



# CMU 15-781

Lecture 16:

Machine Learning II

Teacher:

Gianni A. Di Caro

Part of the slides are adapted from Ziko Kolter

# OUTLINE

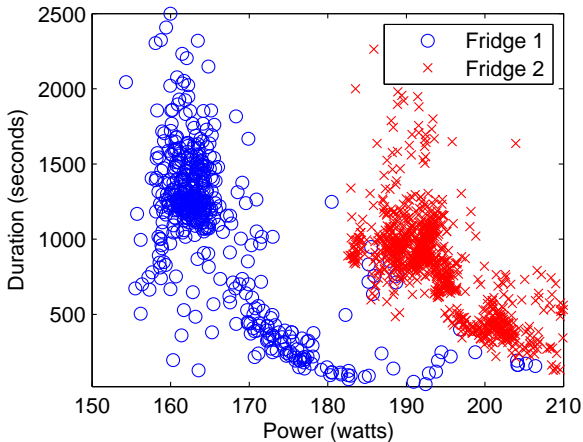
1	Supervised learning: classification .....	2
2	“Non-linear” regression/classification, overfitting, and model selection .....	24
3	PAC learning and generalization bounds .....	47
4	VC-dimension and generalization bounds .....	55

# CLASSIFICATION PROBLEMS

- ▶ Sometimes we want to predict discrete outputs rather than continuous
- ▶ Is the email spam or not? (YES/NO)
- ▶ What digit is in this image? (0/1/2/3/4/5/6/7/8/9)

# EXAMPLE: CLASSIFYING HOUSEHOLD APPLIANCES

- ▶ Differentiate between two refrigerators using their power consumption signatures



# CLASSIFICATION TASKS

- ▶ **Input features:**  $\mathbf{x}^{(i)} \in \mathbb{R}^n$ ,  $i = 1, \dots, m$ 
  - ▶ E.g.:  $\mathbf{x}^{(i)} \in \mathbb{R}^3 = (\text{Duration } i, \text{Power } i, 1)$
- ▶ **Output:**  $y^{(i)} \in \{-1, +1\}$  (binary classification task)
  - ▶ E.g.:  $y^{(i)} = \text{Is it fridge 1?}$
- ▶ **Model Parameters:**  $\boldsymbol{\theta} \in \mathbb{R}^n$
- ▶ **Hypothesis function:**  $h_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ 
  - ▶ Returns *continuous* prediction of the output  $y$ , where the value indicates how “confident” we are that the example is  $-1$  or  $+1$
  - ▶  $\text{sign}(h_{\boldsymbol{\theta}}(\mathbf{x}))$  is the actual binary prediction
  - ▶ We will focus on *linear predictors*  $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta}$

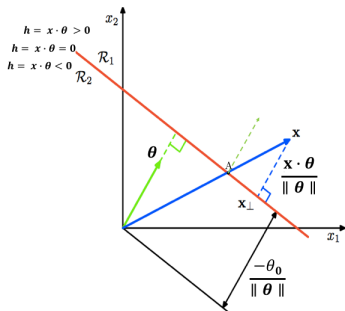
# CLASSIFIER BEHAVIOR: SCORE

- ▶ Given an example  $(\mathbf{x}, y)$ , the value

$$h_{\theta}(\mathbf{x}) = \mathbf{x}^T \theta$$

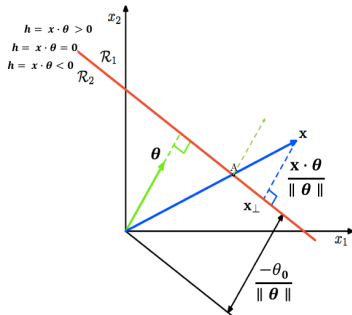
is termed the **score** of the classification performed according to the hypothesis function and current parameters,  $h_{\theta}(\mathbf{x})$

- ▶ Intuitively, the score represents the degree to which the classification is positive or negative, how **confident** the classifier is making the prediction
- ▶ In the context of binary classification with *binary features*  $\mathbf{x}$ , the score has a nice interpretation: it aggregates the contribution of each feature, weighted appropriately. Each feature “votes” on the classification.

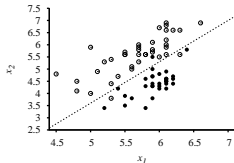
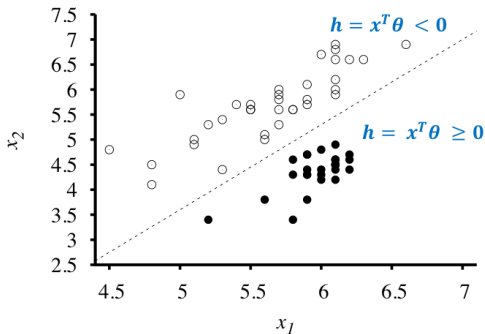


# CLASSIFIER BEHAVIOR: MARGIN

- ▶ The **margin** on an example  $(\mathbf{x}, y)$  is  $(\mathbf{x}^T \boldsymbol{\theta})y$  and represents how **(in)correct** is the classification made by  $\boldsymbol{\theta}$ . The larger the margin, the better.
- ▶ Non-positive margins correspond to *classification errors*
- ▶ Geometrically, if  $\|\boldsymbol{\theta}\| = 1$ , then the margin of an sample input  $\mathbf{x}$  is exactly the signed distance from its feature vector  $h_{\boldsymbol{\theta}}(\mathbf{x})$  to the decision boundary. In the general case, the *distance* of  $x$  from the linear boundary is  $(\mathbf{x}^T \boldsymbol{\theta})/\|\boldsymbol{\theta}\|$
- ▶ **Geometric margin**: actual (signed) distance from a point to decision boundary:  $\frac{(\mathbf{x}^T \boldsymbol{\theta})}{\|\boldsymbol{\theta}\|} y$



# EXAMPLE: SEISMIC DATA

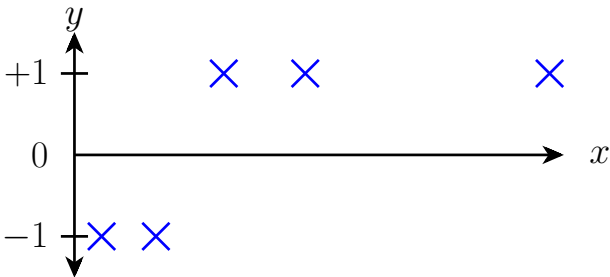


- ▶  $x_1$  = body wave magnitude,  $x_2$  = surface wave magnitude
- ▶ Two features: White = data from earthquakes, black = data from nuclear tests
- ▶ The two classes are **linearly separable**,  
 $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta} = \theta_2 x_2 + \theta_1 x_1 + \theta_0 x_0$  where,  
 $x_0 = 1$ ,  $\theta_0 = -4.9$ ,  $\theta_1 = -1.7$ ,  $\theta_2 = -1$
- ▶ The classification hypothesis is:  $\text{sign}(h_{\boldsymbol{\theta}}(\mathbf{x})) = +1$  if  $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta} \geq 0$ , and -1 otherwise
- ▶ More data, the two classes are non linearly separable: it does not exist a linear decision boundary



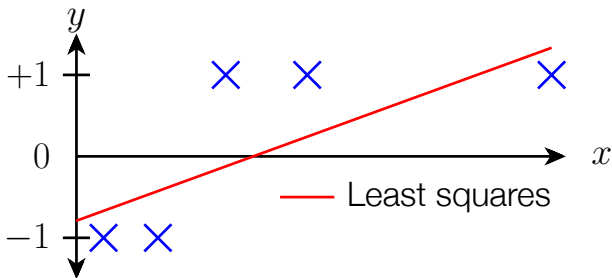
# LOSS FUNCTIONS

- ▶ Example,  $x^{(i)} \in \mathbb{R}$ , one single data feature, binary classification task ( $y^{(i)} \in \{-1, +1\}$ ),  $\theta = [\theta_0, \theta_1]$
- ▶ Linear classifier:  $h_{\theta}(x) = \theta_1 x + \theta_0$
- ▶ Loss function  $\ell : \mathbb{R} \times \{-1, +1\} \rightarrow \mathbb{R}_+$
- ▶ Let's take  $\ell = (\theta_0 + \theta_1 x - y)^2$  and let's find the values for  $\theta$  that minimize the loss over the whole training set



