



Machine Learning

Dr Ajey S.N.R.

Professor & Chairperson

Department of ECE, EC Campus

MACHINE LEARNING

Unit -2: Supervised Learning

Dr Ajey S.N.R

Department of Electronics and Communication
Engineering

- We discuss supervised learning in the following sequence
 - The simplest case, which is learning a class from its positive and negative examples (**two class problem**)
 - We generalize and discuss the case of **multiple classes**,
 - Then **regression**, where the outputs are continuous.
- **Class learning** is finding a **description** that is shared by **all the positive** examples **and none** of the **negative** examples.
- **Prediction**: Given a car that we have **not seen** before, by using the description learned, we will be able to say whether it is a family car or not.

MACHINE LEARNING

Introduction to S L

Learning a Class from Examples

- Class C of a “family car”
 - Prediction: Is car x a family car?
 - Knowledge extraction: What do people expect from a family car?
- Output:
 - Positive (+) and negative (–) examples
- Input representation:
 - x_1 : price, x_2 : engine power

MACHINE LEARNING

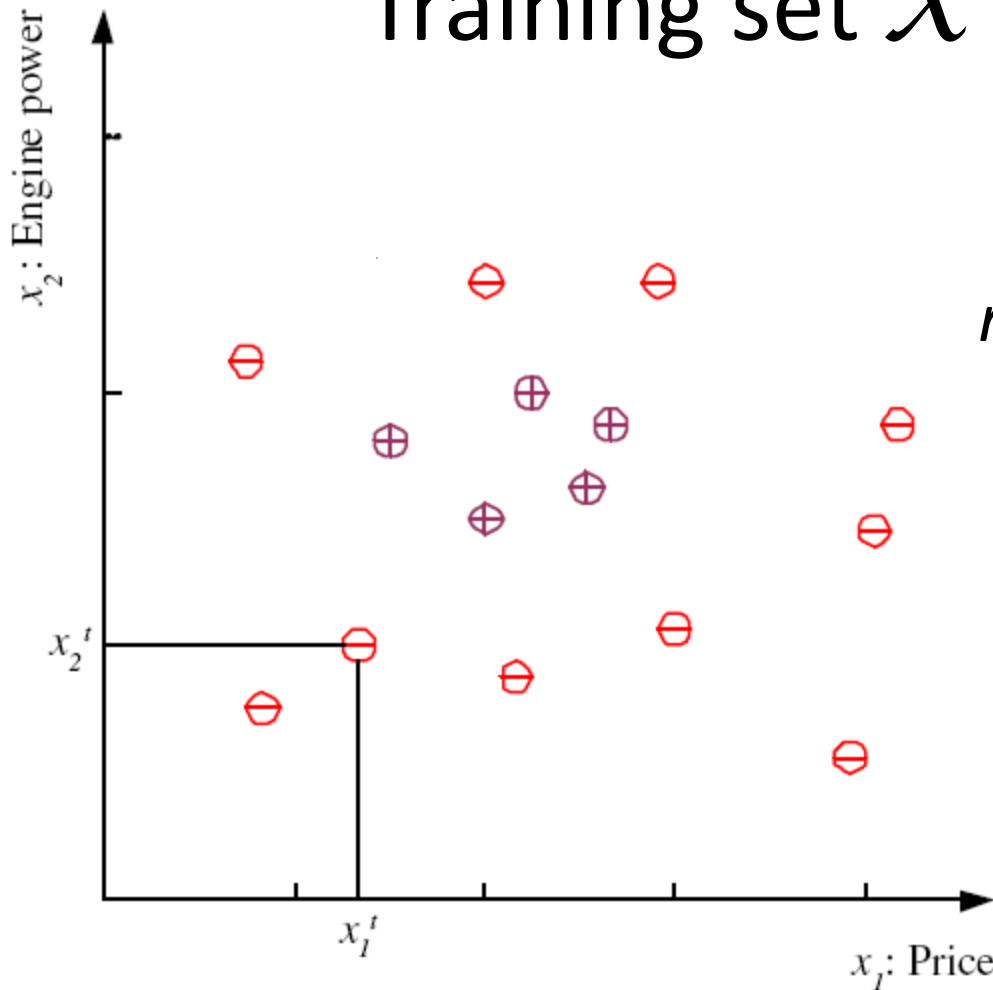
Introduction to SL

Training set \mathcal{X}

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is positive} \\ 0 & \text{if } \mathbf{x} \text{ is negative} \end{cases}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



- After some discussions with experts in the field, let us say that we reach the conclusion that among all features a car may have, the **features that separate** a family car from other type of cars are the **price and engine power**.
- These two **attributes** are the **inputs** to the class recognizer.
- Let us denote price as the first input attribute x_1 and engine power as the second attribute x_2 . Thus we represent each car using two numeric values $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
- and its **label** denotes its type $r = 1$ positive example and $r = 0$ negative example

- Each car is represented by such an ordered pair (x, r) and the training set contains N such examples

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

- where t indexes different examples in the set
- Our training data can now be plotted in the two-dimensional (x₁, x₂) space where each instance t is a data point at coordinates and its type, namely, positive versus negative, is given by r^t

- After further discussions with the expert and the analysis of the data, we may have reason to believe that for a car to be a family car, its price and engine power should be in a certain range

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$

for suitable values of p_1 , p_2 , e_1 , and e_2 .

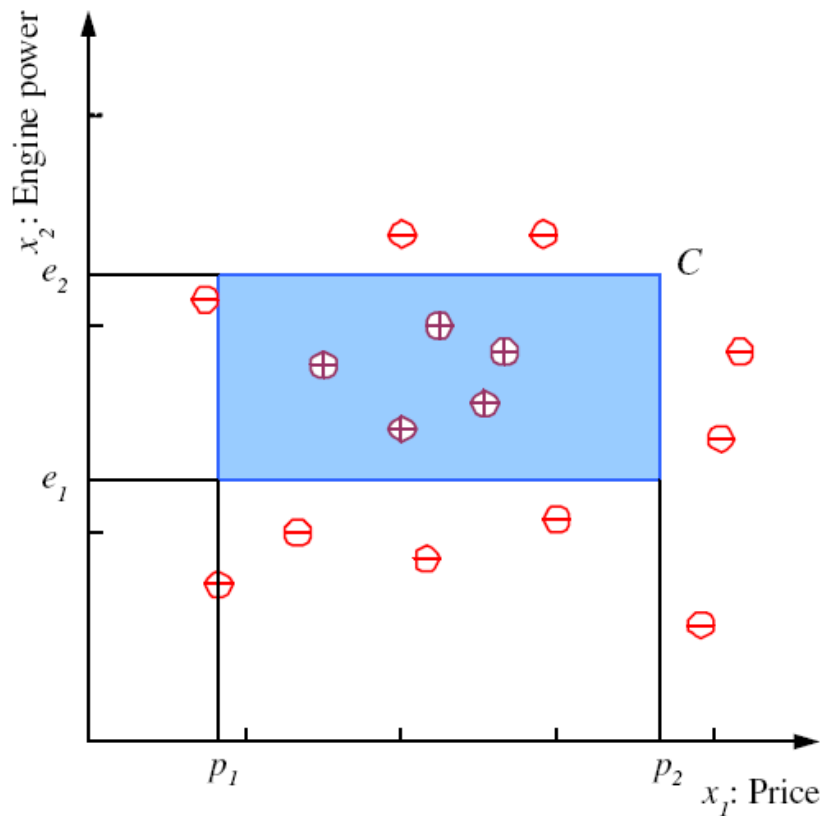
- Thus C is a rectangle in the **Price-Engine power space**.
- H, the hypothesis class from which we believe C is drawn, namely, the **set of rectangles**.
- The learning algorithm then finds the particular hypothesis, $h \in H$, specified by a particular quadruple of $(p_1^h, p_2^h, e_1^h, e_2^h)$,
to approximate C as closely as possible.

MACHINE LEARNING

Introduction to SL

Class C

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$



- Though the expert defines this hypothesis class, the values of the parameters are not known; that is, though we choose H , we do not know which particular $h \in H$ is equal, or closest, to C .
- But once we restrict our attention to this hypothesis class, learning the class reduces to the easier problem of finding the four parameters that define h .
- The aim is to find $h \in H$ that is as similar as possible to C .
- Let us say the hypothesis h makes a prediction for an instance x

$$h(x) = \begin{cases} 1 & \text{if } h \text{ classifies } x \text{ as a positive example} \\ 0 & \text{if } h \text{ classifies } x \text{ as a negative example} \end{cases}$$

- In real life we do not know $C(x)$, so we cannot evaluate how well $h(x)$ matches $C(x)$.
- What we have is the training set X , which is a small subset of the set of all possible x .
- The empirical error is the proportion of training instances where predictions of h do not match the required values given in X . The error of hypothesis h given the training set X is

$$E(h | \mathcal{X}) = \sum_{t=1}^N 1(h(\mathbf{x}^t) \neq r^t)$$

- In our example, the hypothesis class H is the set of all possible rectangles.

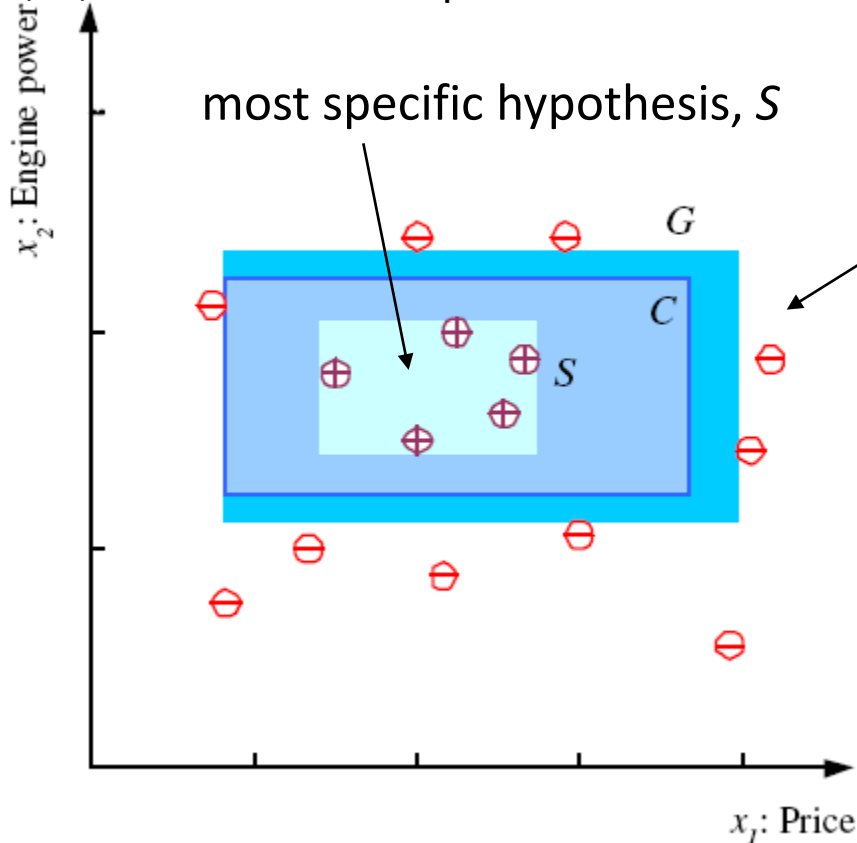
- Each quadruple $(p_1^h, p_2^h, e_1^h, e_2^h)$ defines one hypothesis, 'h' from H, and we need to choose the best one.
- In other words, we need to find the values of these **four parameters given the training set**, to include all the positive examples and none of the negative examples.
- Note that if x_1 and x_2 are real-valued, there are infinitely many such 'h' for which this is satisfied, namely, for which the error, E is 0.

- But given a future example somewhere close to the boundary between positive and negative examples, **different candidate hypotheses may make different predictions**. This is the problem of **generalization**—that is, **how well our hypothesis will correctly classify future examples** that are not part of the training set.
- One possibility is to find the most specific hypothesis, **S**, that is the **tightest rectangle** that includes all the positive examples and none of the negative examples.
- This gives us one hypothesis, $h = S$, as our induced class

MACHINE LEARNING

Introduction to SL

S , G , and the Version Space



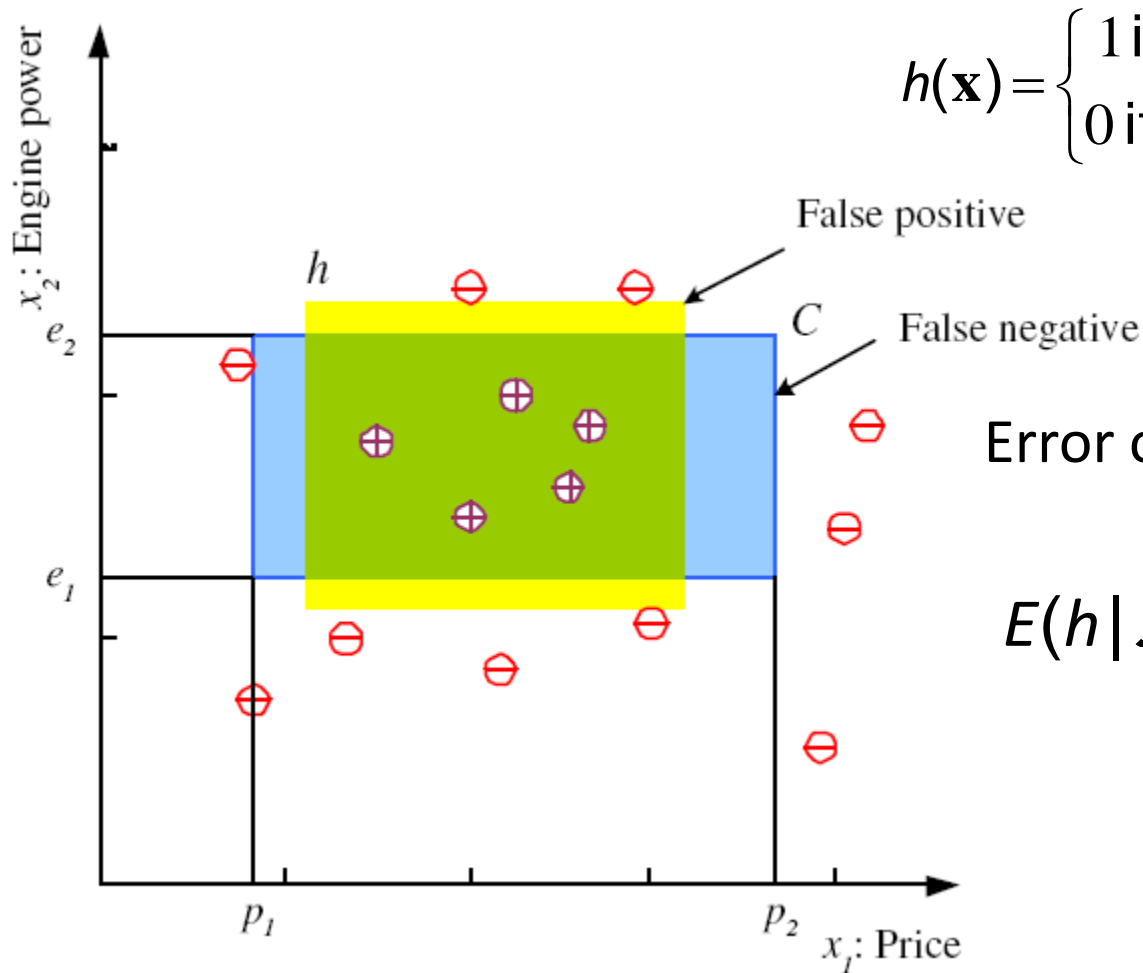
$h \in H$, between S and G is
consistent and make up the
version space
(Mitchell, 1997)

- Note that the actual class C may be larger than S but is never smaller.
- The most general hypothesis, G , is the **largest rectangle** we can draw that includes all the positive examples and none of the negative examples .
- Any $h \in H$ **between S and G** is a valid hypothesis with no error, said to be consistent with the training set, and such h make up the **version space**.
- Given **another training set**, S , G , version space, the parameters and thus the learned hypothesis, h , can be **different** .

