CSCI567 Machine Learning (Spring 2018)

Michael Shindler

Lecture on January 10 2018

Outline

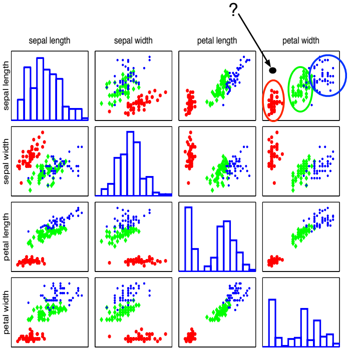
1. Review of Last Lecture
2. Some practical sides of NNC
3. What we have learned
4. More deep understanding about NNC
5. Summary

Outline

1. Review of Last Lecture
   - Overview of Machine Learning
   - Nearest Neighbor Classifiers
2. Some practical sides of NNC
3. What we have learned
4. More deep understanding about NNC
5. Summary

Machine Learning is about identifying patterns and making predictions

Closer to red cluster: so labeling it as setosa



Multi-class classification

Classify data into one of the multiple categories

- Input (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Output (label): $y \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$
- Learning goal: $y = f(\boldsymbol{x})$

Special case: binary classification

- Number of classes: $\mathsf{C} = 2$
- Labels: $\{0, 1\}$ or $\{-1, +1\}$

More terminology

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- They are used for learning $f(\cdot)$

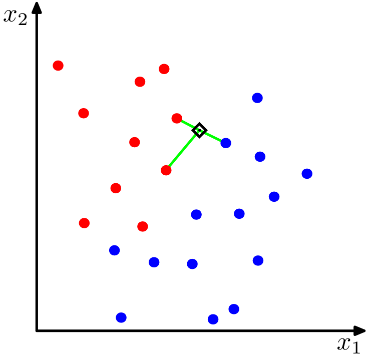
Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Training data and test data should *not* overlap: $\mathcal{D}^{\text{TRAIN}} \cap \mathcal{D}^{\text{TEST}} = \emptyset$

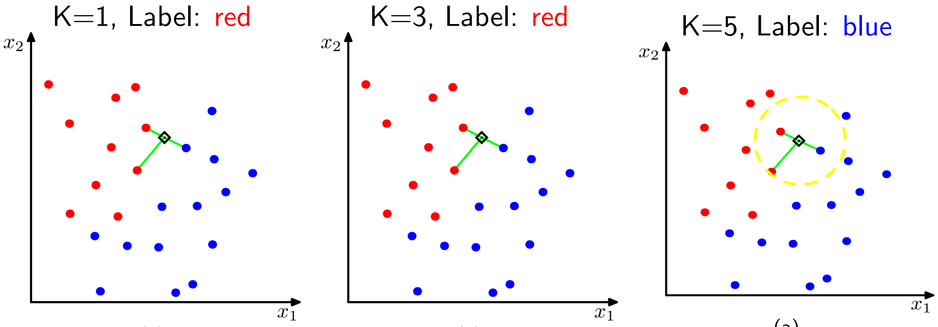
Example of Nearest Neighbor Classification

In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance, thus, $\boldsymbol{x}$ will be labeled as red.



(a)

Example of K-Nearest Neighbor Classification



K=1, Label: red

(a)

K=3, Label: red

(a)

K=5, Label: blue

(a)

K-nearest neighbor (KNN) classification

Algorithm

- 1-nearest neighbor: $\mathsf{nn}_1(\boldsymbol{x}) = \arg\min_{n\in[\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 2nd-nearest neighbor: $\mathsf{nn}_2(\boldsymbol{x}) = \arg\min_{n\in[\mathsf{N}]-\mathsf{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 3rd-nearest neighbor: $\mathsf{nn}_2(\boldsymbol{x}) = \arg\min_{n\in[\mathsf{N}]-\mathsf{nn}_1(\boldsymbol{x})-\mathsf{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$

The set of K-nearest neighbor

$$\mathsf{knn}(\boldsymbol{x}) = \{\mathsf{nn}_1(\boldsymbol{x}), \mathsf{nn}_2(\boldsymbol{x}), \cdots, \mathsf{nn}_K(\boldsymbol{x})\}$$

Classification rule

- Aggregate every nearest neighbor's vote to a class label c

$$v_c = \sum_{n\in\mathsf{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\mathsf{C}]$$

- Label with the majority

$$y = f(\boldsymbol{x}) = \arg\max_{c\in[\mathsf{C}]} v_c$$

Outline

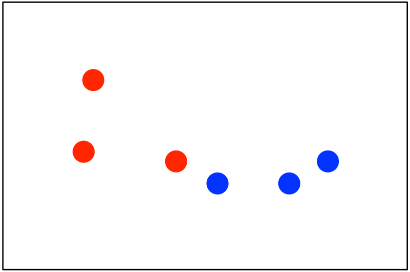
1. Review of Last Lecture
2. Some practical sides of NNC
   - How to measure performance of a classifier?
   - How to tune to get the best out of it?
   - Preprocessing data
3. What we have learned
4. More deep understanding about NNC
5. Summary