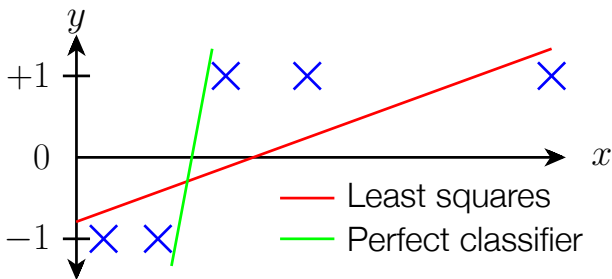
# LOSS FUNCTIONS

- ▶ Example,  $x^{(i)} \in \mathbb{R}$ , one single data feature, binary classification task ( $y^{(i)} \in \{-1, +1\}$ ),  $\theta = [\theta_0, \theta_1]$
- ▶ Linear classifier:  $h_{\theta}(x) = \theta_1 x + \theta_0$
- ▶ Loss function  $\ell : \mathbb{R} \times \{-1, +1\} \rightarrow \mathbb{R}_+$
- ▶ Let's take  $\ell = (\theta_0 + \theta_1 x - y)^2$  and let's find the values for  $\theta$  that minimize the loss over the whole training set
- ▶ Do we need a different loss function?



# LOSS FUNCTIONS

- ▶ Example,  $x^{(i)} \in \mathbb{R}$ , one single data feature, binary classification task ( $y^{(i)} \in \{-1, +1\}$ ),  $\theta = [\theta_0, \theta_1]$
- ▶ Linear classifier:  $h_{\theta}(x) = \theta_1 x + \theta_0$
- ▶ Loss function  $\ell : \mathbb{R} \times \{-1, +1\} \rightarrow \mathbb{R}_+$
- ▶ Let's take  $\ell = (\theta_0 + \theta_1 x - y)^2$  and let's find the values for  $\theta$  that minimize the loss over the whole training set
- ▶ Do we need a different loss function?

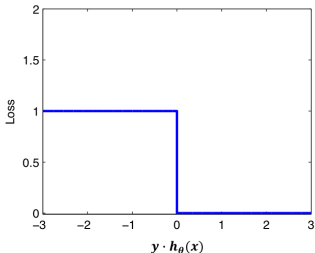


# ACCURACY LOSS FUNCTION

- ▶ Instead of measuring squared deviations, let's measure the **accuracy** of classification
- ▶ Simplest way to measure loss as (0/1) accuracy is: *count the number of mistakes*
- ▶ Again, let's assume a linear hypothesis function, but this time  $\text{sign}(h_\theta(x))$  is used

$$\begin{aligned}\ell(h_\theta(x), y) &= \begin{cases} 1 & \text{if } y \neq \text{sign}(h_\theta(x)) \\ 0 & \text{otherwise} \end{cases} \\ &= \mathbf{1}\{y \cdot h_\theta(x) \leq 0\}\end{aligned}$$

If  $y^{(i)}$  and  $h_\theta(x^{(i)})$  have discording sign (i.e., they disagree on classifying sample  $x^{(i)}$ ), then their product (i.e., the *margin*) is negative and  $\{y^{(i)} \cdot h_\theta(x^{(i)}) \leq 0\}$  equals to 1  $\rightarrow$  add error count



$$\text{minimize}_{\theta} \sum_{i=1}^m \ell(h_\theta(x^{(i)}), y^{(i)})$$

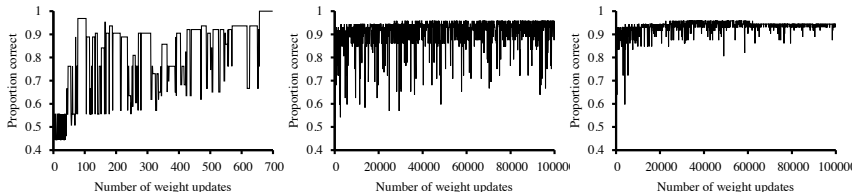
# MINIMIZATION OF ACCURACY LOSS?

- ▶ Unfortunately, minimizing sum of 0/1 losses leads to a hard optimization problem because of the characteristics of the step/threshold loss function in the space of the parameters  $\theta$
- ▶ Trying to apply an analytic method fails, as well as applying gradient descent, since the gradient of  $\ell$  is zero almost everywhere, except at the step point, where  $\theta \cdot \mathbf{x} = 0$ , and the gradient is undefined
- ▶ However, given that the problem is linearly separable, a simple weight update rule exists that converges to the *optimal linear separation*
- ▶ The rule is called the **perceptron learning rule**. For a single training example  $(\mathbf{x}^{(i)}, y^{(i)})$ :

$$\theta_k \leftarrow \theta_k + \alpha(y^{(i)} - h_{\theta}(\mathbf{x}^{(i)}))x_k^{(i)}$$

- ▶ Typically the rule is applied one example at-a-time, choosing the examples at random, similarly to stochastic gradient descent
- ▶ Perceptron learning rule is analogous to SGD for linear regression and squared losses
- ▶ If the problem is *not* linearly separable, perceptron update rule may not converge. However, convergence is guaranteed if  $\alpha$  decays as  $O(1/t)$  and data are presented randomly.

# PERCEPTRON RULE PERFORMANCE



- ▶ Left: earthquake, separable data set
- ▶ Middle: earthquake, non-separable data set
- ▶ Right: earthquake, non-separable data set,  $\alpha(t) = 1000/(1000 + t)$
- ▶ Observation: Quite hectic and unpredictable behavior

# SMOOTHING ACCURACY LOSS

- ▶ Because of the difficulties solving the minimization problem with 0/1, a whole range of alternative “approximations” to 0/1 loss are used instead
- ▶ The hard, non-differential step is replaced by a continuous, differentiable function

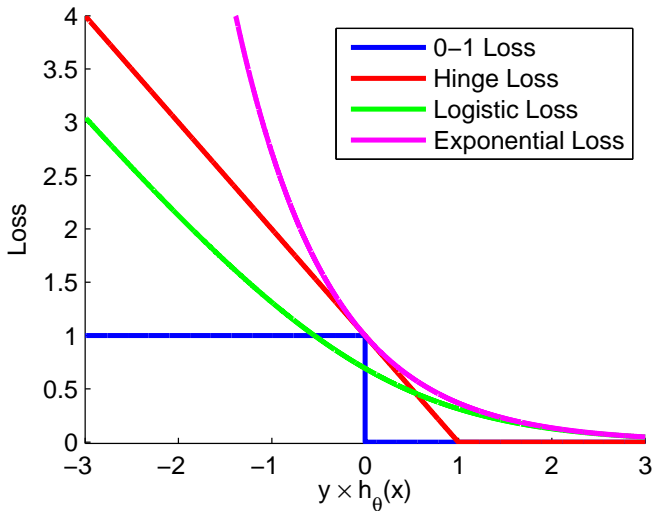
$$\text{Hinge loss: } \ell(h_\theta(x), y) = \max\{1 - y \cdot h_\theta(x), 0\}$$

$$\text{Squared hinge loss: } \ell(h_\theta(x), y) = \max\{1 - y \cdot h_\theta(x), 0\}^2$$

$$\text{Logistic loss: } \ell(h_\theta(x), y) = \log(1 + e^{-y \cdot h_\theta(x)})$$

$$\text{Exponential loss: } \ell(h_\theta(x), y) = e^{-y \cdot h_\theta(x)}$$

# SMOOTHING ACCURACY LOSS

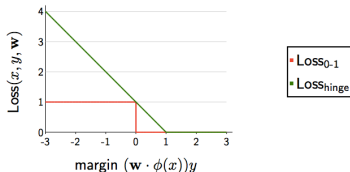


Common loss functions for classification. x-axis: Margin!