MACHINE LEARNING

Introduction to SL

Hypothesis class \mathcal{H}



$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 & \text{if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$

Error of h on \mathcal{H}

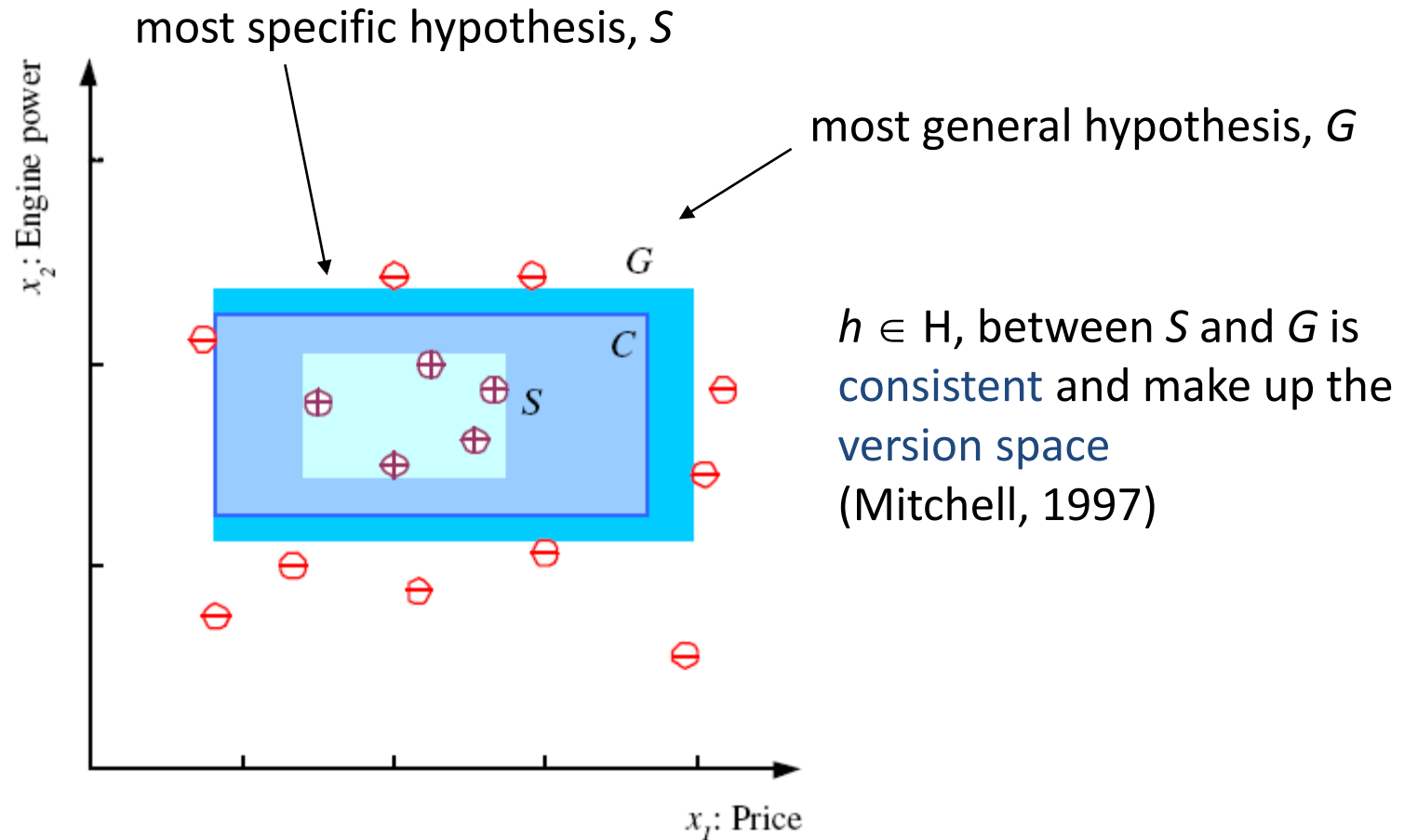
$$E(h | \mathcal{X}) = \sum_{t=1}^N 1(h(\mathbf{x}^t) \neq r^t)$$

- C is the actual class and h is our induced hypothesis.
- The point where C is 1 but h is 0 is a **false negative**, and the point where C is 0 but h is 1 is a **false positive**.
- Other points— namely, **true positives** and **true negatives**—are correctly classified.
- Given X , we can find S , or G , or any h from the version space and use it as our hypothesis, h .

MACHINE LEARNING

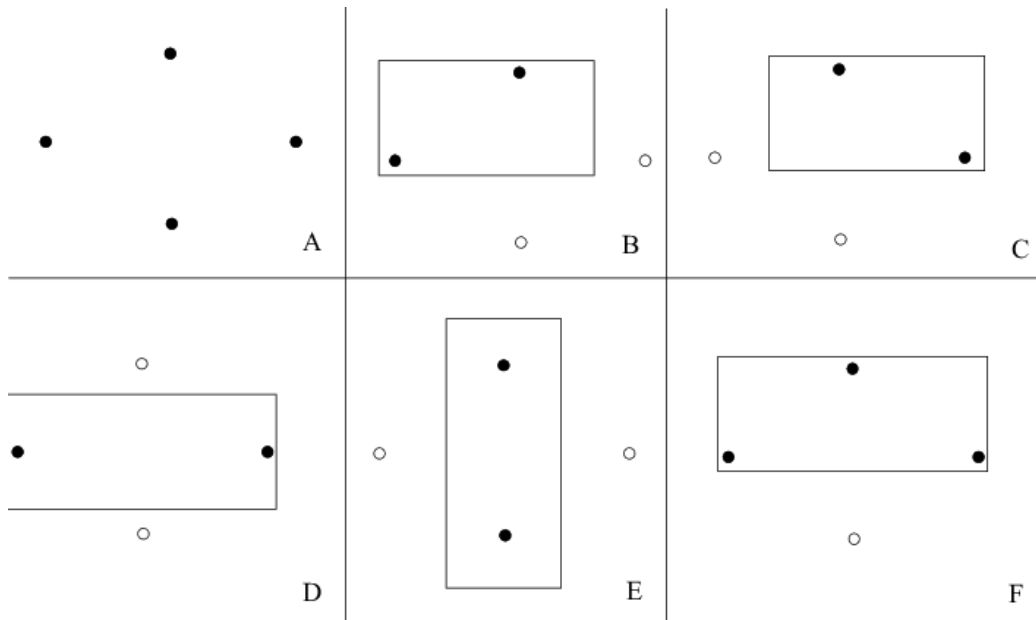
Introduction to S L

S, G, and the Version Space



- Let us say we have a dataset containing N points. These N points can be labeled in 2^N ways as positive and negative.
- Therefore, 2^N different learning problems can be defined by N data points.
- If for any of these problems, we can find a hypothesis $h \in H$ that separates the positive examples from the negative, then we say **H shatters N points**.
- That is, any learning problem definable by N examples can be learned with no error by a hypothesis drawn from H .

- The maximum number of points that can be shattered by H is called the Vapnik Chervonenkis (VC) dimension of H , is denoted as $VC(H)$, and measures the capacity of H .



MACHINE LEARNING

Introduction to SL



E

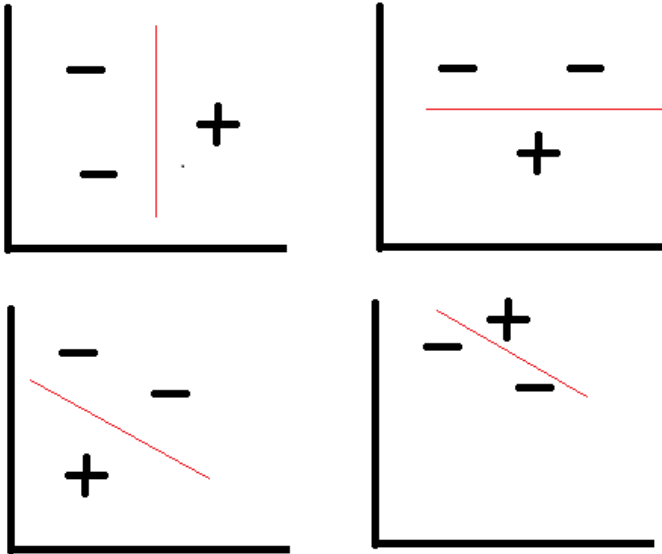


Illustration of shattering 3 points by a line

Fig - 3 (a)

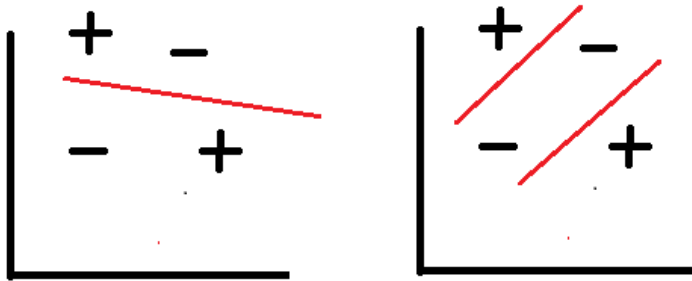


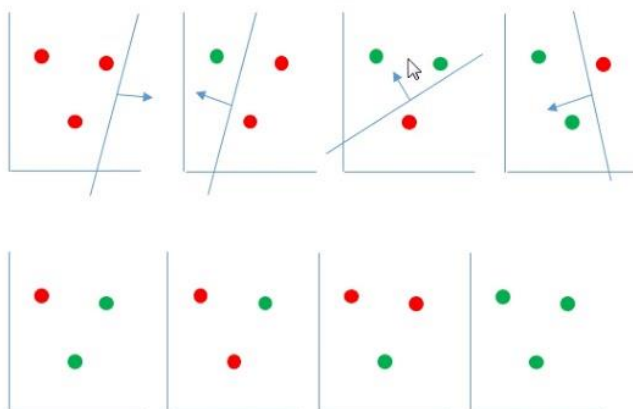
Fig-3 (b) Illustration of case of 4 points where the line was not able to shatter them

MACHINE LEARNING

Introduction to SL

VC Dimension

Can a line hypothesis class shatter 3 data points?



- Using the tightest rectangle, S , as our hypothesis, we would like to find how many examples we need.
- We would like our **hypothesis** to be **approximately correct**, namely, that the error probability be bounded by some value.
- We also would like to be confident in our hypothesis in that we want to know that our hypothesis will be correct **most of the time** (if not always); so we want to be probably correct as well.

Example: of the set of Rectangles is PAC learnable

- In probably approximately correct (PAC) learning, **given a class, C , and examples drawn from some unknown but fixed probability distribution, $p(x)$, we want to find the number of examples, N , such that with probability at least $1 - \delta$, the hypothesis ' h ' has error at most ϵ , (for arbitrary $\delta \leq 1/2$ and $\epsilon > 0$).** **$P\{C \Delta h \leq \epsilon\} \geq 1 - \delta$.**
- Where $C \Delta h$ is the region of difference between C and h .
- Because S is the tightest possible rectangle, the error region between C and $h = S$ is the sum of four rectangular strips.
- Ensure that the probability of a positive example falling in here (and causing an error) is at most ϵ .

- For any of these strips, if we can guarantee that the probability is upper bounded by $\epsilon/4$.
- The total error is at most $4(\epsilon/4) = \epsilon$
- Note that we count the overlaps in the corners twice, and the total actual error in this case is less than $4(\epsilon/4)$.
- The probability that a randomly drawn example misses this strip is $1 - \epsilon/4$.
- The probability that all N independent draws miss the strip is $(1 - \epsilon/4)^N$.
- The probability that all N independent draws miss any of the four strips is at most $4(1 - \epsilon/4)^N$, which we would like to be at most δ . We have the inequality $(1 - x) \leq \exp[-x]$

- Using the inequality $(1 - x) \leq \exp[-x]$
- So if we choose N and δ such that we have
$$4 \exp[-\epsilon N/4] \leq \delta$$
- we can also write $4(1 - \epsilon/4)^N \leq \delta$.
- Dividing both sides by 4 and taking (natural) log and rearranging terms, we have
- **$N \geq (4/\epsilon)\log(4/\delta)$**

- Therefore, provided that we take at least $(4/\epsilon) \log(4/\delta)$ independent examples from C and use the tightest rectangle as our hypothesis h , with confidence probability at least $1-\delta$, a given point will be misclassified with error probability at most ϵ .
- We can have arbitrary large confidence by decreasing δ and arbitrary small error by decreasing ϵ , and
- The number of examples is a slowly growing function (polynomial) of $1/\epsilon$ and $1/\delta$, linear and logarithmic, respectively.

