to solve this problem?
- Newton's method (aka Newton-Raphson method)
  - Repeat until convergence:

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

Newton’s method

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

$f(\mathbf{w})$



w   **w** (new)   **w** (old)

- Geometric intuition:

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

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 **H** is a Hessian matrix evaluated at **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 $\Phi^\top \Phi$

Logistic regression

- Recall: for linear regression, least-squares has a closed-form solution: $\mathbf{w}_{\mathrm{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top y$

- This generalizes to weighted-least-squares with an NxN diagonal weight matrix **R**.

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

- For logistic regression, however, $h(\mathbf{x}, \mathbf{w})$ is non-linear, and there is no closed-form solution. Must iterate (i.e. repeatedly apply Newton steps).

Iterative solution

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

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

- Since **R** depends on **w** (and vice versa), we get *iterative reweighted least squares* (IRLS) where $\mathbf{w}^{(\mathrm{new})} = (\Phi^\top \mathbf{R} \Phi)^{-1} \Phi^\top \mathbf{R} \mathbf{z}$

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

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 **x**, find the *K* training examples that are *closest* to **x**.

$$\mathrm{KNN}(\mathbf{x}) = \left\{ \left(\mathbf{x}^{(1)\prime}, y^{(1)\prime}\right), \left(\mathbf{x}^{(2)\prime}, y^{(2)\prime}\right), \ldots, \left(\mathbf{x}^{(K)\prime}, y^{(K)\prime}\right) \right\}$$

- Predict the most frequent class among all *y'*s from *KNN*(**x**).

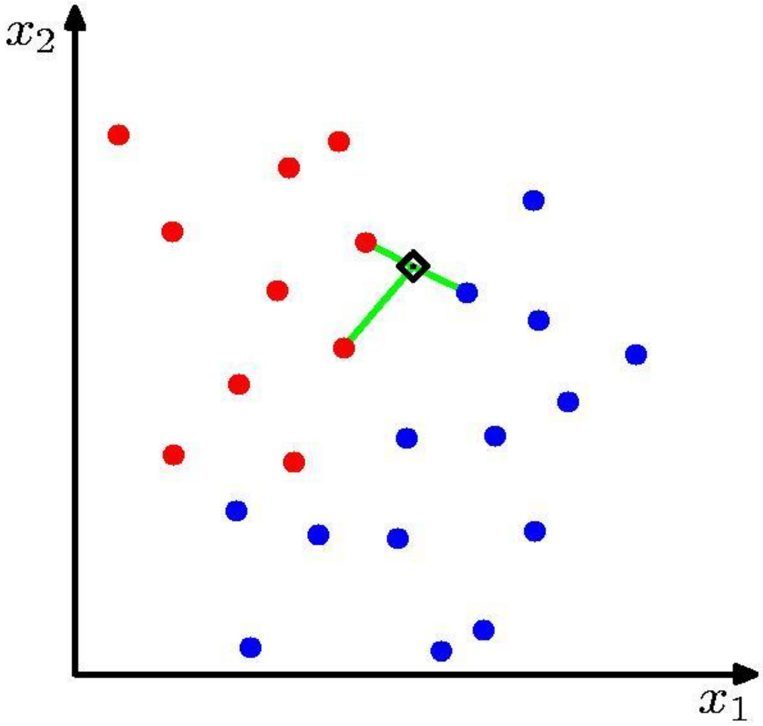
$$h(\mathbf{x}) = \arg\max_{y} \sum_{(\mathbf{x}', y') \in \mathrm{kNN}(\mathbf{x})} \mathbb{I}[y' = y]$$

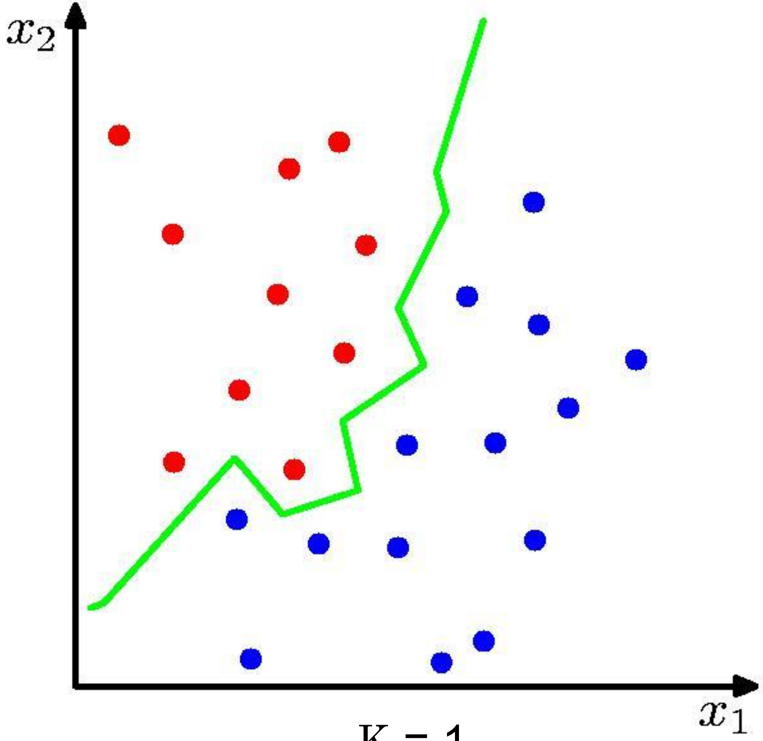
"majority vote"

- <u>Note: this function can be applied to regression!</u>

Slide credit: William Cohen

K-nearest neighbors for classification

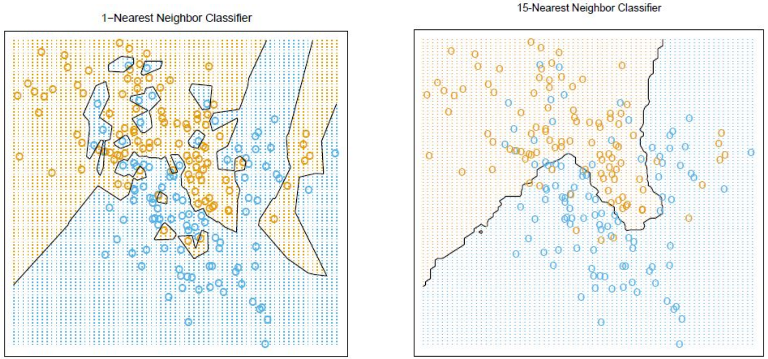




K = 1

Slide credit: Ben Kuipers

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.