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

Training data



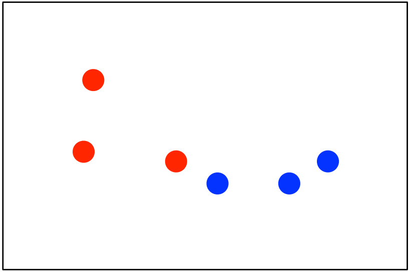
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

Training data



What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 66.67\%(\text{i.e.}, 4/6)$$
$$\varepsilon^{\text{TRAIN}} = 33.33\%(\text{i.e.}, 2/6)$$

Hypeparameters in NNC

Two practical issues about NNC

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance), for example, from the following generalized distance measure

$$\|\boldsymbol{x} - \boldsymbol{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$

for $p \geq 1$.

Those are not specified by the algorithm itself — resolving them requires empirical studies and are task/dataset-specific.

Tuning by using a validation dataset

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Development (or validation) data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{L}}, y_{\mathsf{L}})\}$
- They are used to optimize hyperparameter(s).

Training data, validation and test data should *not* overlap!

Recipe

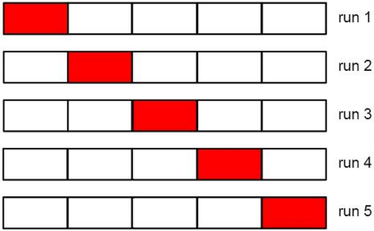
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{DEV}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{DEV}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Cross-validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that *on average*, the model performing the best

$S = 5$: 5-fold cross validation



Special case: when $S = N$, this will be leave-one-out.

Recipe

- Split the training data into S equal parts. Denote each part as $\mathcal{D}_s^{\mathrm{TRAIN}}$
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
    - for every $s \in [1, \mathsf{S}]$
        - Train a model using $\mathcal{D}_{\backslash s}^{\mathrm{TRAIN}} = \mathcal{D}^{\mathrm{TRAIN}} - \mathcal{D}_s^{\mathrm{TRAIN}}$
        - Evaluate the performance of the model on $\mathcal{D}_s^{\mathrm{TRAIN}}$
    - Average the S performance metrics
- Choose the hyperparameter corresponding to the best averaged performance
- Use the best hyerparamter to train on a model using all $\mathcal{D}^{\mathrm{TRAIN}}$
- Evaluate the model on $\mathcal{D}^{\mathrm{TEST}}$

Preprocess data

Normalize data so that the data look like from a normal distribution

- Compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{\mathsf{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Many other ways of normalizing data — you would need/want to try different ones and pick them using (cross)validation

Mini-summary

Advantages of NNC

- Computationally, simple and easy to implement – just computing the distance
- Theoretically, has strong guarantees "doing the right thing"

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a data point
- We need to "carry" the training data around. Without it, we cannot do classification. This type of method is called *nonparametric*.
- Choosing the right distance measure and K can be involved.

Outline

1. Review of Last Lecture
2. Some practical sides of NNC
3. What we have learned
4. More deep understanding about NNC
5. Summary

Summary so far

- Described a simple learning algorithm called Nearest Neighbor Classification
    - Used intensively in practical applications — you will get a taste of it in your homework
    - Discussed a few practical aspects, such as tuning hyperparameters, with (cross)validation

Outline

1. Review of Last Lecture
2. Some practical sides of NNC
3. What we have learned
4. More deep understanding about NNC
   - Step 1: Expected risk
   - A small review on necessary probability concepts
   - Step 2: The ideal classifier
   - Step 3: Comparing NNC to the ideal classifier
5. Summary

Is NNC too simple to do the right thing?

To answer this question, we proceed in 3 steps

1. We define *more carefully* a performance metric for a classifier/algorithm .
2. We hypothesize an ideal classifier - *the best possible one there*.
3. We then compare our simple NNC classifier to the ideal one and show that it performs *nearly as good*.

Drawback of the metrics we have talked about so far

They are dataset-specific!

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\mathrm{TEST}}_{\mathcal{D}_1}, A^{\mathrm{TEST}}_{\mathcal{D}_2}, \cdots, A^{\mathrm{TEST}}_{\mathcal{D}_q}, \cdots$$

Drawback of the metrics we have talked about so far

They are dataset-specific!