Probably Approximately Correct (PAC) Learning: Summary

- How many training examples N should we have, such that with **probability at least $1 - \delta$** , h has error at most ϵ ?

Each strip is at most $\epsilon/4$

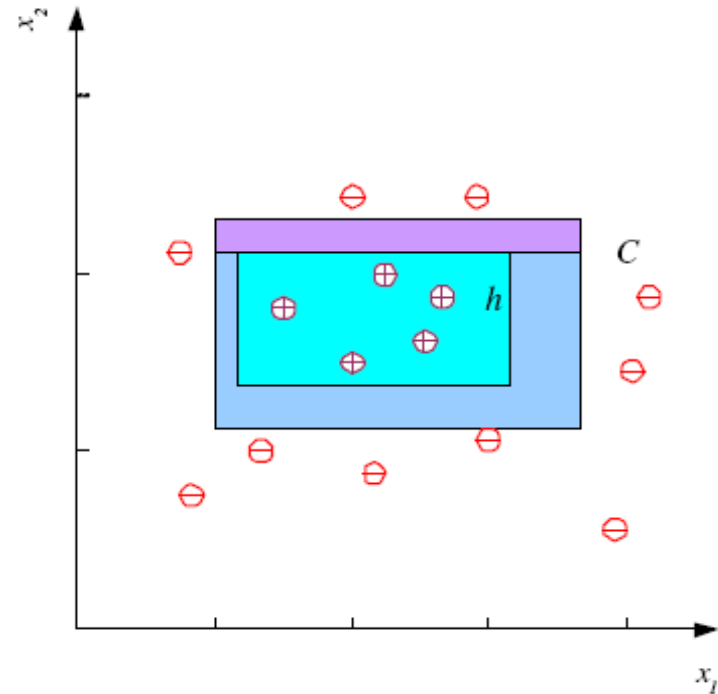
Pr that we miss a strip $1 - \epsilon/4$

Pr that N instances miss a strip $(1 - \epsilon/4)^N$

Pr that N instances miss 4 strips $4(1 - \epsilon/4)^N$

$4(1 - \epsilon/4)^N \leq \delta$ and $(1 - x) \leq \exp(-x)$

$4\exp(-\epsilon N/4) \leq \delta$ and $N \geq (4/\epsilon)\log(4/\delta)$



- Noise is any **unwanted** anomaly in the data and due to noise, the class may be more difficult to learn and zero error may be infeasible with a simple hypothesis class.
- There are several **interpretations** of noise:
 - There may be **imprecision** in recording the input attributes, which may **shift the data points** in the input space.
 - There may be **errors in labeling** the data points, which may re-label positive instances as negative and vice versa. This is sometimes called **teacher noise**.

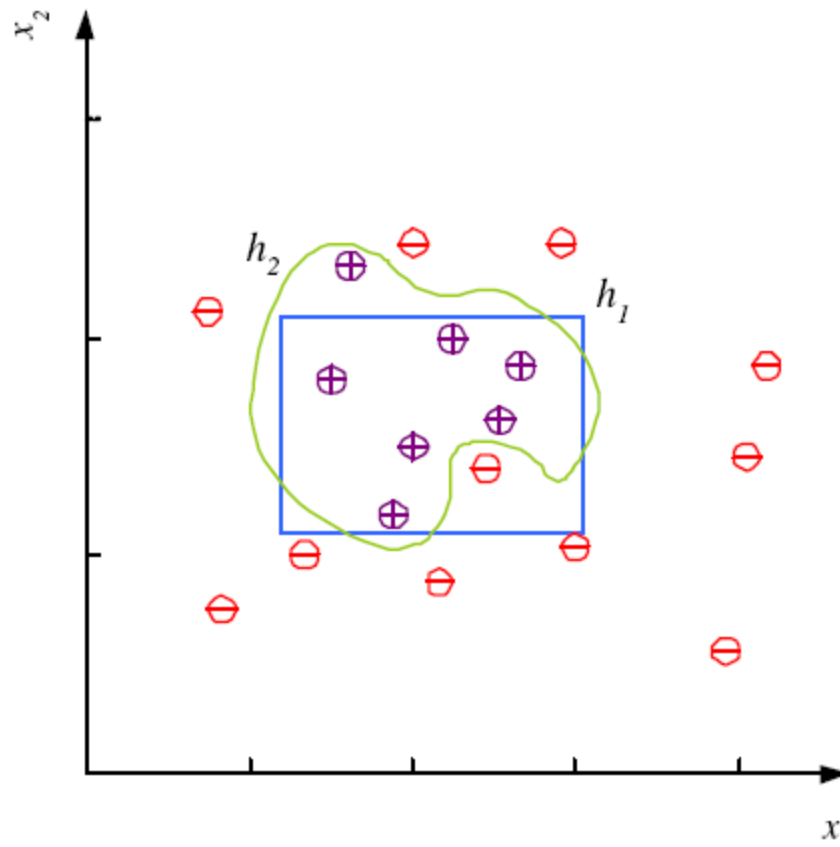
- There may be additional **attributes**, which we have **not taken into account**, that affect the label of an instance. Such attributes may be hidden or latent in that they may be unobservable. The effect of these **neglected attributes** is thus modeled as a random component and is included in “**noise**”
- When there is noise, there is **not a simple boundary** between the positive and negative instances and to separate them, one needs a **complicated hypothesis** that corresponds to a hypothesis class with larger capacity.
- Keep the model **simple** and **allow** some error.

Machine Learning

Noise and Model Complexity

Use the simpler one because

- Simpler to use
(lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain
(more interpretable)
- Generalizes better (lower variance - Occam's razor)



MACHINE LEARNING

Multiple Classes, C_i $i=1,\dots,K$



- In our example of learning a family car, we have positive examples belonging to the class family car and the negative examples belonging to all other cars. This is a two-class problem.
- In the general case, we have K classes denoted as $C_i, i = 1, \dots, K$, and an input instance belongs to one and exactly one of them.
- The training set is now of the form

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

- where r has K dimensions and $r_i^t = \begin{cases} 1 & \text{if } \mathbf{x}^t \in C_i \\ 0 & \text{if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$

MACHINE LEARNING

Multiple Classes, C_i $i=1,...,K$



- In machine learning for classification, we would like to learn the boundary separating the instances of one class from the instances of all other classes.
- Thus we view a K-class classification problem as **K two-class problems**.
- The training examples belonging to C_i are the positive instances of hypothesis h_i and the examples of all other classes are the negative instances of h_i . $h_i(\mathbf{x})$, $i=1,...,K$
- Thus in a K-class problem, we have 'K' hypotheses to learn such that

$$h_i(\mathbf{x}^t) = \begin{cases} 1 & \text{if } \mathbf{x}^t \in C_i \\ 0 & \text{if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

- The total empirical error takes a sum over the predictions for all classes over all instances:

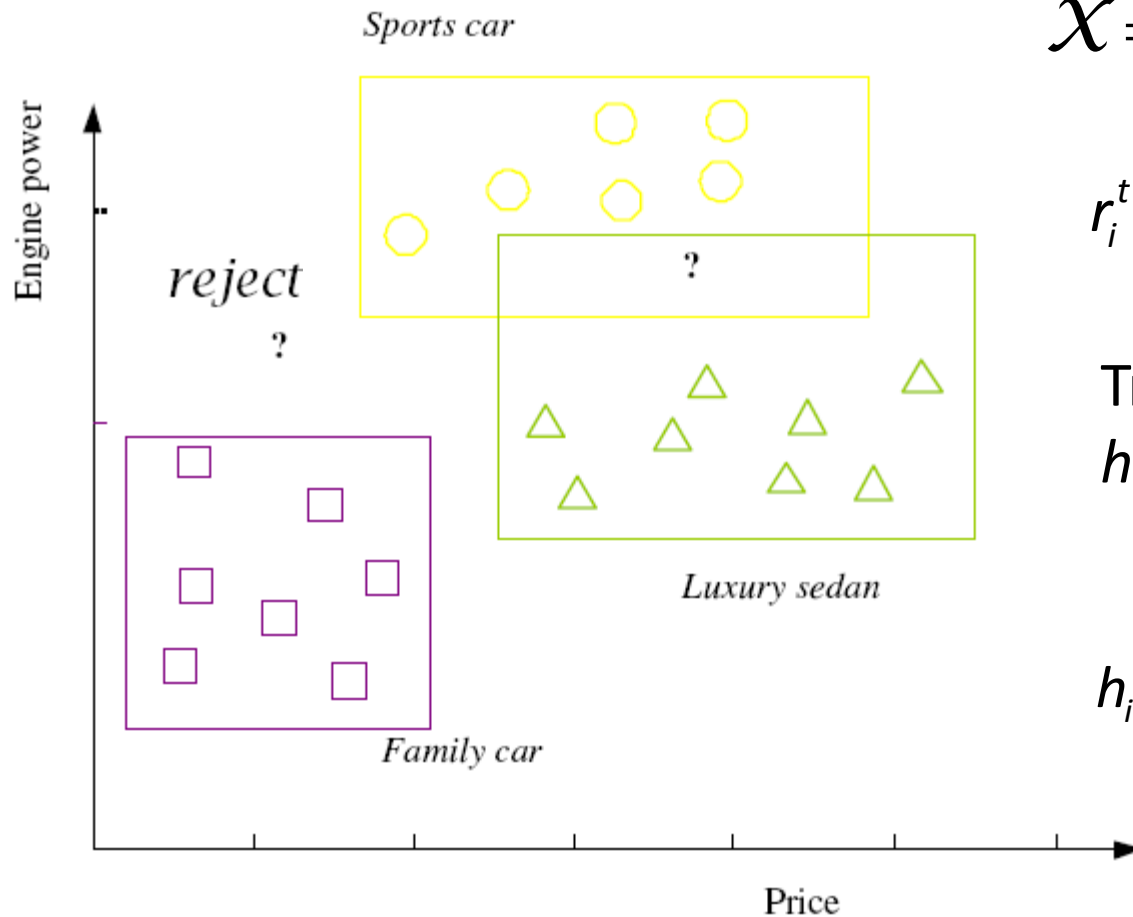
$$E(\{h_i\}_{i=1}^K | \mathcal{X}) = \sum_{t=1}^N \sum_{i=1}^K 1(h_i(\mathbf{x}^t) \neq r_i^t)$$

- For a given \mathbf{x} , ideally only one of $h_i(\mathbf{x})$, $i = 1, \dots, K$ is 1 and we can choose a class.
- But when no, or two or more, $h_i(\mathbf{x})$ is 1, we cannot choose a class, and this is the case of doubt and the classifier rejects such cases.

MACHINE LEARNING

Multiple Classes, $C_i, i=1, \dots, K$

Multiple Classes, $C_i, i=1, \dots, K$



$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r_i^t = \begin{cases} 1 & \text{if } \mathbf{x}^t \in C_i \\ 0 & \text{if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses

$h_i(\mathbf{x}), i = 1, \dots, K$:

$$h_i(\mathbf{x}^t) = \begin{cases} 1 & \text{if } \mathbf{x}^t \in C_i \\ 0 & \text{if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

- In classification, given an input, the output that is generated is Boolean; it is a yes/no answer.
- When the output is a numeric value, what we would like to learn is not a class, $C(x) \in \{0, 1\}$, but is a numeric function.
- In machine learning, the function is **not known** but we have a training set of examples drawn from it
- Where $r^t \in \mathbb{R}$. If there is **no noise**, the task is **interpolation**.
- We would like to find the function $f(x)$ that passes through these points such that we have
$$r^t = f(x^t)$$