# HINGE LOSS FUNCTION (SVMs)

$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{1 - (\mathbf{w} \cdot \phi(x))y, 0\}$$



- **Intuition:** hinge loss upper bounds 0-1 loss, has non-trivial gradient
- Try to increase margin if less than 1

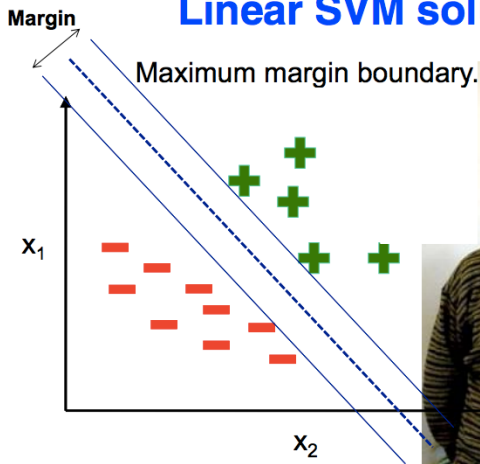
- ▶ Loss = 0 is equivalent to the margin being at least 1
- ▶ **Support Vector Machine (SVM):** hinge loss + *regularization penalty*, linear prediction

$$\text{minimize}_{\theta} \sum_{i=1}^m \max\{1 - y^{(i)} \cdot x^{(i)T} \theta, 0\} + \lambda \sum_{i=1}^n \theta_i^2$$

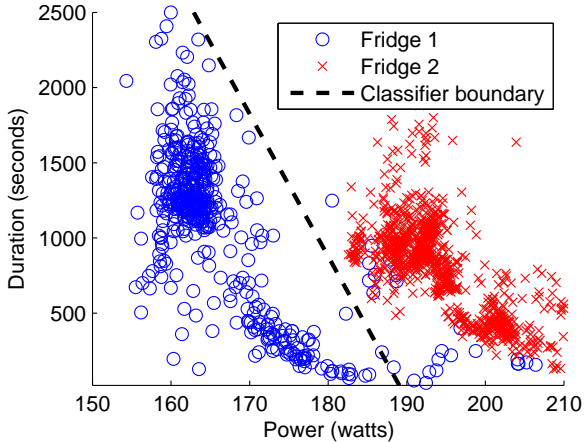
- ▶ The regularization penalty is to avoid that parameters get too high values → make hypothesis simple → here “simplicity” is related to the length of  $\theta$
- ▶ Geometric interpretation: Loss = 0 → (Margin)  $(\mathbf{x}^T \theta)y \geq 1$  → (Geometric margin)  $\|\theta\|^{-1}(\mathbf{x}^T \theta)y \geq \|\theta\|^{-1}$  → Keeping  $\|\theta\|$  small is a way to increase the geometric margin
- ▶ Lagrangian interpretation in terms of constraints on the  $\theta$ s

# MAX MARGIN SVM

## Linear SVM solution



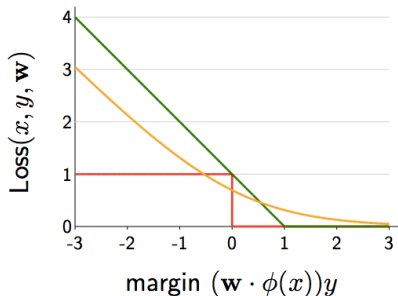
Slide from Quaid Morris



Classification boundary of support vector machine

# LOGISTIC REGRESSION

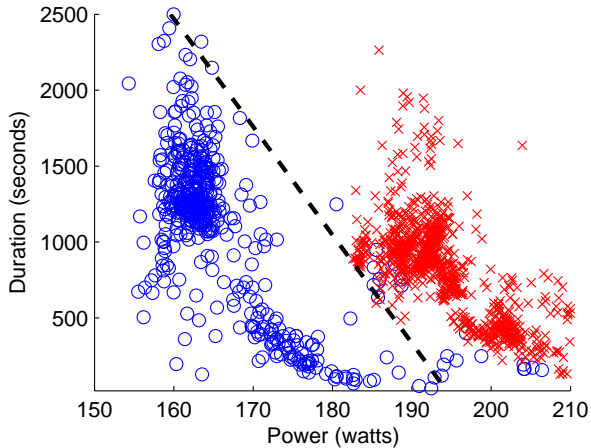
$$\text{Loss}_{\text{logistic}}(x, y, \mathbf{w}) = \log(1 + e^{-(\mathbf{w} \cdot \phi(x))y})$$



- ▶ Logistic regression uses logistic loss

$$\underset{\theta}{\text{minimize}} + \sum_{i=1}^m \log(1 + e^{-y \cdot x^{(i)T} \theta}) + \lambda \sum_{i=1}^n \theta_i^2$$

- ▶ Intuition: Try to increase margin even when it already exceeds 1
- ▶ No matter how correct you are predicting, you will have non-zero loss, and so there is still an incentive (although a diminishing one) to push the margin even larger.
- ▶ Every single example results in an update of the parameters  $\theta$
- ▶ Again, gradient descent is a reasonable algorithm



Classification boundary of logistic regression

# PROBABILISTIC INTERPRETATION OF LOGISTIC REGRESSION

- ▶ Like least squares, logistic regression has a probabilistic interpretation
- ▶ For binary classification problem, suppose that

$$p(y|x; \theta) = \frac{1}{1 + \exp(-y \cdot h_{\theta}(x))}$$

and for each data point  $x^{(i)}$ ,  $y^{(i)}$  is sampled randomly from this distribution

- ▶ Then

$$\begin{aligned} \underset{\theta}{\text{minimize}} & - \sum_{i=1}^m \log p(y^{(i)} | x^{(i)}; \theta) \\ \equiv \underset{\theta}{\text{minimize}} & \log \left( 1 + \exp \left( -y^{(i)} \cdot h_{\theta}(x^{(i)}) \right) \right) \end{aligned}$$

# MULTI-CLASS CLASSIFICATION

- ▶ When classification is not binary  $y \in 0, 1, \dots, k$  (i.e., classifying digit images), a common approach is **one-vs-all** method
- ▶ Create a new set of  $y$ 's for the binary classification problem “is the label of this example equal to  $j$ ”?

$$y^{(i)} = \begin{cases} 1 & \text{if } y^{(i)} = j \\ -1 & \text{otherwise} \end{cases}$$

and solve for the corresponding parameter  $\theta^j$

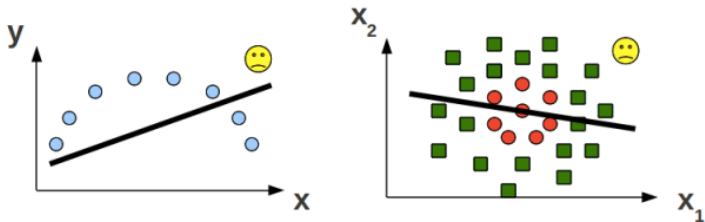
- ▶ For an input  $\mathbf{x}$ , classify according to the hypothesis with the highest confidence:  
 $\operatorname{argmax}_j h_{\theta^j}(\mathbf{x})$

# OUTLINE

1	Supervised learning: classification .....	2
2	“Non-linear” regression/classification, overfitting, and model selection .....	24
3	PAC learning and generalization bounds .....	47
4	VC-dimension and generalization bounds .....	55



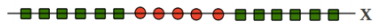
# NON-LINEAR REGRESSION / CLASSIFICATION



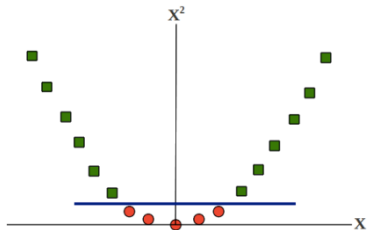
Figures from Piyush Rai

# IDEA OF KERNEL METHODS

- ▶ Map data to higher dimensions where it exhibits linear patterns
- ▶ Apply the linear model in the new input space
- ▶ Mapping = changing the feature representation
- ▶ Linear classifiers/regressors can be used!