Slide credit: Ben Kuipers

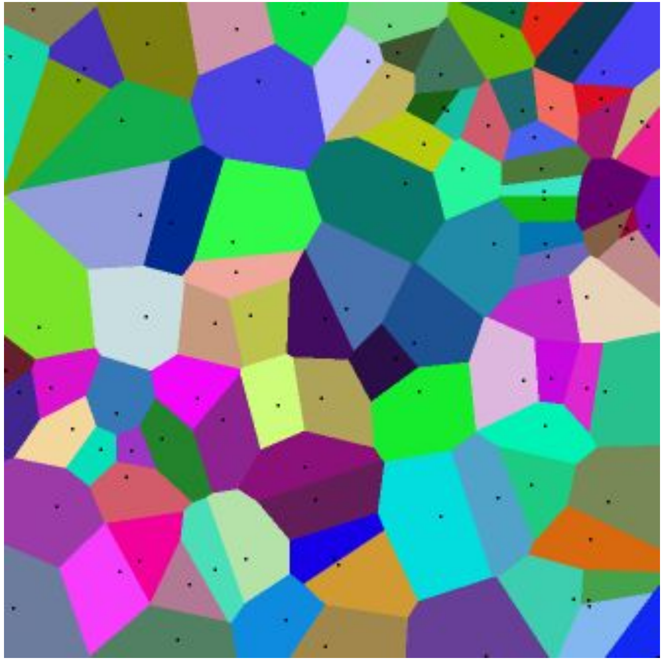
Factors (hyperparameters) affecting KNN

- Distance metric $D(\mathbf{x}, \mathbf{x'})$
  - How to define distance between two examples **x** and **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



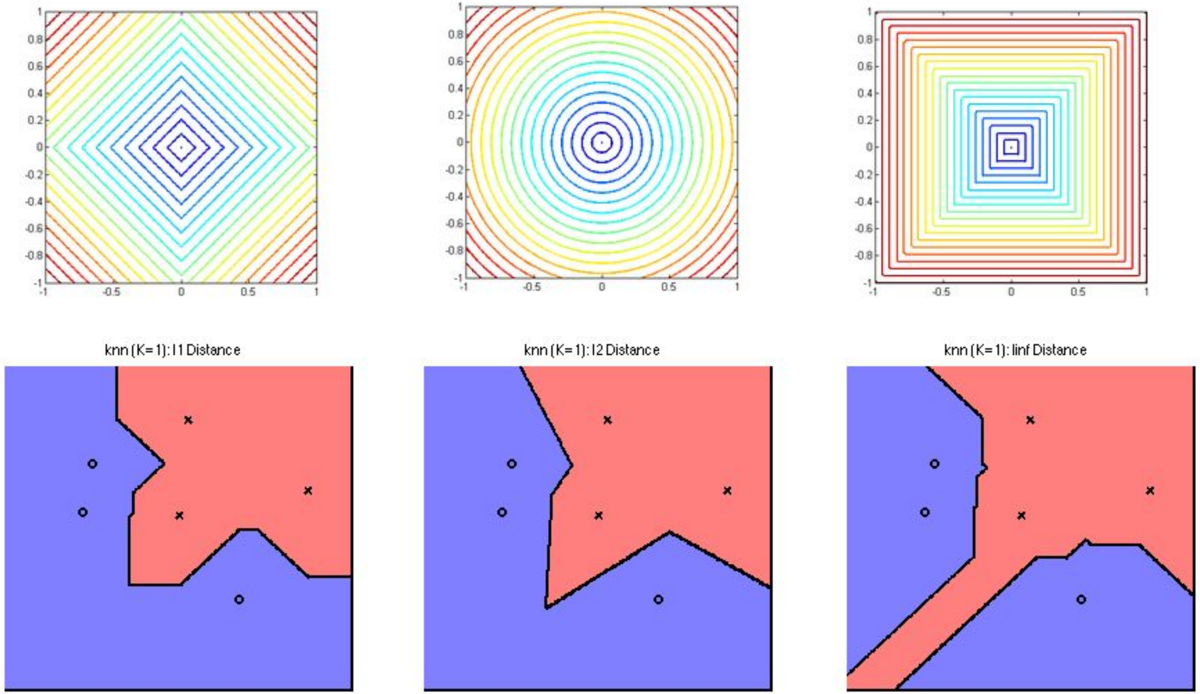
Note: Each region corresponds kNN's prediction when K=1

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

Slide credit: William Cohen

Dependence on distance metric (L^q norm)

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



Slide credit: Ben Taskar

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.

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

- For regression, where the label *y* is real-valued numbers, we take "average" over target labels.

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

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 **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/99jeftYTaozJvCEF8)

Change Log of lecture slides:

https://docs.google.com/document/d/e/2PACX-1vRKx40eOJKACqrKWraio0AmlFS1_xBMINuWcc-jzpfo-ySj_gBuqTVdfHy8v4HDmqDJ3b3TvAW1FVuH/pub