Regression

Regression

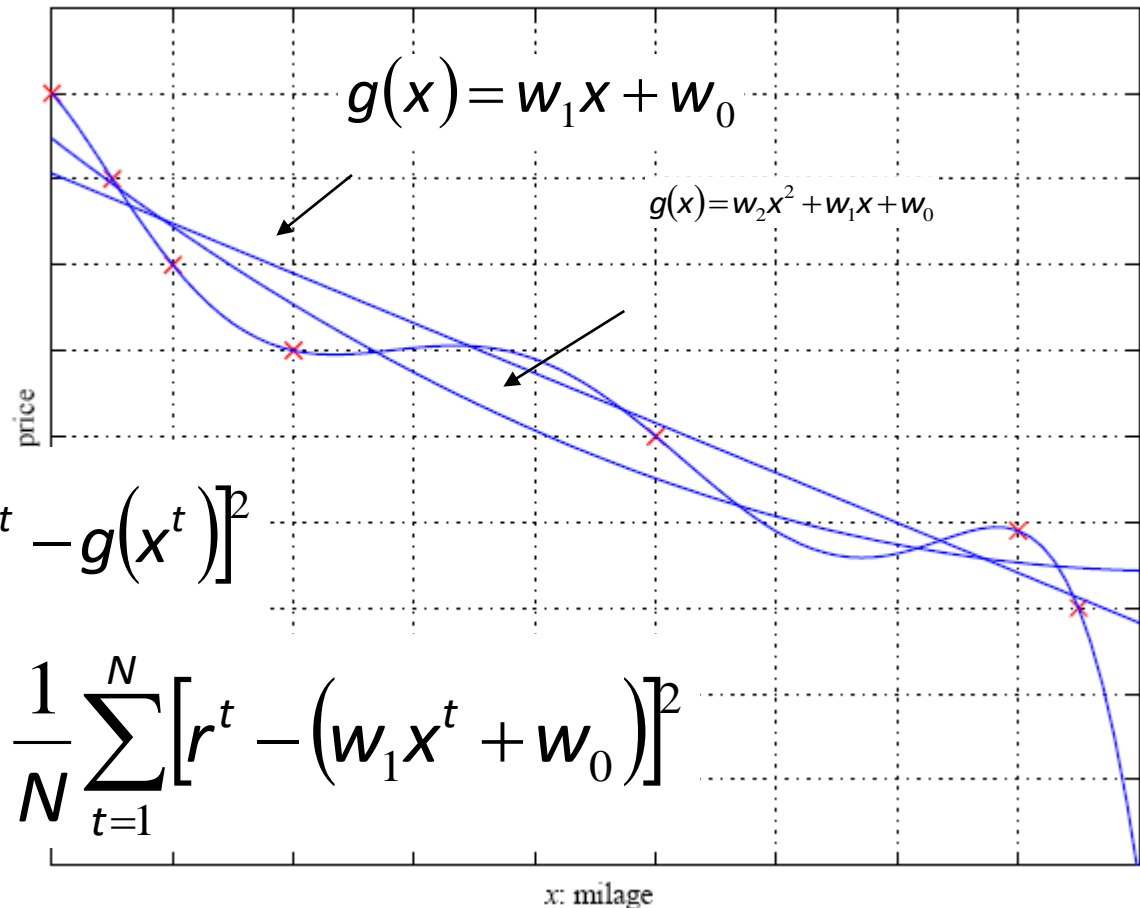
$$\mathcal{X} = \{x^t, r^t\}_{t=1}^N$$

$$r^t \in \mathbb{R}$$

$$r^t = f(x^t) + \varepsilon$$

$$E(g | \mathcal{X}) = \frac{1}{N} \sum_{t=1}^N [r^t - g(x^t)]^2$$

$$E(w_1, w_0 | \mathcal{X}) = \frac{1}{N} \sum_{t=1}^N [r^t - (w_1 x^t + w_0)]^2$$



$$\triangleright \mathcal{X} = \{x^t, r^t\}_{t=1}^N$$

$$r^t \in \mathbb{R}$$

$$r^t = f(x^t) + \varepsilon$$

- In **Polynomial interpolation**, given N points, we find the $(N - 1)^{\text{st}}$ degree polynomial that we can use to predict the output for any x.
- This is called **Extrapolation** if x is outside of the range of x^t in the training set.
- In time-series prediction, for example, we have data up to the present and we want to predict the value for the future.

- In Regression, there is noise added to the output of the unknown function $\mathbf{r}^t = \mathbf{f}(\mathbf{x}^t) + \epsilon$
- Where $f(x) \in \mathfrak{R}$ is the **unknown function** and ϵ is random noise.
- The explanation for noise is that there are **extra hidden variables** that we cannot observe.
- $\mathbf{r}^t = \mathbf{f}^*(\mathbf{x}^t \mathbf{z}^t)$, \mathbf{z}^t denote those **hidden variables**.
- We would like to approximate the output by our model $g(x)$.

- The empirical error on the training set X is

$$E(g | \mathcal{X}) = \frac{1}{N} \sum_{t=1}^N [r^t - g(x^t)]^2$$

- Because r and $g(x)$ are numeric quantities, for example, $\in \mathbb{R}$, there is an ordering defined on their values and we can define a distance between values, as the square of the difference, which gives us more information than equal/not equal, as used in classification.
- The square of the difference is one error (loss) function that can be used; another is the absolute value of the difference.

- Our aim is to find $g(\cdot)$ that minimizes the empirical error.
- Again our approach is the same; we assume a hypothesis class for $g(\cdot)$ with a small set of parameters.
- If we assume that $g(x)$ is linear, we have

$$g(\mathbf{x}) = w_1x_1 + \dots + w_dx_d + w_0 = \sum_{j=1}^d w_jx_j + w_0$$

- If the assumed model is linear $g(x) = w_1x + w_0$

where w_1 and w_0 are the parameters to learn from data. The w_1 and w_0 values should minimize

$$E(w_1, w_0 | \mathcal{X}) = \frac{1}{N} \sum_{t=1}^N [r^t - (w_1x^t + w_0)]^2$$

- Its minimum point can be calculated by taking the partial derivatives of E with respect to w_1 and w_0 , setting them equal to 0, and solving for the two unknowns.

$$w_1 = \frac{\sum_t x^t r^t - \bar{X} \bar{r} N}{\sum_t (x^t)^2 - N \bar{X}^2}$$
$$w_0 = \bar{r} - w_1 \bar{X}$$

where $\bar{X} = \sum_t x^t / N$ and $\bar{r} = \sum_t r^t / N$.

- If the linear model is too simple, it is too constrained and incurs a large approximation error, and in such a case, the output may be taken as a higher-order function of the input— for example, quadratic.

- If the training set we are given contains only a **small subset of all possible instances**, as it generally does — that is, if we know what the output should be for only a small percentage of the **cases—the solution is not unique**. This is an example of an ill-posed problem
- As we see **more training examples, we know more about the underlying function**, and we carve out more hypotheses that are inconsistent from the hypothesis class, but we still are left with many consistent hypotheses.
- Because learning is **ill-posed**, and data by itself is not sufficient to find the solution, we should make some extra **assumptions** to have a unique solution with the data we have

- The set of assumptions we make to have learning possible is called the **inductive bias** of the learning algorithm
- One way we introduce inductive bias is when we assume a hypothesis class H .
- In learning the class of family cars, there are infinitely many ways of separating the positive examples from the negative examples.

Model selection & Generalization

- Assuming the shape of a rectangle is one inductive bias, and then the rectangle with the largest margin, for example, is another inductive bias. In linear regression, assuming a linear function is an inductive bias, and among all lines, choosing the one that minimizes squared error is another inductive bias.
- Learning is not possible without inductive bias, and now the question is **how to choose the right bias**. This is called **model selection**, which is choosing between possible H.
- How well a model trained on the training set predicts the right output **for new instances** is called **generalization**.

- For best generalization, we should **match** the complexity of the hypothesis class H with the complexity of the function underlying the data. If H is less complex than the function, we have **under-fitting**.
- For example, when trying to fit a line to data sampled from a third-order polynomial.
- When fitting a sixth- order polynomial to noisy data sampled from a third-order polynomial. This is called **over-fitting**.

- In all learning algorithms that are trained from example data, there is a **trade-off between three factors**:
 - The complexity of the hypothesis we fit to data, namely, the capacity of the hypothesis class,
 - The amount of training data,
 - And the generalization error on new examples.

- We can measure the generalization ability of a hypothesis, namely, the quality of its inductive bias, if we have access to data outside the training set.

- We simulate this by dividing the dataset we have into **two parts**. We use one part for **training** (i.e., to fit a hypothesis), and the remaining part is called the **validation set** and is used to test the generalization ability.
- That is, given a set of possible hypothesis **classes H_i** , for each we fit the best $h_i \in H_i$ on the training set.
- Then, assuming large enough training and validation sets, the hypothesis that is the most accurate on the validation set is the best one (the one that has the best inductive bias).
- This process is called **cross-validation**.

- For example, to find the right order in polynomial regression, given a number of candidate polynomials of different orders where polynomials of different orders correspond to H_i , for each order, we find the coefficients on the training set, calculate their errors on the validation set, and take the one that has the least validation error as the best polynomial.
- We have used the validation set to choose the best model, and it has effectively become a part of the training set.
- We need a third set, a **test set**, sometimes also called the **publication set**, containing examples not used in training or validation.

- Learning is an **ill-posed problem**; data is not sufficient to find a unique solution
- The need for **inductive bias**, assumptions about \mathcal{H}
- **Generalization**: How well a model performs on new data
- Overfitting: \mathcal{H} more complex than C or f
- Underfitting: \mathcal{H} less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
 1. Complexity of \mathcal{H} , $c(\mathcal{H})$,
 2. Training set size, N ,
 3. Generalization error, E , on new data
- ☐ As N , $E \downarrow$
 - ☐ As $c(\mathcal{H})$, first $E \downarrow$ and then E

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
 - Training set (50%)
 - Validation set (25%)
 - Test (publication) set (25%)
- Resampling when there is few data

- Let us now recapitulate and generalize.

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

- The sample is independent and identically distributed (iid);
- The ordering is not important and all instances are drawn from the same joint distribution $p(\mathbf{x}, r)$.
- 't' indexes one of the N instances,
- \mathbf{x}^t is the arbitrary dimensional input, and r^t is the associated desired output.
- r^t is 0/1 for two-class learning, is a K-dimensional binary vector (where exactly one of the components is 1 and all others 0) for $(K > 2)$ -class classification,
- and is a real value in Regression.

- The aim is to build a good and useful approximation to r^t using the model $g(x^t | \theta)$.
- In doing this, there are three decisions we must make
 1. **Model** we use in learning, denoted as $g(x | \theta)$
where $g(\cdot)$ is the model, x is the input, and θ are the parameters.
 - $g(\cdot)$ defines the hypothesis class H , and
 - a particular value of θ instantiates one hypothesis $h \in H$.

MACHINE LEARNING

Dimensions of a supervised machine learning algorithm



- For example, in class learning, we have taken a rectangle as our model whose four coordinates make up θ ;
- In linear regression, the model is the linear function of the input whose slope and intercept are the parameters learned from the data.
- The model (inductive bias), or H , is fixed by the machine learning system designer based on his or her knowledge of the application and the hypothesis h is chosen (parameters are tuned) by a learning algorithm using the training set, sampled from $p(x, r)$.

2. Loss function $L(\cdot)$: To compute the difference between the desired output, r^t , and our approximation to it, $g(\mathbf{x}^t | \theta)$, given the current value of the parameters, θ .

- The approximation error, or loss, is the sum of losses over the individual instances.

$$E(\theta | \mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t | \theta))$$

- In class learning where outputs are 0/1, $L(\cdot)$ checks for equality or not; in regression, because the output is a numeric value, we have ordering information for distance and one possibility is to use the square of the difference.

3. Optimization procedure to find θ^* that minimizes the

total error $\theta^* = \arg\min_{\theta} E(\theta | \mathcal{X})$

where arg min returns the argument that minimizes.

- In polynomial regression, we can solve analytically for the optimum, but this is not always the case.
- With other models and error functions, the complexity of the optimization problem becomes important. We are especially interested in whether it has a single minimum corresponding to a globally optimal solution, or whether there are multiple minima corresponding to locally optimal solutions.

MACHINE LEARNING

Dimensions of a Supervised Learner: Summary.

1. Model: $g(\mathbf{x} | \theta)$

2. Loss function: $E(\theta | \mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t | \theta))$

3. Optimization procedure:

$$\theta^* = \arg \min_{\theta} E(\theta | \mathcal{X})$$

- Data comes from a process that is not completely known.
- This lack of knowledge is indicated by modeling the process as a **random process**.
- The extra pieces of knowledge that we do not have access to are named the **unobservable variables**.
- In the coin tossing example, the only **observable variable** is the outcome of the toss.
- Denoting the unobservables by z and the observable as x , in reality we have $x = f(z)$ where $f(\cdot)$ is the deterministic function that defines the outcome from the unobservable pieces of knowledge.

Bayesian Decision Theory

Probability and Inference

- Result of tossing a coin is $\in \{\text{Heads}, \text{Tails}\}$
- **Random var** $X \in \{1, 0\}$

$$\text{Bernoulli: } P\{X=1\} = p_o^X (1 - p_o)^{(1-X)}$$

- **Sample:** $\mathbf{X} = \{x^t\}_{t=1}^N$

$$\text{Estimation: } p_o = \# \{\text{Heads}\} / \#\{\text{Tosses}\} = \sum_t x^t / N$$

- **Prediction** of next toss:

Heads if $p_o > 1/2$, Tails otherwise

- Because we cannot model the process this way, we define the **outcome X as a random variable** drawn from a probability distribution $P(X = x)$ that specifies the process.
- $X = 1$ denotes that the outcome of a toss is heads and $X = 0$ denotes tails.
- Such X are Bernoulli-distributed
- If we do not know $P(X)$ and want to estimate this from a given sample, then we are in the realm of statistics.
- We have a sample, χ , containing examples drawn from the probability distribution of the observables x^t , denoted as $p(x)$.
- The aim is to build an approximator to it, $\widehat{p(x)}$, using the sample X .

Bayesian Decision Theory(classification)

- As an example, the credibility of a customer is denoted by a Bernoulli random variable C conditioned on the observables $X = [X_1, X_2]^T$ where $C = 1$ indicates a high-risk customer and $C = 0$ indicates a low-risk customer.
- Thus if we know $P(C|X_1, X_2)$, when a new application arrives with $X_1 = x_1$ and $X_2 = x_2$, we can

or equivalently

$$\text{choose} \begin{cases} C = 1 & \text{if } P(C = 1|x_1, x_2) > 0.5 \\ C = 0 & \text{otherwise} \end{cases}$$

$$\text{choose} \begin{cases} C = 1 & \text{if } P(C = 1|x_1, x_2) > P(C = 0|x_1, x_2) \\ C = 0 & \text{otherwise} \end{cases}$$

- The Bernoulli random variable C is conditioned on two other observable variables.