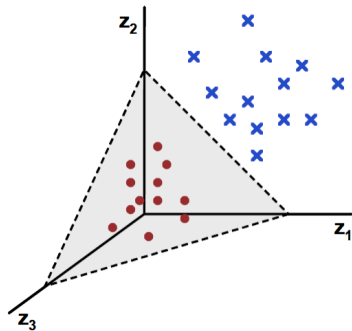
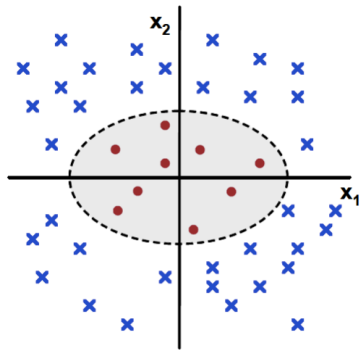
- ▶ Each example represented by a single feature  $x$
- ▶ No linear separator exists for this data



- ▶ Map each example as  $x \rightarrow \{x, x^2\}$
- ▶ Each example now has *two* features (“derived” from the old representation)
- ▶ Data becomes linearly separable in the new representation

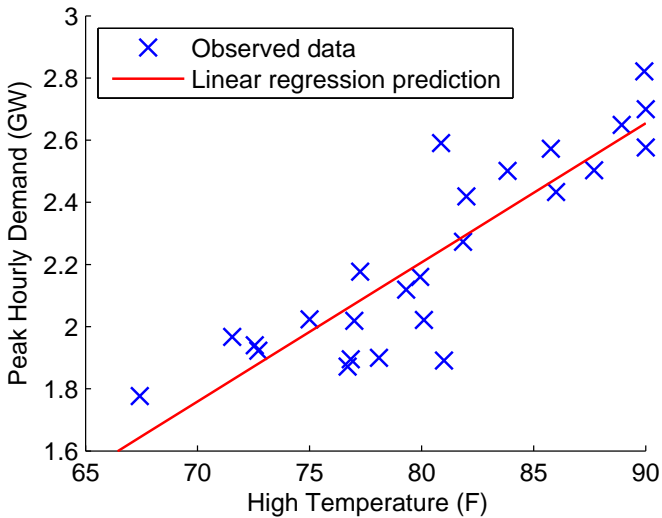
Figures from Piyush Rai

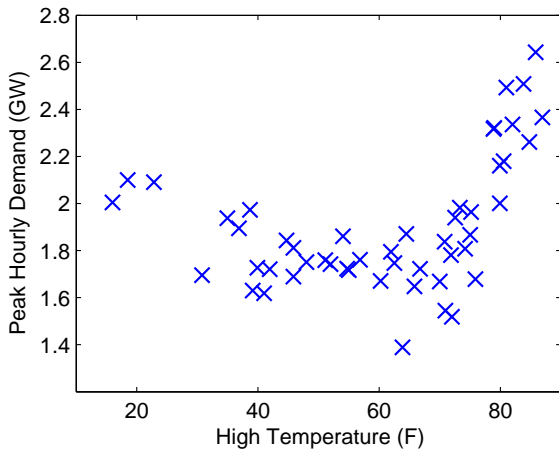
# NON-LINEAR FEATURE TRANSFORMATION



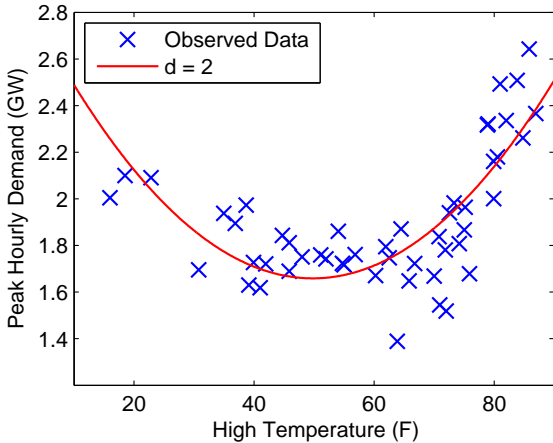
- ▶ Each example represented by two features  
 $\mathbf{x} = \{x_1, x_2\}$

- ▶ Map each example as  
 $\mathbf{x} = \{x_1, x_2\} \rightarrow \mathbf{z} = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
- ▶ Each example now has *three* features

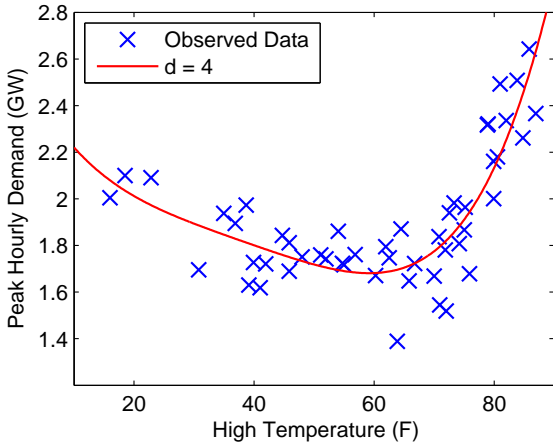




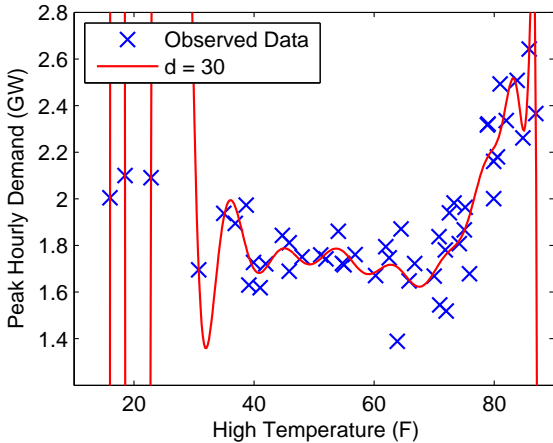
Several days of peak demand vs. high temperature in Pittsburgh over all months



Linear regression with second degree polynomial features



Linear regression with fourth degree polynomial features



Linear regression with 30th degree polynomial features



# OVERFITTING

- ▶ We can either transform feature spaces and use linear hypothesis, or take the feature spaces as they are given and use non-linear hypothesis
- ▶ In both cases the *complexity* of our hypothesis will depend on the number of parameters and on the functional of the parametric hypothesis function
- ▶ Given the training data, we can always make the hypothesis more and more “complex” in order to fit the data better and better
- ▶ To which extent should we push this way of proceeding?
- ▶ This would guarantee that the loss on the training data would get smaller and smaller
- ▶ Is this that we are aiming to?

# GENERALIZATION LOSS AND EMPIRICAL LOSS

- ▶ *Fundamental problem:* we are looking for parameters that optimize

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

but what we really care about is loss of prediction on *new* examples  $(x', y')$   
→ **Generalization error**)

- ▶ This is the expected loss over all input-output pairs the learning machine will see ...
- ▶ To quantify this expectation, we need to define a **prior probability distribution**  $P(\mathbf{X}, Y)$  **over the examples**, which we assume as **stationary** ( $P$  doesn't change)
- ▶ The **expected generalization loss** is:

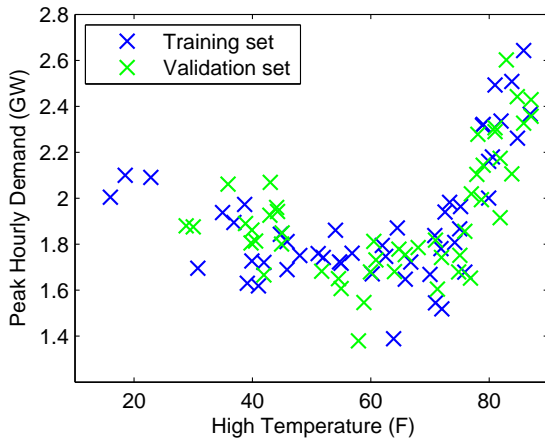
$$L_{gen} = \sum_{i=1}^M \ell(h_{\theta}(x^{(i)}), y^{(i)}) P(x^{(i)}, y^{(i)})$$

- ▶ But  $P(\mathbf{X}, Y)$  is not known, therefore it is only possible to estimate the generalization loss with the **empirical loss** on a set of examples  $m \ll M$ :

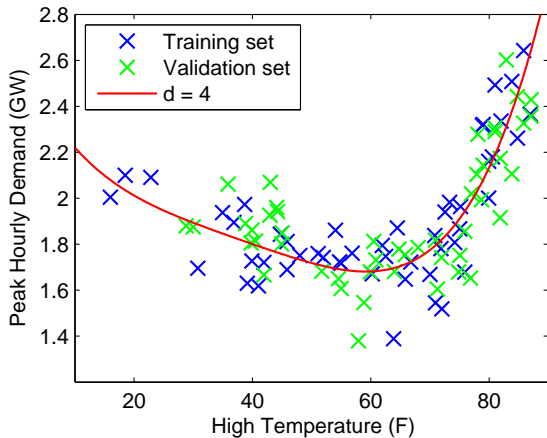
$$L_{emp} = \frac{1}{m} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