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\text{TEST}}_{\mathcal{D}_1}, A^{\text{TEST}}_{\mathcal{D}_2}, \cdots, A^{\text{TEST}}_{\mathcal{D}_q}, \cdots$$

These are called *"empirical" accuracies (or errors)*.

Drawback of the metrics we have talked about so far

They are dataset-specific!

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\text{TEST}}_{\mathcal{D}_1}, A^{\text{TEST}}_{\mathcal{D}_2}, \cdots, A^{\text{TEST}}_{\mathcal{D}_q}, \cdots$$

These are called *"empirical" accuracies (or errors)*.

Can we understand the algorithm itself in a "more certain" nature, by removing the uncertainty caused by the datasets?

This will allow us to compare *algorithms themselves*.

Probability: basic definitions

Sample Space: a set of all possible outcomes or realizations of some random trial.
Example: Toss a coin twice; the sample space is $\Omega = \{HH, HT, TH, TT\}$.

Event: A subset of sample space
Example: the event that at least one toss is a head is $A = \{HH, HT, TH\}$.

Probability: We assign a real number $P(A)$ to each event A, called the probability of A. For example,

$$P(A) = \frac{3}{4}$$

Random Variables

Definition: A random variable is a function that maps from a random event to a real number, i.e. $X : \Omega \to R$, that assigns a real number $X(\omega)$ to each outcome ω.

Example: In coin tossing, we let $H \to 1$ and let $T \to 0$.

Random Variables

Definition: A random variable is a function that maps from a random event to a real number, i.e. $X : \Omega \to R$, that assigns a real number $X(\omega)$ to each outcome ω.

Example: In coin tossing, we let $H \to 1$ and let $T \to 0$.

The event "at least one toss is a head" then can be shortened as $X_1 + X_2 > 0$, where X_1 and X_2 are the random variables (ie, 1, or 0 corresponding to the first toss and the second toss respectively).

Random Variables

Definition: A random variable is a function that maps from a random event to a real number, i.e. $X : \Omega \to R$, that assigns a real number $X(\omega)$ to each outcome ω.

Example: In coin tossing, we let $H \to 1$ and let $T \to 0$.

The event "at least one toss is a head" then can be shortened as $X_1 + X_2 > 0$, where X_1 and X_2 are the random variables (ie, 1, or 0 corresponding to the first toss and the second toss respectively).

Data The data are specific realizations of random variables.

$$(X_1 = 1, X_2 = 0), (X_1 = 1, X_2 = 1), (X_1 = 0, X_2 = 0)$$

are 3 observations from the coin toss experiments (note that each experiment involves tossing twice).

Important characterization of random variables

Probability mass function

$$P(X = x) : \text{probability of } X \text{ takes the value of } x$$

For example, a fair coin $P(X = 1) = 1/2$, where X is either 0 ('T') or 1 ('H').

Important characterization of random variables

Probability mass function

$$P(X = x) : \text{probability of } X \text{ takes the value of } x$$

For example, a fair coin $P(X = 1) = 1/2$, where X is either 0 ('T') or 1 ('H').

Expected value/Mean

$$\mu = \mathbb{E}_P X = \sum_{x \in \mathcal{X}} x P(X = x)$$

For example, the μ for tossing a coin is

$$\mu = 1 \times P(X = 1) + 0 \times P(X = 0) = 1/2$$

Important characterization of random variables

Variances

$$\nu = \mathbb{E}_P(X-\mu)^2 = \sum_{x \in \mathcal{X}} (x-\mu)^2 P(X=x)$$

For example, the variance for tossing a coin is

$$\nu = (1-1/2)^2 P(X=1) + (0-1/2)^2 P(X=0) = \frac{1}{4}$$

Important characterization of random variables

Variances

$$\nu = \mathbb{E}_P(X-\mu)^2 = \sum_{x\in\mathcal{X}} (x-\mu)^2 P(X=x)$$

For example, the variance for tossing a coin is

$$\nu = (1-1/2)^2 P(X=1) + (0-1/2)^2 P(X=0) = \frac{1}{4}$$

All those can be extended to continuous random variables – more on this as the semester progresses.

Multivariate Distributions

Dealing with two random variables

$$P(X = x, Y = y) : \text{probability of } X \text{ taking } x \textit{ and } Y \text{ taking } y$$

Example. Let X represent 'height' and Y represent 'male' or 'female'

$$P(X = 6 \text{ ft 2in}, Y =' male')$$

probability of finding a person with height of 6 feet 2 inches and is male

Multivariate Distributions

Dealing with two random variables

$$P(X = x, Y = y) : \text{probability of } X \text{ taking } x \textit{ and } Y \text{ taking } y$$

Example. Let X represent 'height' and Y represent 'male' or 'female'

$$P(X = 6 \text{ ft } 2\text{in}, Y =' male')$$

probability of finding a person with height of 6 feet 2 inches and is male

Marginal distribution

$$P(X = x) = \sum_y P(X = x, Y = y), P(Y = y) = \sum_x P(X = x, Y = y)$$

represent the probability of finding a person who is x tall, or the probability of a finding a person whose sex is y.