MACHINE LEARNING

Classification

- Credit scoring: Inputs are income and savings.
Output is low-risk vs high-risk
- Input: $\mathbf{x} = [x_1, x_2]^T$, Output: $C \in \{0, 1\}$
- Prediction:

$$\text{choose } \begin{cases} C = 1 \text{ if } P(C = 1 | x_1, x_2) > 0.5 \\ C = 0 \text{ otherwise} \end{cases}$$

or

$$\text{choose } \begin{cases} C = 1 \text{ if } P(C = 1 | x_1, x_2) > P(C = 0 | x_1, x_2) \\ C = 0 \text{ otherwise} \end{cases}$$

- The problem then is to be able to calculate $P(C|x)$.
- Using Bayes' rule, it can be written as

$$P(C|x) = \frac{P(C)p(x|C)}{p(x)}$$

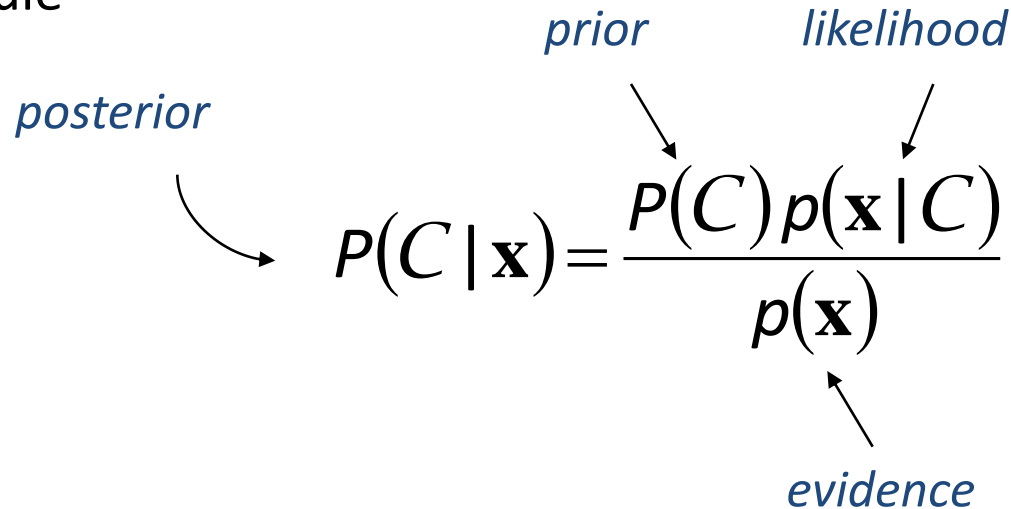
- $P(C = 1)$ is called the **prior probability** that C takes the value 1, which in our example corresponds to the probability that a customer is high-risk, regardless of the x value—It is the proportion of high-risk customers in our customer base.
- It is called the prior probability because it is the knowledge we have as to the value of C before looking at the observables x ,
- Satisfying $P(C = 0) + P(C = 1) = 1$

- $p(x|C)$ is called the **class likelihood** and is the conditional probability that an event belonging to C has the associated observation value x .
- In our case, $p(x_1, x_2|C = 1)$ is the probability that a high-risk customer has his or her $X_1 = x_1$ and $X_2 = x_2$. It is what the data tells us regarding the class.
- $p(x)$, the **evidence**, is the marginal probability that an observation x is seen, regardless of whether it is a positive or negative example
- **Combining the prior and what the data tells us** using Bayes' rule, we calculate the **posterior probability** of the concept, $P(C|x)$, after having seen the observation, x

Bayesian Decision Theory

66

Bayes' Rule



The diagram shows the equation for Bayes' Rule: $P(C | \mathbf{x}) = \frac{P(C)p(\mathbf{x} | C)}{p(\mathbf{x})}$. Annotations include: 'posterior' with a curved arrow pointing to $P(C | \mathbf{x})$; 'prior' with an arrow pointing to $P(C)$; 'likelihood' with an arrow pointing to $p(\mathbf{x} | C)$; and 'evidence' with an arrow pointing to $p(\mathbf{x})$.

$$P(C | \mathbf{x}) = \frac{P(C)p(\mathbf{x} | C)}{p(\mathbf{x})}$$

$$P(C = 0) + P(C = 1) = 1$$

$$p(\mathbf{x}) = p(\mathbf{x} | C = 1)P(C = 1) + p(\mathbf{x} | C = 0)P(C = 0)$$

$$p(C = 0 | \mathbf{x}) + p(C = 1 | \mathbf{x}) = 1$$

Bayesian Decision Theory

67

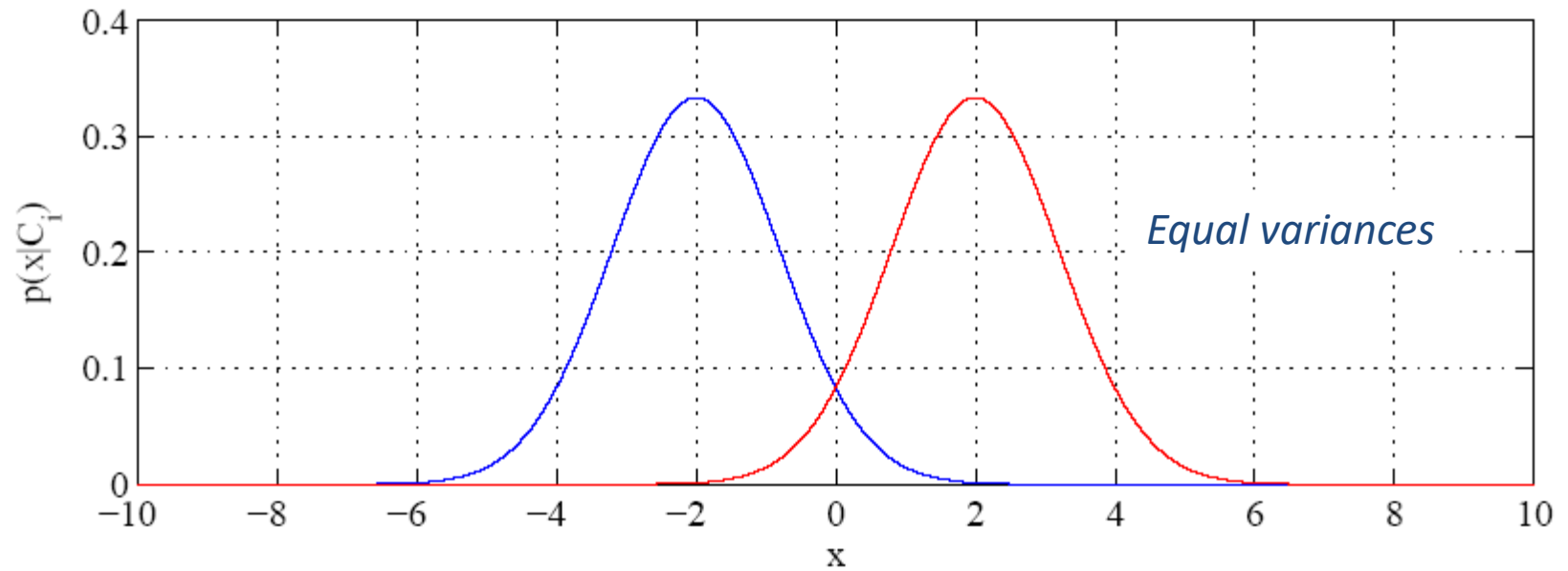
Bayes' Rule: $K > 2$ Classes

$$\begin{aligned} P(C_i | \mathbf{x}) &= \frac{p(\mathbf{x} | C_i)P(C_i)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x} | C_i)P(C_i)}{\sum_{k=1}^K p(\mathbf{x} | C_k)P(C_k)} \end{aligned}$$

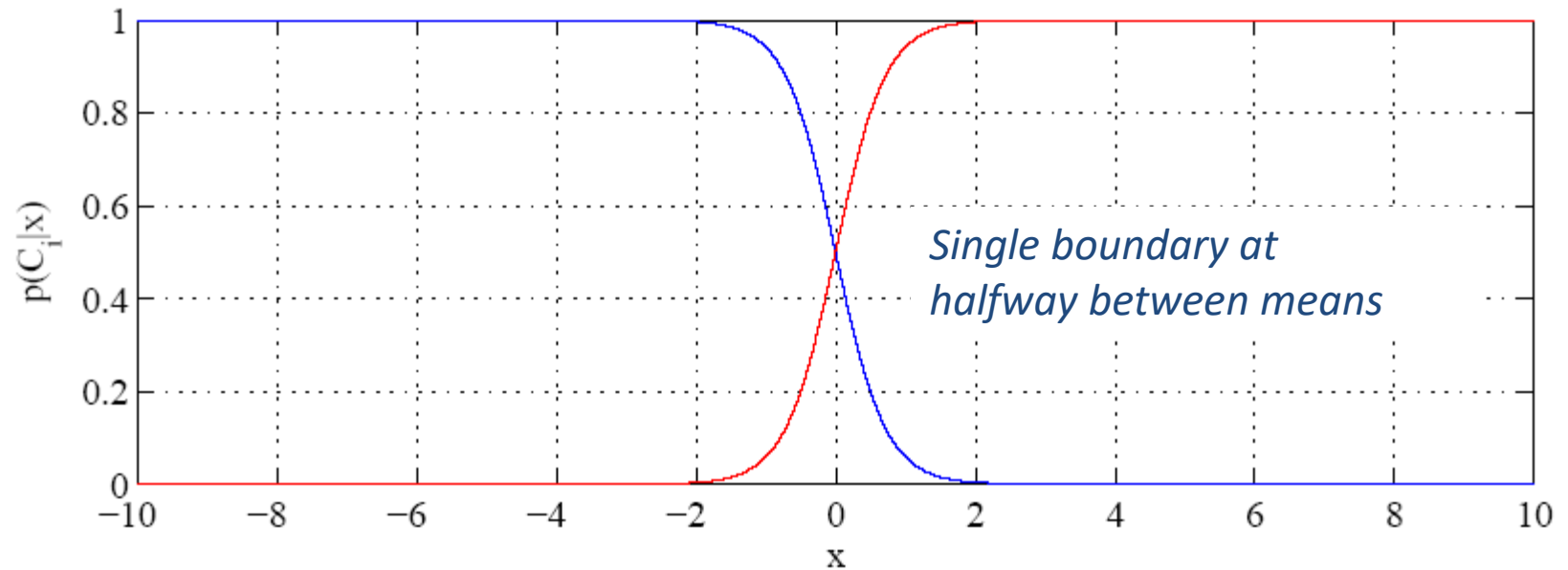
$$P(C_i) \geq 0 \text{ and } \sum_{i=1}^K P(C_i) = 1$$