# TRAINING AND VALIDATION LOSS

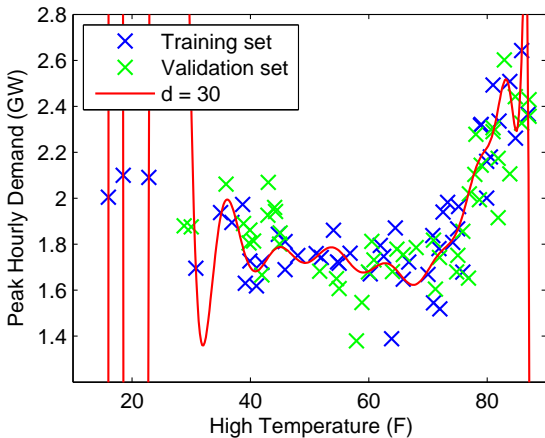
- ▶ **Fundamental problem:** we are looking for parameters that optimize the generalization loss, using the empirical loss
- ▶ Solving the minimization problems for the empirical loss not necessarily brings the same optimal generalization loss because of:
  - ▶ *unrealizability*: the true hypothesis is not included in the considered universe
  - ▶ *variance*: resulting from sampling different subsets of the possible data
  - ▶ *noise*: predictions can differ for the same samples
  - ▶ *computational complexity*: it might not be feasible to solve the problem to optimality
- ▶ Divide data into **training set** (used to find parameters for a fixed hypothesis class  $h_\theta$ ), and **validation set** (used to choose hypothesis class)
- ▶ What is the negative effect of doing this?



Training set and validation set



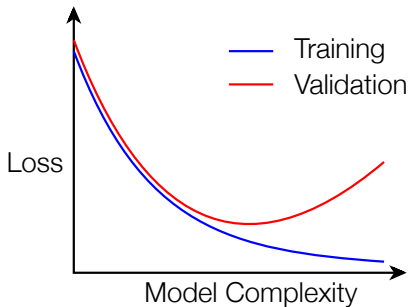
Training set and validation set, fourth degree polynomial



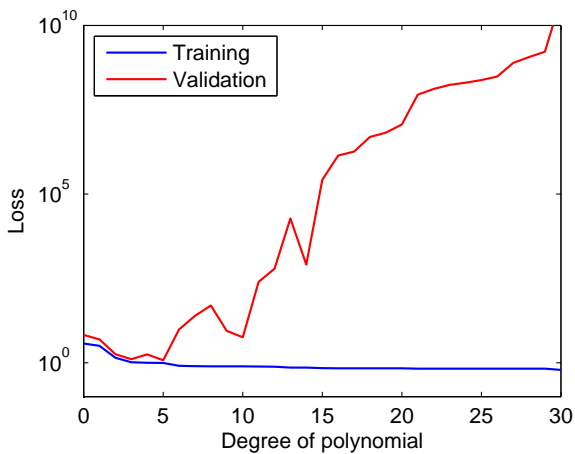
Training set and validation set, 30th degree polynomial: the the loss looks like?

# TRAINING VS. VALIDATION LOSS

- ▶ General intuition for training and validation loss



- ▶ We would like to choose hypothesis class that is at the “sweet spot” of minimizing validation loss



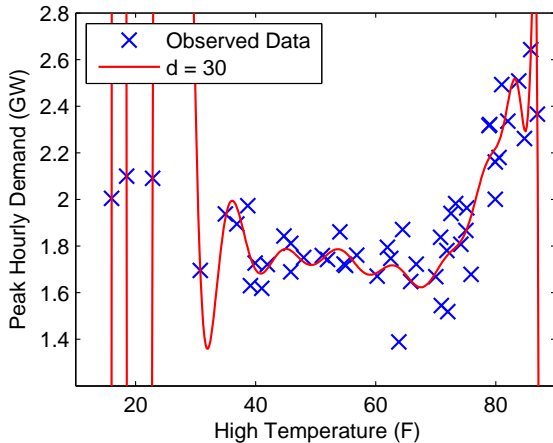
Training and validation loss on peak demand prediction



# MODEL COMPLEXITY AND REGULARIZATION

- ▶ A number of different ways to control “model complexity”
- ▶ An obvious one we have just seen: keep the number of features (number of parameters) low
- ▶ A less obvious method: keep the *magnitude* of the parameters small

- Intuition: a 30th degree polynomial that passes exactly through many of the data points requires very large entries in  $\theta$



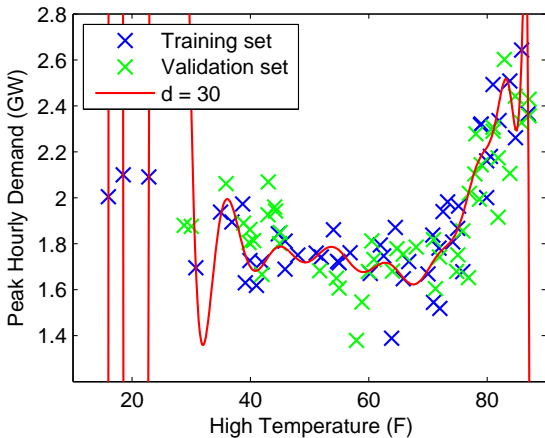
# REGULARIZED LOSS MINIMIZATION PROBLEM

- ▶ We can directly prevent large entries in  $\theta$  by penalizing the magnitude of its entries
- ▶ Leads to **regularized loss minimization** problem

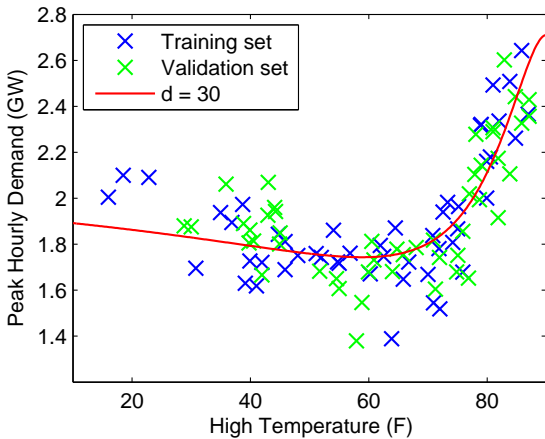
$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell \left( h_{\theta}(x^{(i)}), y^{(i)} \right) + \lambda \sum_{i=1}^n \theta_i^2$$

where  $\lambda \in \mathbb{R}_+$  is a *regularization parameter* that weights the relative penalties of the size of  $\theta$  and the loss

- ▶ Think about imposing a constraint on each parameter  $\rightarrow$  Lagrange multipliers



Degree 30 polynomial, with  $\lambda = 0$  (unregularized)



Degree 30 polynomial, with  $\lambda = 1$

# EVALUATING ML ALGORITHMS

- ▶ The proper way to evaluate an ML algorithm (e.g., look at Cross-Validation):
  - 1 Break all data into training/testing sets (e.g., 70%/30%)
  - 2 Break training set into training/validation set (e.g., 70%/30% again)
  - 3 Choose hyperparameters using validation set
  - 4 (Optional) Once we have selected hyperparameters, retrain using all the training set
  - 5 Evaluate performance on the testing set

# OUTLINE

1	Supervised learning: classification .....	2
2	“Non-linear” regression/classification, overfitting, and model selection .....	24
3	PAC learning and generalization bounds .....	47
4	VC-dimension and generalization bounds .....	55

# USEFUL BOUNDS?

- ▶ **Generalization bounds (how good?):**

Given a finite amount of data, if we learn a classifier, can we have a guarantee on how well that classifier will do on future data?

- ▶ **Sample complexity (how many samples?):**

If we require a bound on the classifier accuracy on future data points, can we bound how many training samples we need to get such a classifier?

- ▶ **Probably Approximately Correct (PAC) model of learning:**

Any hypothesis that is seriously wrong will almost certainly be “found out” with high probability after a small number of examples, because it will make incorrect predictions.