Multivariate Distributions

Conditional distribution

$$P(X = x|Y = y)$$

represents that among the all the people whose sex is y, what is the probability of finding that person with a height of x?

$$P(Y = y|X = x)$$

represents that among the all the people whose height is x, what is the probability of finding that person whose sex is y?

Multivariate Distributions

Important relation, ie, Bayes theorem

$$P(Y=y|X=x)=\frac{P(X=x,Y=y)}{P(X=x)}=\frac{P(X=x|Y=y)P(Y=y)}{P(X=x)}$$

No need to stress!

Simple Toy Example

Height	Sex	# of people
6’	male	20
6’	female	10
5’ 4”	male	5
5’ 4”	female	10

No need to stress!

Simple Toy Example

Height	Sex	# of people
6'	male	20
6'	female	10
5' 4"	male	5
5' 4"	female	10

Jointly

$$P(X = 6', Y = male) = \frac{20}{20 + 10 + 5 + 10} = \frac{20}{45} = \frac{4}{9}$$

No need to stress!

Simple Toy Example

Height	Sex	# of people
6'	male	20
6'	female	10
5' 4"	male	5
5' 4"	female	10

Jointly

$$P(X = 6', Y = male) = \frac{20}{20 + 10 + 5 + 10} = \frac{20}{45} = \frac{4}{9}$$

Marginally

$$P(X = 6') = \frac{30}{45} = \frac{2}{3}, P(Y = female) = \frac{20}{45} = \frac{4}{9}$$

No need to stress!

Simple Toy Example

Height	Sex	# of people
6'	male	20
6'	female	10
5' 4"	male	5
5' 4"	female	10

Jointly

$$P(X = 6', Y = male) = \frac{20}{20 + 10 + 5 + 10} = \frac{20}{45} = \frac{4}{9}$$

Marginally

$$P(X = 6') = \frac{30}{45} = \frac{2}{3}, P(Y = female) = \frac{20}{45} = \frac{4}{9}$$

Conditionally

$$P(Y = male|X = 6') = \frac{20}{10 + 20} = \frac{2}{3} = \frac{\frac{4}{9}}{\frac{2}{3}}$$

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y, with $f(x)$ being the classifier,

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y, with $f(x)$ being the classifier,

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

- Expected classification mistake on a single data point $\boldsymbol{x}$

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y, with $f(x)$ being the classifier,

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

- Expected classification mistake on a single data point $\boldsymbol{x}$

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

- The average classification mistake by the classifier itself

$$R(f) = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})} R(f, \boldsymbol{x}) = \mathbb{E}_{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)} L(f(\boldsymbol{x}), y)$$

Jargons

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.

Jargons

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.
- Expected risk

$$R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)} L(f(\boldsymbol{x}), y)$$

Jargons

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.
- Expected risk

$$R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)} L(f(\boldsymbol{x}), y)$$

- Empirical risk

$$R_{\mathcal{D}}(f) = \frac{1}{\mathsf{N}} \sum_{n} L(f(\boldsymbol{x}_n), y_n)$$

Obviously, this is our empirical error (rates).
We can show that this empirical risk is close to the expected risk if we have a lot of (test) data. So we can concentrate on comparing $R(f)$!

Bayes optimal classifier

Assume its existence

Theorem

There exists a labeling function $f^*(x)$ such that

$$R(f^*) \leq R(f)$$

Namely f^* is optimal. We can call it Bayes optimal.

What does f^* look like?

In fact, we can write down what f^* looks like but it is not computable. We will talk about it later in the semester.

Comparing NNC to Bayes optimal classifier

How well does our NNC do?

Theorem

For the NNC rule f^{NNC} for binary classification, we have,

$$R(f^*) \leq R(f^{\mathrm{NNC}}) \leq 2R(f^*)$$

Namely, the expected risk by the classifier is at worst twice that of the Bayes optimal classifier.

In short, NNC seems doing a reasonable thing

Outline

1 Review of Last Lecture

2 Some practical sides of NNC

3 What we have learned

4 More deep understanding about NNC

5 Summary

Typically, how machine learning systems are developed?

- Get data, split into training, validation and evaluation datasets
- Pick a model/an algorithm
- Train the model on the training dataset and use the validation dataset to pick the best model
- Find the best model and apply to the evaluation dataset
- Report the evaluation result
- (optionally) you can show how good your algorithm is (theoretically)