choose C_i if $P(C_i | \mathbf{x}) = \max_k P(C_k | \mathbf{x})$

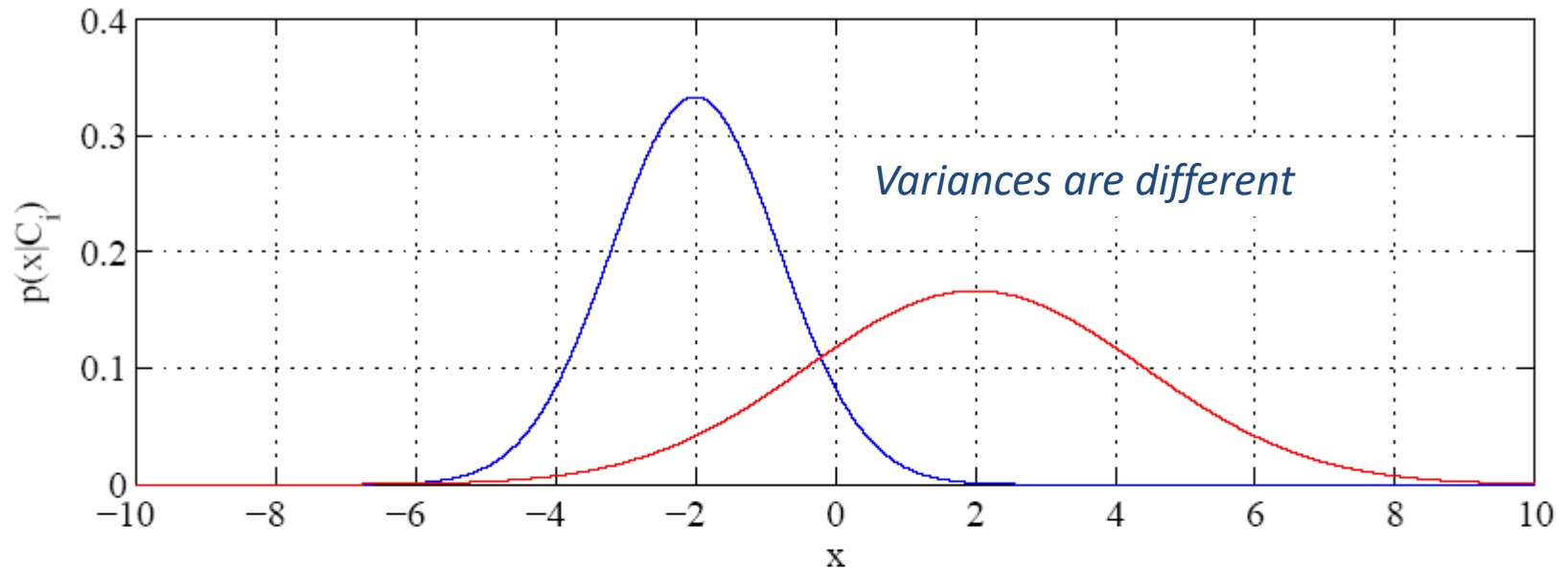
Likelihoods



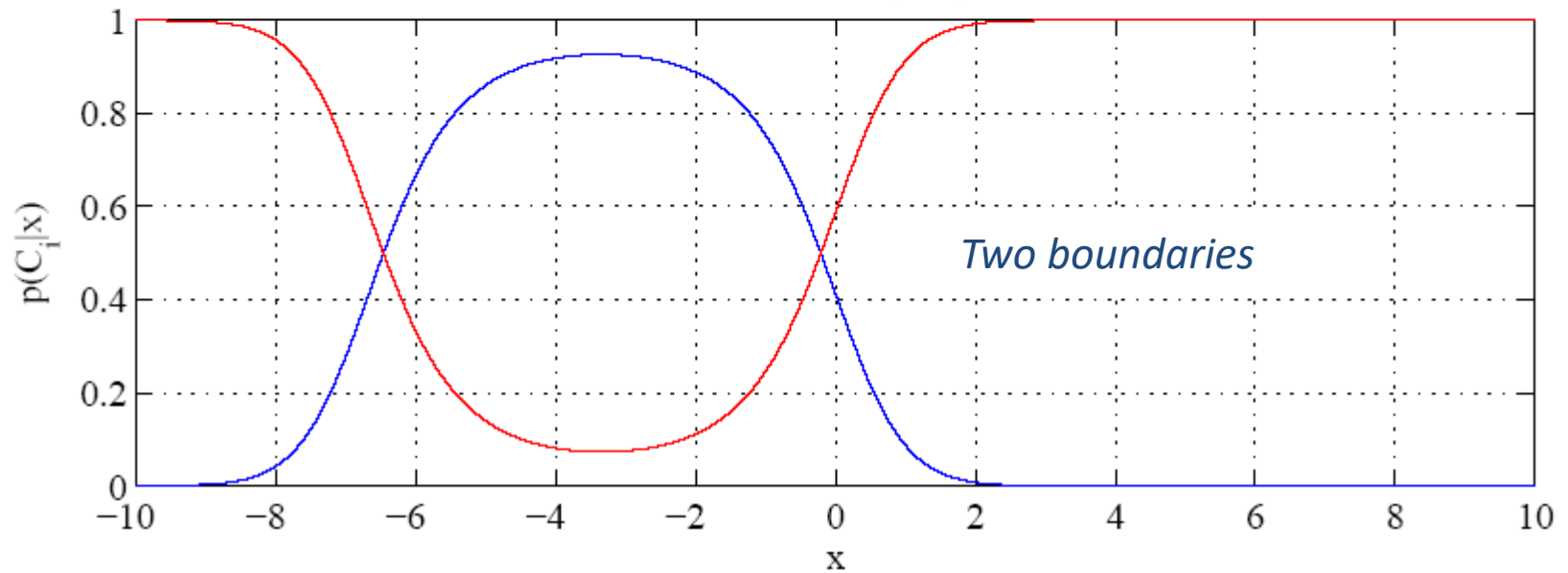
Posteriors with equal priors

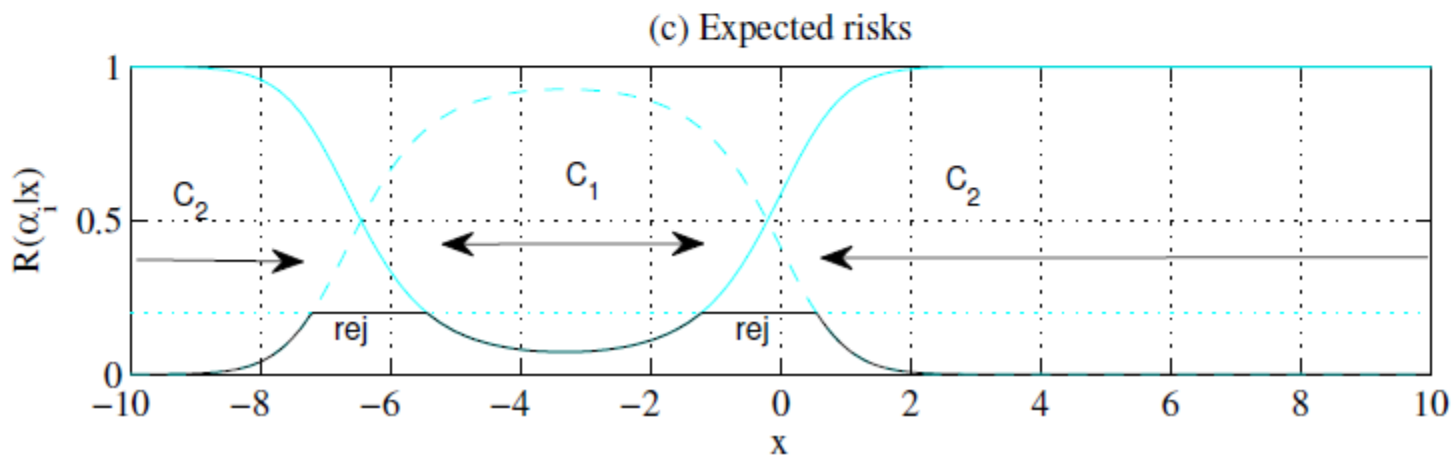
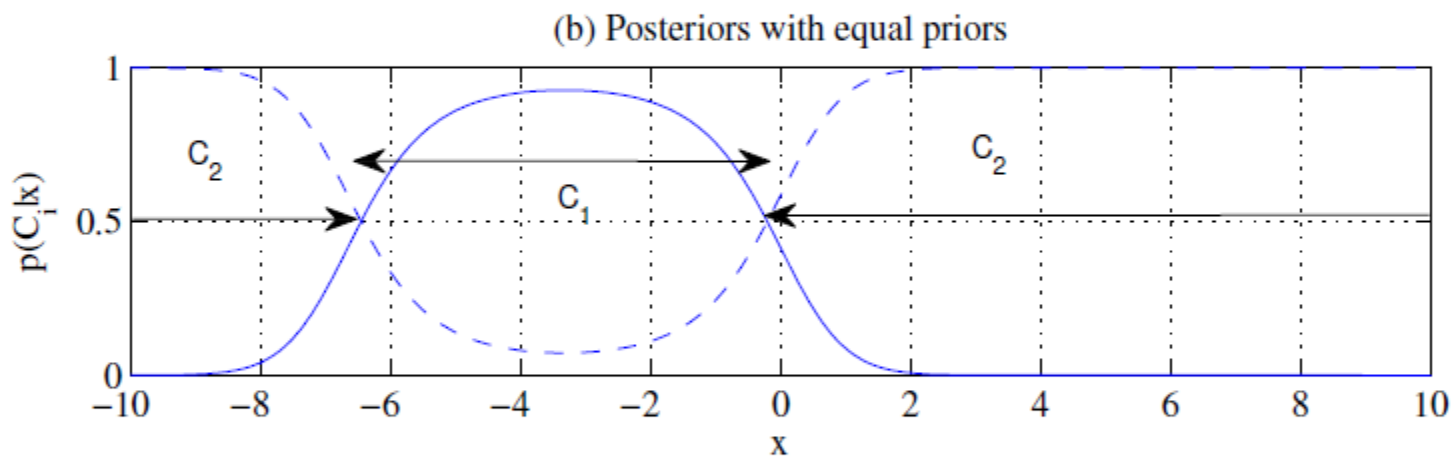
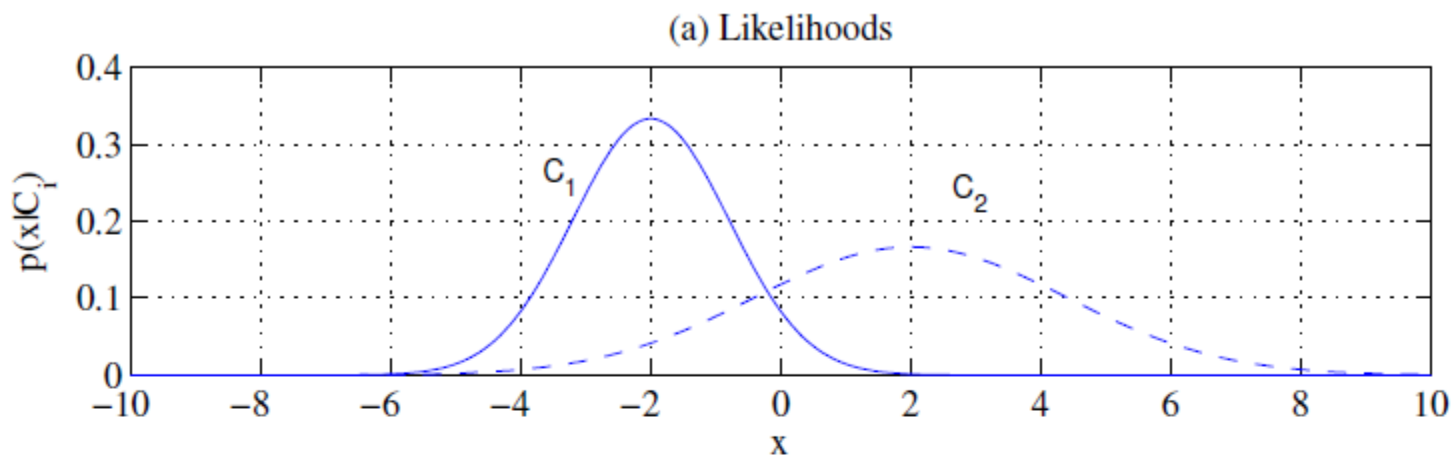


Likelihoods



Posteriors with equal priors





$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

- **Because of normalization by the evidence, the posteriors sum up to 1:**

$$P(C = 0 | x) + P(C = 1 | x) = 1$$

- $p(x | C_i)$ is the probability of seeing x as the input when it is **known to belong to class C_i** . The posterior probability of class C_i can be calculated as

$$P(C_i | x) = \frac{p(x | C_i)P(C_i)}{p(x)} = \frac{p(x | C_i)P(C_i)}{\sum_{k=1}^K p(x | C_k)P(C_k)}$$

- And for **minimum error**, the Bayes' classifier chooses the class with the highest posterior probability; that is, we

$$\text{choose } C_i \text{ if } P(C_i|\mathbf{x}) = \max_k P(C_k|\mathbf{x})$$

- For now, we assume that we know the prior and likelihoods; in later chapters, we discuss how to estimate $P(C)$ and $p(\mathbf{x}|C)$ from a given training sample.

Bayesian Decision Theory: Losses and risks

- It may be the case that decisions are not equally good or costly.
- The loss for a high-risk applicant erroneously accepted may be different from the potential gain for an erroneously rejected low-risk applicant.
- The situation is much more critical and far from symmetry in other domains like medical diagnosis or earthquake prediction.
- Let us define action α_i as the decision to assign the input to class C_i and λ_{ik} as the loss incurred for taking action α_i when the input actually belongs to C_k . Then the **expected risk for taking action α_i** is

$$R(\alpha_i|\mathbf{x}) = \sum_{k=1}^K \lambda_{ik} P(C_k|\mathbf{x})$$

➤ and we choose the action with minimum risk:

$$\text{Choose } \alpha_i \text{ if } R(\alpha_i|\mathbf{x}) = \min_k R(\alpha_k|\mathbf{x})$$

- Let us define K actions α_i , $i = 1, \dots, K$, where α_i is the action of assigning x to C_i .
- In the special case of the 0/1 loss where

$$\lambda_{ik} = \begin{cases} 0 & \text{if } i = k \\ 1 & \text{if } i \neq k \end{cases}$$

- All correct decisions have no loss and all errors are equally costly.
- The risk of taking action α_i is

$$\begin{aligned} R(\alpha_i|\mathbf{x}) &= \sum_{k=1}^K \lambda_{ik} P(C_k|\mathbf{x}) \\ &= \sum_{k \neq i} P(C_k|\mathbf{x}) \\ &= 1 - P(C_i|\mathbf{x}) \end{aligned}$$

Because $\sum_k P(C_k|\mathbf{x}) = 1$.

- Thus to minimize risk, we choose the most probable class.

- Note that In later chapters, for simplicity, we will always assume this case and choose the class with the highest posterior, but this is indeed a special case and rarely do applications have a symmetric, 0/1 loss.
- In the general case, it is a simple post-processing to go from posteriors to risks and to take the action to minimize the risk.

Bayesian Decision Theory Losses and Risks

- Actions: α_i
- Loss of α_i when the state is C_k : λ_{ik}
- Expected risk (Duda and Hart, 1973)

$$R(\alpha_i | \mathbf{x}) = \sum_{k=1}^K \lambda_{ik} P(C_k | \mathbf{x})$$

choose α_i if $R(\alpha_i | \mathbf{x}) = \min_k R(\alpha_k | \mathbf{x})$

Bayesian Decision Theory Losses and Risks: 0/1 Loss

78

$$\lambda_{ik} = \begin{cases} 0 & \text{if } i = k \\ 1 & \text{if } i \neq k \end{cases}$$

$$\begin{aligned} R(\alpha_i | \mathbf{x}) &= \sum_{k=1}^K \lambda_{ik} P(C_k | \mathbf{x}) \\ &= \sum_{k \neq i} P(C_k | \mathbf{x}) \\ &= 1 - P(C_i | \mathbf{x}) \end{aligned}$$

For minimum risk, choose the most probable class

Bayesian Decision Theory: Discriminant functions

- Classification can also be seen as implementing a set of discriminant functions, $g_i(x)$, $i = 1, \dots, K$, such that we

choose C_i if $g_i(x) = \max_k g_k(x)$

- We can represent the Bayes' classifier in this way by setting $g_i(x) = -R(\alpha_i | x)$
- and the maximum discriminant function corresponds to minimum conditional risk.
- When we use the 0/1 loss function, we have $g_i(x) = P(C_i | x)$ or ignoring the common normalizing term, $p(x)$, we can write $g_i(x) = p(x | C_i)P(C_i)$

Bayesian Decision Theory

- This divides the feature space into K decision regions R_1, \dots, R_K , where $R_i = \{x \mid g_i(x) = \max_k g_k(x)\}$.
- The regions are separated by decision boundaries, surfaces in feature space where ties occur among the largest discriminant functions
- When there are two classes, we can define a single discriminant $g(x) = g_1(x) - g_2(x)$ and we

$$\text{choose } \begin{cases} C_1 & \text{if } g(x) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

- An example is a two-class learning problem where the positive examples can be taken as C_1 and the negative examples as C_2 . When $K = 2$, the classification system is a **dichotomizer** and for $K \geq 3$, it is a **polychotomizer**.