- ▶ → Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong
  - ▶ → The (learned) hypothesis must be **probably approximately correct**
- ▶ A PAC learning model provides guarantees on the accuracy of the learning machine when generalizing from the training examples



# PAC MODEL

- ▶ **Input feature space  $X$**
- ▶  $P(X, Y)$ , **distribution over  $X$** : unknown but fixed and stationary
- ▶ **Hypothesis space** (or *Concept class*)  $\mathcal{H}$  of functions (also called *concepts*)  
 $h : X \mapsto \{0, 1\}$
- ▶ A *target function* is given for the examples,  $y_t \in \mathcal{H}$
- ▶ Samples are *independent and identically distributed*, according to  $P$
- ▶ **Training set of examples**  $Z = \{(\mathbf{x}_i, y_t(\mathbf{x}_i)), i = 1, \dots, m\}$

# PAC MODEL

- ▶ **Error rate of a hypothesis**  $h$ , which is the same as the *expected generalization error*:

$$err(h) = Pr_{P(\mathbf{X}, Y)}[\mathbf{x} : h(\mathbf{x}) \neq y_t(\mathbf{x})]$$

$err(h)$  is the probability that  $h$  misclassifies a new example, that is, the expected proportion of mistakes the hypothesis will make

- ▶ **Accuracy**,  $\epsilon > 0$ :

We would like to find a hypothesis  $h$  with

$$err(h) \leq \epsilon$$

Such a hypothesis is called *approximately correct*, meaning that it's “close” to the real target function. Technically, it is located inside the  $\epsilon$ -ball around the true hypothesis function. In the hypothesis space  $\mathcal{H}$ , let's indicate with  $\mathcal{H}_{PAC}$  this ball, and with  $\mathcal{H}_{bad}$  everything that lies outside of the  $\epsilon$ -ball (the “seriously wrong” hypotheses)

- ▶ **Confidence**,  $\delta > 0$ :

We would like to achieve  $Pr[err(h) \leq \epsilon] \geq 1 - \delta$

# PAC LEARNING ALGORITHM

▶ **PAC Learning algorithm/machine/agent  $L$ :**

A function  $L : Z \mapsto \mathcal{H}$  from the training examples to  $\mathcal{H}$ , such that for every  $\epsilon, \delta > 0$  there exists a number  $m_0(\epsilon, \delta)$  such that for every  $m \geq m_0$  and every  $\mathbf{X}$ , if  $m$  examples  $Z$  are drawn from  $P(\mathbf{X}, Y)$  then:

$$Pr[err(h) \geq \epsilon] \leq 1 - \delta,$$

where  $h \in \mathcal{H}$  is the hypothesis learned by  $L$  according to  $Z$ , that is,  $L(Z) = h$

- ▶ The hypothesis space  $\mathcal{H}$  is (PAC-)learnable if there is a learning algorithm  $L$  for  $\mathcal{H}$
- ▶ In other words, if a learning algorithm  $L$  returns a hypothesis  $h$  that is **consistent** with at least  $m_0$  examples (i.e., it classifies them correctly) then with a probability of at least  $1 - \delta$ ,  $L$  has a generalization error of at most  $\epsilon \rightarrow$  It is PAC! ( $h$  lies in  $\mathcal{H}_{PAC}$ )
- ▶ The number  $m_0(\epsilon, \delta)$  of required samples is the **sample complexity** of the hypothesis space, and is **independent of  $P$** . It depends on the characteristics of  $\mathcal{H}$  and on  $\mathbf{X}$
- ▶ Unfortunately this bound is typically very large :(

# VALUE OF THE SAMPLE COMPLEXITY

- ▶ All hypotheses in  $\mathcal{H}_{bad}$  are such that the expected generalization error is greater than  $\epsilon$ :  $err(h) > \epsilon, \forall h \in \mathcal{H}_{bad}$
- ▶ What is the probability that a hypothesis  $h \in \mathcal{H}_{bad}$  is consistent with the first  $m_0$  samples? (i.e., it classifies them correctly?)
- ▶ By definition,  $err(h) > \epsilon \Rightarrow$  The probability that  $h$  agrees with a given single example  $\mathbf{x}_i$  is *at most*  $1 - \epsilon$ :  $Pr_P[\mathbf{x}_i : h(\mathbf{x}_i) = y_t(\mathbf{x}_i)] \leq 1 - \epsilon$
- ▶ Since all examples are independent, the bound on  $h$  being consistent with a set  $Z$  of  $m_0$  examples is:

$$Pr_P[\mathbf{x}_i, i = 1, \dots, m_0 : h(\mathbf{x}_i) = y_t(\mathbf{x}_i)] \leq (1 - \epsilon)^{m_0}$$

- ▶ The probability that  $\mathcal{H}_{bad}$  contains at least *one* consistent hypothesis is bounded by the sum of the individual probabilities of each hypothesis in  $\mathcal{H}_{bad}$ , therefore, from the previous relation:

$$P(\mathcal{H}_{bad} \text{ contains a consistent hypothesis}) \leq |\mathcal{H}_{bad}|(1 - \epsilon)^{m_0} \leq |\mathcal{H}|(1 - \epsilon)^{m_0} \leq |\mathcal{H}|e^{-\epsilon m_0}$$

where the last inequality derives from a general mathematical bound

# VALUE OF THE SAMPLE COMPLEXITY

- ▶ We aim to make the probability  $P(\mathcal{H}_{bad} \text{ contains a consistent hypothesis})$  being less than a small positive number  $\delta$ :

$$|\mathcal{H}|e^{-\epsilon m_0} \leq \delta$$

- ▶ In fact, when this is true, the probability that a hypothesis  $h$  is an inconsistent one (i.e., belongs to  $\mathcal{H}_{bad}$ ) after being consistent on  $m_0$  samples is less than  $\delta$ .
- ▶ In other words, with a probability  $1 - \delta$  the machine  $L$  returns a hypothesis  $h$  that has an expected error rate of at most  $\epsilon$
- ▶ The required number  $m_0$  of samples that guarantees PAC learning when  $h$  shows consistency on all the  $m_0$  samples, is found by solving wrt to  $m_0$  the inequality  $|\mathcal{H}|e^{-\epsilon m_0} \leq \delta$  using the logarithms:

$$m_0 \geq \frac{1}{\epsilon} \left( \log \frac{1}{\delta} + \log |\mathcal{H}| \right)$$

- ▶  $m_0$  is the **sample complexity** of the hypothesis space  $\mathcal{H}$
- ▶ The sample complexity is  $\propto$  to the log of cardinality of the hypothesis space, such that it explodes for large numerable spaces and it is not defined for infinite spaces
- ▶ The concept of **VC-dimension** extends the notions of analysis of expected the generalization errors to hypothesis sets of infinite cardinality and provides tighter bounds. It provides a substitute for the term  $\log |\mathcal{H}|$

# VALUE OF THE SAMPLE COMPLEXITY

- ▶ If  $\mathcal{H}$  is the set of all Boolean functions  $h : \mathbf{X} \mapsto \{0, 1\}$  and  $\mathbf{X}$  is an  $n$ -dimensional feature space, then  $|\mathcal{H}| = 2^{2^n}$  (all possible mappings from  $n$  inputs to 2 outputs)  
⇒  $m_0 = O(2^n)$ , it grows exponentially with the number of the input features  
→ It is necessary to see almost ALL the possible examples!
- ▶ Intuitively: since  $\mathcal{H}$  contains all generic mappings  $h$ , then, for any set  $Z$  of  $m$  examples, the set of hypothesis consistent with  $Z$  contains equal numbers of hypotheses that would classify a new example  $\mathbf{x}_{m+1}$  as 0 and as 1  
→ to assess something about generalization we would need to see all the possible examples.
- ▶ To obtain useful results for generalization to new, unseen examples, one way is to restrict  $\mathcal{H}$ , avoiding to be too general. However, this might remove the possibility to have PAC learning, since  $\mathcal{H}$  might not include any feasible hypothesis function in the  $\epsilon$ -ball.

# OUTLINE

1	Supervised learning: classification .....	2
2	“Non-linear” regression/classification, overfitting, and model selection .....	24
3	PAC learning and generalization bounds .....	47
4	VC-dimension and generalization bounds .....	55

# BETTER BOUNDS USING VC-DIMENSION

This is what we will obtain using the VC-dimension...

$$\text{test error} \leq \text{training error} + \sqrt{\frac{C \left( \log \left( \frac{2M}{C} \right) + 1 \right) - \log \left( \frac{\eta}{4} \right)}{M}}$$

Gives Upper Bound of Test Error with probability  $1 - \eta$

M=Number of Training Samples

C is related to capacity of machine and is called Vapnik-Chervonenkis (VC) Dimension

**test error**  $\leq$  **training error** + **penalty (complexity)**



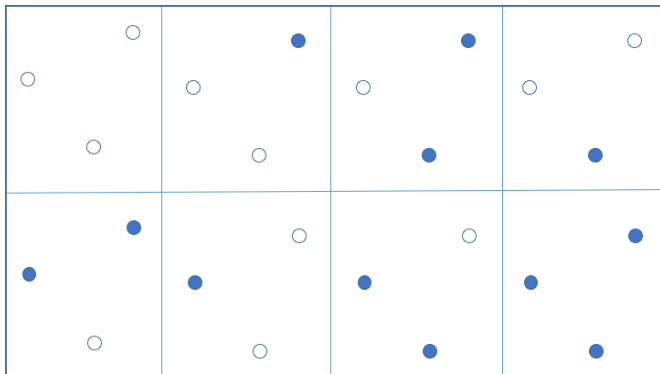


# DEFINITIONS: SET SHATTERING AND VC-DIMENSION

- ▶ **Shattering:** Let  $\mathcal{H}$  be a hypothesis space (also called a **concept class**) defined over an instance (feature) space  $\mathbf{X}$ . Let  $Z = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subseteq \mathbf{X}$  a subset (of examples) from the instance space. The concept class  $\mathcal{H}$  *shatters* (“to break into pieces”)  $Z$  if every possible function on  $Z$  can be represented by some  $h \in \mathcal{H}$ . A function on  $Z$  is a *mapping* from an input  $\mathbf{x}$  to an output  $y$ .
- ▶ Restricting the reasoning that follow to *binary classification tasks* where  $y \in \{0, 1\}$ , we can rephrase it saying that **a set of instances  $Z$  is shattered by  $\mathcal{H}$  if for any binary labeling of the elements in  $Z$  there is a consistent hypothesis in  $\mathcal{H}$**  (i.e., there is a choice of the learning parameters  $\theta$  such that the training error goes to zero).
- ▶ The number of possible mappings (i.e., binary labelings) on  $Z$  is equal to  $2^{|Z|}$
- ▶ **Vapnik-Chervonenkis (VC) dimension:**  $VC(\mathcal{H})$  is the maximum number of points that can be shattered by  $\mathcal{H}$  (i.e., the maximum cardinality of a set shattered by  $\mathcal{H}$ ). The VC-dimension is  $\infty$  if the maximum does not exist.

# SHATTERING 3 POINTS IN $\mathbb{R}^2$ WITH CIRCLES

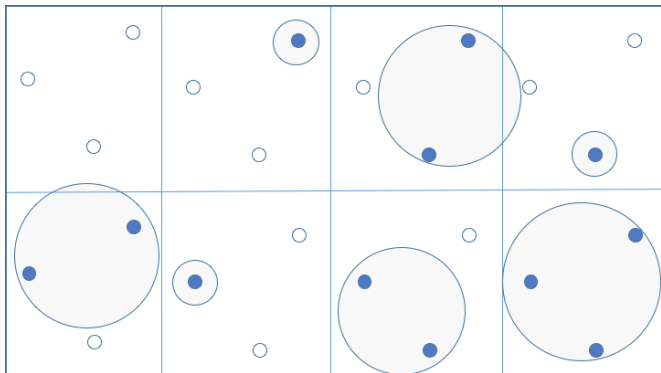
All possible binary labelings of a set  $Z$  of three points in the plane



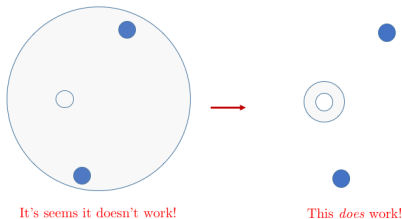
- ▶ Can we find circle functions that shatter the set **for all possible labelings**?
- ▶ Circles must separate the “negative” (blue) labels from the “positive” (white) ones: circles must either enclose all negatively labeled points without enclosing any positively labeled point, or vice versa. It does not matter which class is which, since swapping the labels would only require the classifier to be inverted (i.e., to change sign).

# SHATTERING 3 POINTS IN $\mathbb{R}^2$ WITH CIRCLES

Every possible labeling can be covered by a circle, so we can shatter the 3 points set



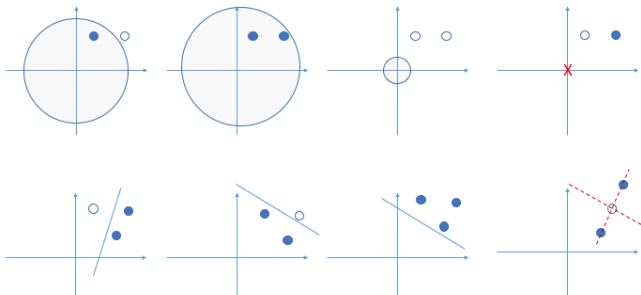
# SHATTERING 3 POINTS IN $\mathbb{R}^2$ WITH CIRCLES



- ▶ We should be precise when defining the class of functions  $h$  we are considering
- ▶ In this case, the class is  $h(x_1, x_2) = a[(x_1 - c_1)^2 + (x_2 - c_2)^2] - r^2$ , that is circles centered in  $(c_1, c_2)$  of radius  $r$ . The coefficient  $a$  can be  $+1$  or  $-1$ . If  $a = +1$ ,  $\text{sign}(h(x_1, x_2)) > 0$  classifies as “1” the samples falling outside of the circle, and “0” those falling inside the circle. Vice versa when  $a < 0$ .
- ▶ Therefore, in the case of the example, the circle is around the white (“1”) labeled sample, but the classifier can still correctly classify the blue (“0”) samples as negative ones using  $a = +1$ .
- ▶ In general, we can assume that we use functions such that the sign of the classifier can be inverted

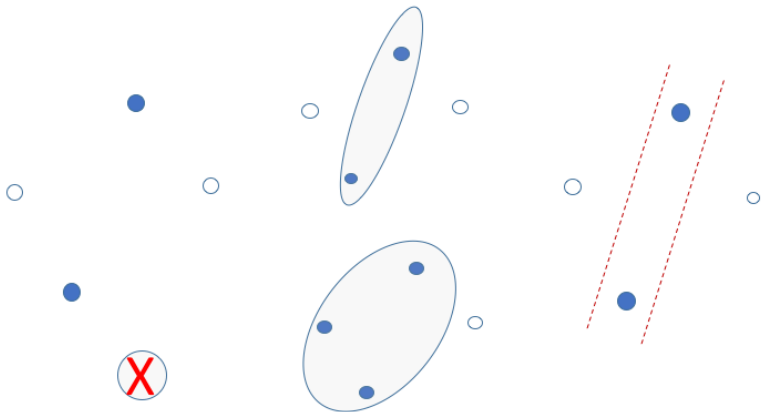
# OTHER SHATTERING EXAMPLES IN 2D

- ▶ In the first row,  $h(x_1, x_2) = x_1^2 + x_2^2 - r^2$ , and the classifier is as usual the  $\text{sign}(h(x_1, x_2))$  function. The last case can't be shattered, differently from the previous example, since the parameter  $a$  is missing, and the sign function classifies as positive what is outside of the origin-centered circle, while the white point would lie inside the circle. There is no way to place a circle centered in the origin such that the two samples are classified correctly: the circle will always enclose the blue point and classify the white point in the same way. In general, it doesn't matter how the two points are placed, such a situation will always arise since one of the two points is necessarily closer to the origin than the other.
- ▶ In the second row,  $h_{\theta}(x_1, x_2) = \theta_1 x_1 + \theta_2 x_2 + \theta_0$ . It's always possible to find a vector  $\theta$  such that the function  $h_{\theta}$  (an oriented line) correctly classifies three points, as long as they are **not collinear** (i.e., as long as they are in *general positions*)



# SHATTERING FOUR POINTS IN 2D

- ▶ There's no way to shatter four points using circles, since the labeling in the figure won't be feasible for a circle (left)
- ▶ It's however possible using *ellipsis* (center)
- ▶ There's no way to shatter four points using lines, again the labeling in the figure is not achievable by a linear separator (right)