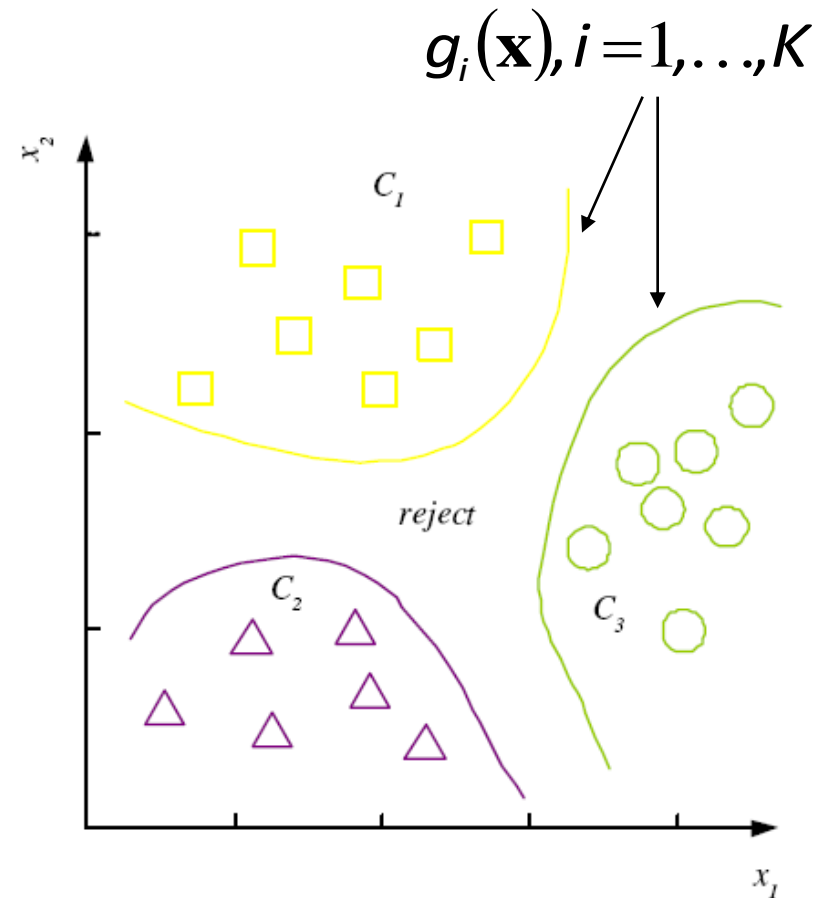
Bayesian Decision Theory Discriminant Functions

choose C_i if $g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})$

$$g_i(\mathbf{x}) = \begin{cases} -R(\alpha_i | \mathbf{x}) \\ P(C_i | \mathbf{x}) \\ p(\mathbf{x} | C_i)P(C_i) \end{cases}$$

K decision regions $\mathcal{R}_1, \dots, \mathcal{R}_K$

$$\mathcal{R}_i = \{\mathbf{x} | g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})\}$$



MACHINE LEARNING

Bayesian Decision Theory $K=2$ Classes

- Dichotomizer ($K=2$) vs Polychotomizer ($K>2$)
- $g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x})$

$$\text{choose } \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

- *Log odds:*

$$\log \frac{P(C_1 | \mathbf{x})}{P(C_2 | \mathbf{x})}$$

Parametric Estimation

- $\mathcal{X} = \{x^t\}_t$ where $x^t \sim p(x)$; samples drawn from distribution $p(x)$
- Parametric estimation:
Assume a form for $p(x | \theta)$ and estimate θ , (its sufficient statistics, using X)
e.g., $N(\mu, \sigma^2)$ where $\theta = \{\mu, \sigma^2\}$

Maximum Likelihood Estimation

- Likelihood of θ given the sample \mathcal{X}

$$l(\vartheta|\mathcal{X}) = p(\mathcal{X}|\vartheta) = \prod_t p(x^t|\vartheta)$$

- Log likelihood

$$\mathcal{L}(\vartheta|\mathcal{X}) = \log l(\vartheta|\mathcal{X}) = \sum_t \log p(x^t|\vartheta)$$

- Maximum likelihood estimator (MLE)

$$\vartheta^* = \operatorname{argmax}_{\vartheta} \mathcal{L}(\vartheta|\mathcal{X})$$

- From Bayes' rule, we can write the posterior probability of class C_i as

$$P(C_i|x) = \frac{p(x|C_i)P(C_i)}{p(x)} = \frac{p(x|C_i)P(C_i)}{\sum_{k=1}^K p(x|C_k)P(C_k)}$$

- and use the discriminant function $g_i(x) = p(x|C_i)P(C_i)$

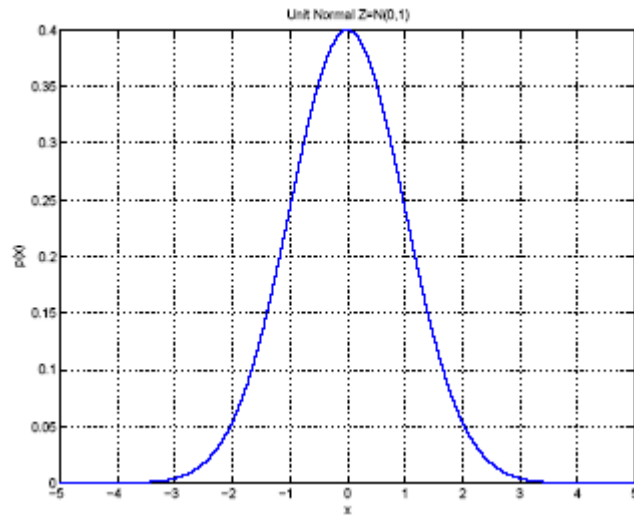
or equivalently

$$g_i(x) = \log p(x|C_i) + \log P(C_i)$$

- If we can assume that $p(x|C_i)$ are Gaussian

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left[-\frac{(x - \mu_i)^2}{2\sigma_i^2} \right]$$

Gaussian (Normal) Distribution



μ σ

$$m = \frac{\sum_t x^t}{N}$$
$$s^2 = \frac{\sum_t (x^t - m)^2}{N}$$

- $p(x) = \mathcal{N}(\mu, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right]$$

- MLE for μ and σ^2 :

MACHINE LEARNING

Bias and Variance

Unknown parameter θ

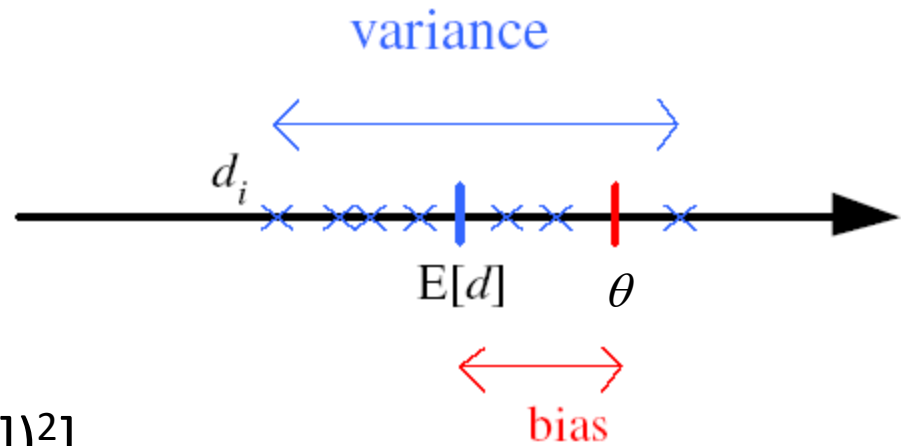
Estimator $d_i = d(X_i)$ on sample X_i

Bias: $b_{\theta}(d) = E[d] - \theta$

Variance: $E[(d - E[d])^2]$

Mean square error:

$$\begin{aligned} r(d, \theta) &= E[(d - \theta)^2] \\ &= (E[d] - \theta)^2 + E[(d - E[d])^2] \\ &= \text{Bias}^2 + \text{Variance} \end{aligned}$$



equation becomes

$$g_i(x) = -\frac{1}{2} \log 2\pi - \log \sigma_i - \frac{(x - \mu_i)^2}{2\sigma_i^2} + \log P(C_i)$$

- Let us see an **example**: Assume we are a car company selling K different cars, and for simplicity, let us say that the sole factor that affects a customer's choice is his or her **yearly income**, which we denote by x.
- Then $P(C_i)$ is the proportion of customers who buy car type i.
- If the yearly income distributions of such customers can be approximated with a Gaussian, then $p(x|C_i)$, the probability that a customer who bought car type i has income x, can be taken as

Bayesian Decision Theory

- , $\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i is the **mean** income of such customers and σ_i^2 is their income **variance**.
- When we do not know $P(C_i)$ and $p(x|C_i)$, we estimate them from a sample and plug in their estimates to get the estimate for the discriminant function.
- We are given a sample $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$
- where $x \in \mathfrak{R}$ is one-dimensional and $r \in \{0, 1\}^K$ such that

$$r_i^t = \begin{cases} 1 & \text{if } x^t \in C_i \\ 0 & \text{if } x^t \in C_k, k \neq i \end{cases}$$

- For each class separately, the estimates for the means and variances are

$$m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t}$$
$$s_i^2 = \frac{\sum_t (x^t - m_i)^2 r_i^t}{\sum_t r_i^t}$$

- and the estimates for the priors are

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N}$$

- Plugging these estimates into equation we get

$$g_i(x) = -\frac{1}{2} \log 2\pi - \log s_i - \frac{(x - m_i)^2}{2s_i^2} + \log \hat{P}(C_i)$$

- The first term is a constant and can be dropped because it is common in all $g_i(x)$.
- If the priors are equal, the last term can also be dropped.
- If we can further assume that variances are equal, we can write

$$g_i(x) = -(x - m_i)^2$$

- and thus we assign x to the class with the nearest mean:

$$\text{Choose } C_i \text{ if } |x - m_i| = \min_k |x - m_k|$$

- With two adjacent classes, the midpoint between the two means is the threshold of decision .

$$g_1(x) = g_2(x)$$

$$\begin{aligned} (x - m_1)^2 &= (x - m_2)^2 \\ x &= \frac{m_1 + m_2}{2} \end{aligned}$$

- When the variances are different, there are two thresholds which can be calculated easily .
- If the priors are different, this has the effect of moving the threshold of decision toward the mean of the less likely class.
- Here we use the maximum likelihood estimators for the parameters but if we have some prior information about them, for example, for the means, we can use a Bayesian estimate of $p(x|C_i)$ with prior on μ_i .
- One **note** of caution is necessary here: When x is continuous, we should not immediately rush to use Gaussian densities for $p(x|C_i)$.

Bayesian Decision Theory

- This is the **likelihood-based approach to classification** where we use **data to estimate the densities** separately, **calculate posterior** densities using Bayes' rule, and then **get the discriminant**.
- In later chapters, we discuss the discriminant based approach where we **bypass the estimation of densities** and directly estimate the discriminants

Bayesian Decision Theory: Regression

- In regression, we would like to write the numeric output, called the dependent variable, as a function of the input, called the independent variable.
- We assume that the numeric output is the sum of a deterministic function of the input and random noise:

$$r = f(x) + \epsilon$$

where **$f(x)$ is the unknown function**, which we would like to **approximate by our estimator, $g(x|\theta)$** , defined up to a set of parameters θ .

- If we assume that **ϵ is zero mean Gaussian** with constant variance σ^2 , namely, $\epsilon \sim N(0, \sigma^2)$, and placing our estimator $g(\cdot)$ in place of the unknown function $f(\cdot)$, we have
- $$p(r|x) \sim \mathcal{N}(g(x|\theta), \sigma^2)$$

- We again use maximum likelihood to learn the parameters θ .
- The pairs (x^t, r^t) in the training set are drawn from an unknown joint probability density $p(x, r)$, which we can write as $p(x, r) = p(r|x)p(x)$
- $p(r|x)$ is the probability of the output given the input, and $p(x)$ is the input density.
- Given an iid sample $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$, the log likelihood

is

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^N p(x^t, r^t) \\ &= \log \prod_{t=1}^N p(r^t|x^t) + \log \prod_{t=1}^N p(x^t)\end{aligned}$$

- We can ignore the second term since it does not depend on our estimator, and we have

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp \left[-\frac{[r^t - g(x^t|\theta)]^2}{2\sigma^2} \right] \\ &= \log \left(\frac{1}{\sqrt{2\pi}\sigma} \right)^N \exp \left[-\frac{1}{2\sigma^2} \sum_{t=1}^N [r^t - g(x^t|\theta)]^2 \right] \\ &= -N \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{t=1}^N [r^t - g(x^t|\theta)]^2\end{aligned}$$

- The first term is independent of the parameters θ and can be dropped, as can the factor $1/\sigma^2$.
- Maximizing this is equivalent to minimizing

$$E(\theta|\mathcal{X}) = \frac{1}{2} \sum_{t=1}^N [r^t - g(x^t|\theta)]^2$$

- Which is the most frequently used error function, and θ that minimize it are called the least squares estimates. This is a transformation frequently done in statistics:
- When the likelihood l contains exponents, instead of maximizing l , we define an error function, **$E = -\log l$** , and minimize it.
- **In linear regression**, we have a **linear model** $g(x^t | w_1, w_0) = w_1 x^t + w_0$ and taking the derivative of the sum of squared errors with respect to w_1 and w_0 , we have two equations in two unknowns

$$\begin{aligned}\sum_t r^t &= Nw_0 + w_1 \sum_t x^t \\ \sum_t r^t x^t &= w_0 \sum_t x_t + w_1 \sum_t (x^t)^2\end{aligned}$$

- which can be written in vector-matrix form as $\mathbf{A}\mathbf{w} = \mathbf{y}$ where

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t \\ \sum_t x^t & \sum_t (x^t)^2 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \end{bmatrix}$$

and can be solved as $\mathbf{w} = \mathbf{A}^{-1}\mathbf{y}$. In the general case of **polynomial regression**, the model is a polynomial in x of order k

$$g(x^t | \mathbf{w}_k, \dots, \mathbf{w}_2, \mathbf{w}_1, \mathbf{w}_0) = \mathbf{w}_k (x^t)^k + \dots + \mathbf{w}_2 (x^t)^2 + \mathbf{w}_1 x^t + \mathbf{w}_0$$

- The model is still linear with respect to the parameters and taking the derivatives, we **get $k + 1$ equations in $k + 1$ unknowns**, which can be written in vector matrix form $\mathbf{A}\mathbf{w} = \mathbf{y}$ where we have

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t & \sum_t (x^t)^2 & \dots & \sum_t (x^t)^k \\ \sum_t x^t & \sum_t (x^t)^2 & \sum_t (x^t)^3 & \dots & \sum_t (x^t)^{k+1} \\ \vdots & & & & \\ \sum_t (x^t)^k & \sum_t (x^t)^{k+1} & \sum_t (x^t)^{k+2} & \dots & \sum_t (x^t)^{2k} \end{bmatrix}$$
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \\ \sum_t r^t (x^t)^2 \\ \vdots \\ \sum_t r^t (x^t)^k \end{bmatrix}$$

We can write $\mathbf{A} = \mathbf{D}^T \mathbf{D}$ and $\mathbf{y} = \mathbf{D}^T \mathbf{r}$ where

$$\mathbf{D} = \begin{bmatrix} 1 & x^1 & (x^1)^2 & \dots & (x^1)^k \\ 1 & x^2 & (x^2)^2 & \dots & (x^2)^k \\ \vdots & & & & \\ 1 & x^N & (x^N)^2 & \dots & (x^N)^k \end{bmatrix}, \mathbf{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

- and we can then solve for the parameters as

$$\mathbf{w} = (\mathbf{D}^T \mathbf{D})^{-1} \mathbf{D}^T \mathbf{r}$$

- Assuming Gaussian distributed error and maximizing likelihood corresponds to minimizing the sum of squared errors.
- Another measure is the relative square error (RSE)

$$E_{RSE} = \frac{\sum_t [r^t - g(x^t | \theta)]^2}{\sum_t (r^t - \bar{r})^2}$$

- A measure to check the goodness of fit by regression is the coefficient of determination that is $R^2 = 1 - E_{RSE}$

- and for regression to be considered useful, we require R^2 to be close to 1
- Remember that for best generalization, we should adjust the complexity of our learner model to the complexity of the data.
- In polynomial regression, the complexity parameter is the order of the fitted polynomial, and therefore we need to find a way to choose the best order that minimizes the generalization error, that is, **tune the complexity of the model to best fit the complexity of the function inherent in the data.**

Tuning model complexity : Bias /Variance dilemma

- Let us say that a sample $X = \{x^t, r^t\}$ is drawn from some unknown joint probability density $p(x, r)$. Using this sample, we construct our estimate $g(\cdot)$. The expected square error (over the joint density) at x can be written as

$$E[(r - g(x))^2 | x] = \underbrace{E[(r - E[r|x])^2 | x]}_{\text{noise}} + \underbrace{(E[r|x] - g(x))^2}_{\text{squared error}}$$

- The first term on the right is the variance of r given x ; it does not depend on $g(\cdot)$ or X . It is the variance of noise added, σ^2 . This is the part of error that can never be removed, no matter what estimator we use.
- The second term quantifies how much $g(x)$ deviates from the regression function, $E[r|x]$. This does depend on the estimator and the training set.

Tuning model complexity : Bias /Variance dilemma

- It may be the case that for one sample, $g(x)$ may be a very good fit; and for some other sample, it may make a bad fit. **To quantify how well an estimator $g(\cdot)$ is, we average over possible datasets.**
- **The expected value** (average over samples X , all of size N and drawn from the same joint density $p(r, x)$) is

$$E_X[(E[r|x] - g(x))^2|x] = \underbrace{(E[r|x] - E_X[g(x)])^2}_{\text{bias}} + \underbrace{E_X[(g(x) - E_X[g(x)])^2]}_{\text{variance}}$$

- **Bias measures** how much $g(x)$ is wrong disregarding the effect of varying samples, and
- **variance measures** how much $g(x)$ fluctuate around the expected value, $E[g(x)]$, as the sample varies.
- We want both to be small.

Tuning model complexity : Bias /Variance dilemma

- As the order of the polynomial increases, small changes in the dataset cause a greater change in the fitted polynomials; **thus variance increases.**
- But a complex model on the average allows a better fit to the underlying function; **thus bias decreases.**
- This is called the bias/variance dilemma and is true for any machine learning system and not only for polynomial regression.
- To decrease bias, the model should be flexible, at the risk of having high variance. If the variance is kept low, we may not be able to make a good fit to data and have high bias. The optimal model is the one that has the best trade-off between the bias and the variance.

Bias variance trade off

- If there is bias, this indicates that our model class does not contain the solution; this is **under fitting**.
- If there is variance, the model class is too general and also learns the noise; this is **over fitting**.
- If $g(\cdot)$ is of the same hypothesis class with $f(\cdot)$, for example, a polynomial of the same order, we have an unbiased estimator, and estimated bias decreases as the number of models increases.

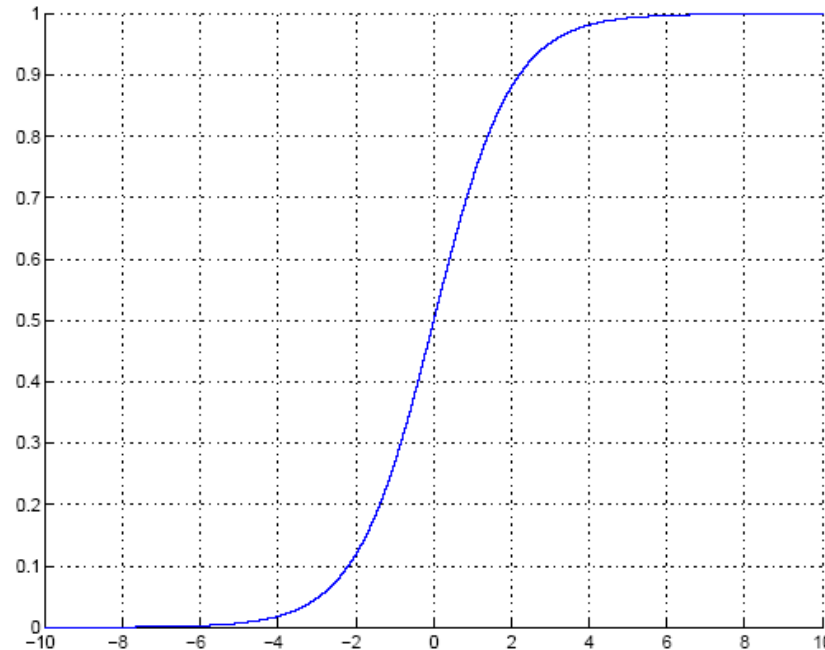
Bayesian Decision Theory

- This shows the error-reducing effect of choosing the right model (which we called inductive bias —the two uses of “bias” are different but not unrelated).
- As for variance, it also depends on the size of the training set; the variability due to sample decreases as the sample size increases.
- **To sum up, to get a small value of error, we should have the proper inductive bias (to get small bias in the statistical sense) and have a large enough dataset so that the variability of the model can be constrained with the data.**

Bayesian Decision Theory

- Note that when the variance is large, bias is low: This indicates that $g(x)$ is a good estimator. So to get a small value of error, we can take a large number of high-variance models and use their average as our estimator.

Sigmoid (Logistic) Function



Calculate $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ and choose C_1 if $g(\mathbf{x}) > 0$, or

Calculate $y = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0)$ and choose C_1 if $y > 0.5$

Gradient-Descent

- $E(\mathbf{w} | X)$ is error with parameters \mathbf{w} on sample X
 $\mathbf{w}^* = \arg \min_{\mathbf{w}} E(\mathbf{w} | X)$

- Gradient

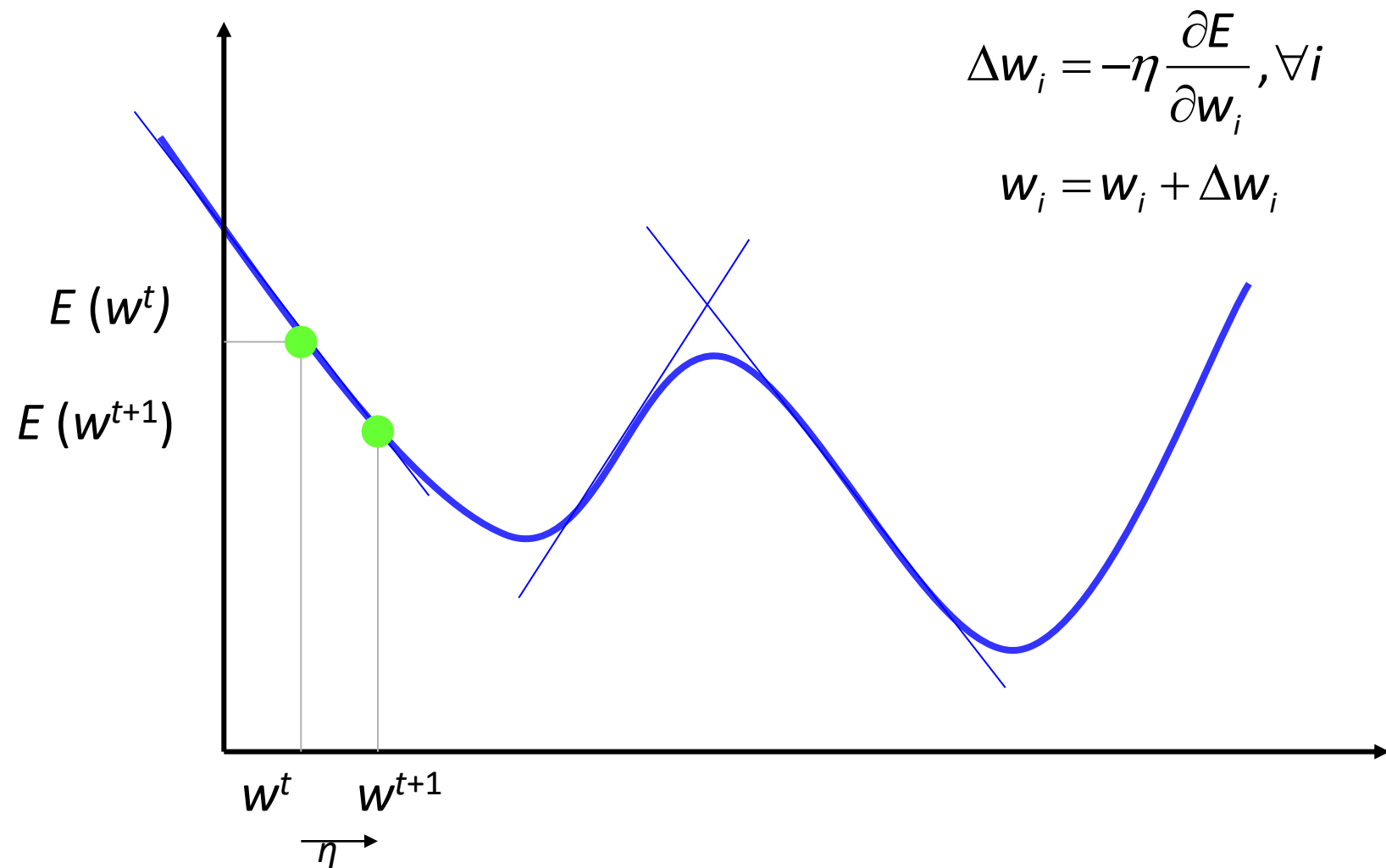
$$\nabla_{\mathbf{w}} E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_d} \right]^T$$

- Gradient-descent:

Starts from random \mathbf{w} and updates \mathbf{w} iteratively in the negative direction of gradient

MACHINE LEARNING

Gradient-Descent



Two classes: Assume log likelihood ratio is linear

$$\log \frac{p(\mathbf{x} | C_1)}{p(\mathbf{x} | C_2)} = \mathbf{w}^T \mathbf{x} + w_0^o$$

$$\begin{aligned} \text{logit}(P(C_1 | \mathbf{x})) &= \log \frac{P(C_1 | \mathbf{x})}{1 - P(C_1 | \mathbf{x})} = \log \frac{p(\mathbf{x} | C_1)}{p(\mathbf{x} | C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \mathbf{w}^T \mathbf{x} + w_0 \end{aligned}$$

$$\text{where } w_0 = w_0^o + \log \frac{P(C_1)}{P(C_2)}$$

$$y = \hat{P}(C_1 | \mathbf{x}) = \frac{1}{1 + \exp \left[-(\mathbf{w}^T \mathbf{x} + w_0) \right]}$$

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_t \quad r^t | \mathbf{x}^t \sim \text{Bernoulli}(y^t)$$

$$y = P(C_1 | \mathbf{x}) = \frac{1}{1 + \exp \left[-(\mathbf{w}^T \mathbf{x} + w_0) \right]}$$

$$l(\mathbf{w}, w_0 | \mathcal{X}) = \prod_t (y^t)^{(r^t)} (1 - y^t)^{(1-r^t)}$$

$$E = -\log l$$

$$E(\mathbf{w}, w_0 | \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

Training: Gradient-Descent

$$E(\mathbf{w}, w_0 | \mathcal{X}) = - \sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

$$\text{If } y = \text{sigmoid}(a) \quad \frac{dy}{da} = y(1 - y)$$

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t \\ &= \eta \sum_t (r^t - y^t) x_j^t, j = 1, \dots, d \end{aligned}$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)$$

K>2 Classes

$$\mathcal{X} = \{\mathbf{x}^t, \mathbf{r}^t\}_t \quad r^t | \mathbf{x}^t \sim \text{Mult}_K(1, \mathbf{y}^t)$$

$$\log \frac{p(\mathbf{x} | C_i)}{p(\mathbf{x} | C_K)} = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

$$y = \hat{P}(C_i | \mathbf{x}) = \frac{\exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}]}{\sum_{j=1}^K \exp[\mathbf{w}_j^T \mathbf{x} + w_{j0}]}, i = 1, \dots, K$$

$$l(\{\mathbf{w}_i, w_{i0}\}_i | \mathcal{X}) = \prod_t \prod_i (y_i^t)^{(r_i^t)}$$

$$E(\{\mathbf{w}_i, w_{i0}\}_i | \mathcal{X}) = - \sum_t r_i^t \log y_i^t$$

$$\Delta \mathbf{w}_j = \eta \sum_t (r_j^t - y_j^t) \mathbf{x}^t \quad \Delta w_{j0} = \eta \sum_t (r_j^t - y_j^t)$$

MACHINE LEARNING

Model



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory



MACHINE LEARNING

Bayesian Decision Theory





THANK YOU

Dr Ajey SNR

Department of ECE

ajeysnr@pes.edu

+91 80 266186626