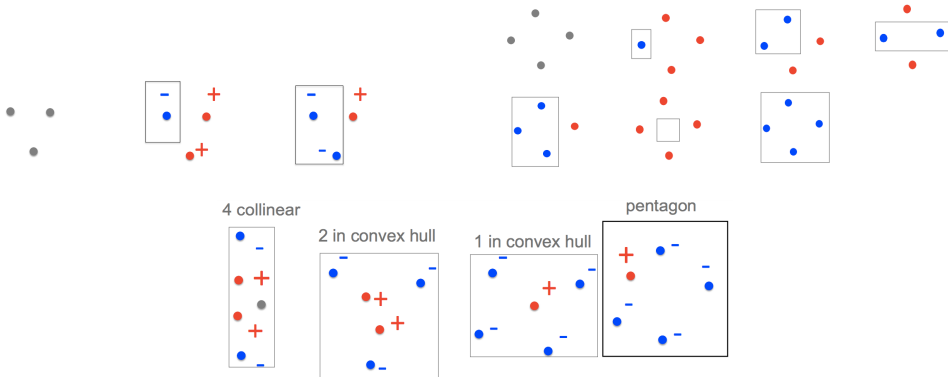


# MEASURING THE VC-DIMENSION

- ▶ The quantify the VC-dimension of a concept class  $\mathcal{H}$  over an instance space  $\mathbf{X}$ :
  - 1 Demonstrate it can shatter one set of size  $n$  (lower bound). This does not mean that  $\mathcal{H}$  can shatter any set of size  $n$ !
  - 2 Demonstrate it cannot shatter *any* set of size  $n + 1$  (upper bound). For all  $n + 1$ -dimensional sets there is (at least) one labeling that can't be shattered
- ▶ Over a  $\mathbb{R}^2$  instance space, a linear classifier (oriented line)  $h_{\theta}(x_1, x_2) = \theta_1 x_1 + \theta_2 x_2 + \theta_0$  can shatter sets of 3 points (as long as they are not collinear) but cannot shatter *any* set of four points. Therefore, the **VC-dimension of a linear learning machine over  $\mathbb{R}^2$  is 3**.
- ▶ The result can be generalized: the **VC-dimension of the set of the oriented hyperplanes in  $\mathbb{R}^n$  is  $n + 1$** .
- ▶ The VC-dimension of axis-aligned hyperplanes is also  $n + 1$ . In fact, as a special case of the general oriented hyperplanes, it's possible to find an  $n$ -dimensional example, but it would clearly fail shattering in the  $n + 1$ -dimensional case
- ▶ Over a  $\mathbb{R}^2$  instance space, the **VC-dimension of general circle functions is 3**. Instead, for origin-centered circles, the VC-dimension is 2 (they can clearly consistently classify set of one element, but not sets of two elements)
- ▶ The VC-dimension gives concreteness to the notion of “power” (**capacity**) of a learning machine: an upper bound on the number of examples that a class of hypothesis could consistently classify

# AXIS PARALLEL RECTANGLES IN 2D

- ▶  $X$  is the set of all points in  $\mathbb{R}^2$
- ▶  $\mathcal{H}$  is the set of all axis parallel rectangles in 2D
- ▶ (Left)  $VC \geq 3$  since there is a placement of 3 points that can be shattered
- ▶ (Right)  $VC \geq 4$  since there is a placement of 4 points that can be shattered
- ▶ (Middle)  $VC = 4$  since for all placements of 5 points, there exists a labeling that can't be shattered



Figures from Barnabás Póczos



# INTERVALS ON THE REAL LINE

- ▶ The concept class is defined by two parameters  $\theta_1$  and  $\theta_2$  in  $[0,1]$ , that define an interval  $[\theta_1, \theta_2]$  on the real line. A concept function tags an input sample  $x \in (0, 1)$  as positive if  $\theta_1 \leq x \leq \theta_2$ , and negative otherwise.
- ▶  $VC\text{-dim} \geq 2$ . Selected a sample of 2 points  $x_1$  and  $x_2$  in  $(0, 1)$ , we need to show that there are values of  $\theta_1$  and  $\theta_2$  which realize all the possible four labelings:  $\{(+, +), (-, -), (+, -), (-, +)\}$ .
- ▶ This is clearly possible as one can place the interval  $[\theta_1, \theta_2]$  such that the intersection with the interval  $[x_1, x_2]$  is null, (thus producing  $(-, -)$ ), or to fully include  $[x_1, x_2]$  (thus producing  $(+, +)$ ) or to partially intersect  $[x_1, x_2]$  such that  $x_1$  or  $x_2$  are excluded (thus producing the remaining two labelings).
- ▶  $VC\text{-dim}$  cannot be more than 2, since any sample of three points  $\{x_1, x_2, x_3\}$  on the line  $(0, 1)$  cannot be shattered ( $x_1 < x_2 < x_3$ ). It is sufficient to show that one of the labelings is not realizable: in particular, the labeling  $(+, -, +)$  cannot be realizable by any interval  $[\theta_1, \theta_2]$  because if  $x_1, x_3$  are labeled positive then by definition the interval  $[\theta_1, \theta_2]$  must fully include the interval  $[x_1, x_3]$  and since  $x_1 < x_2 < x_3$  then  $x_2$  must be labeled positive as well, which makes the labeling unfeasible.

# ADDITIONAL SOURCES

- ▶ More examples, as well as an accessible treatment of the topics related to the VC dimension and PAC learning can be found in <http://www.liaolin.com/Courses/vc-dimension.pdf>
- ▶ The following are the slides from Emma, that summarize the results deriving from the use of the VC dimension

# VC DIMENSION

- **Poll 3:**  $X =$  real line,  $\mathcal{C} =$  intervals, what is  $\text{VC-dim}(\mathcal{C})$ ?

1            3

2             $\infty$

- **Poll 4:**  $X =$  real line,  $\mathcal{C} =$  unions of intervals, what is  $\text{VC-dim}(\mathcal{C})$ ?

2            4

3             $\infty$

# SAMPLE COMPLEXITY

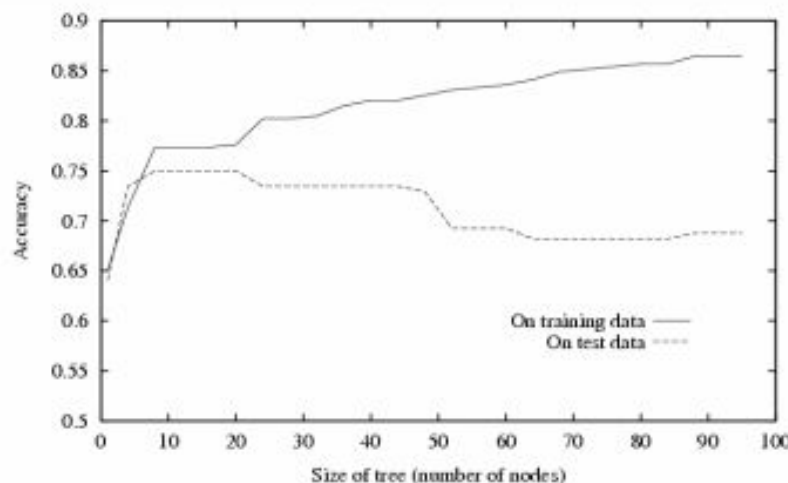
- **Theorem:** a concept class  $\mathcal{C}$  with  $\text{VC-dim}(\mathcal{C}) = \infty$  is not PAC learnable
- **Theorem:** Let  $\mathcal{C}$  with  $\text{VC-dim}(\mathcal{C}) = d$ . Let  $L$  be an algorithm that produces an  $h \in \mathcal{C}$  that is **consistent** with the given samples  $S$ . Then  $L$  is a learning algorithm for  $\mathcal{C}$  with  $m_0 = c_0 \left( \frac{1}{\epsilon} \log \frac{1}{\delta} + \frac{d}{\epsilon} \log \frac{1}{\epsilon} \right)$

# Agnostic Learning: VC Bounds

[Schölkopf and Smola, 2002]

With probability at least  $(1-\delta)$  every  $h \in H$  satisfies

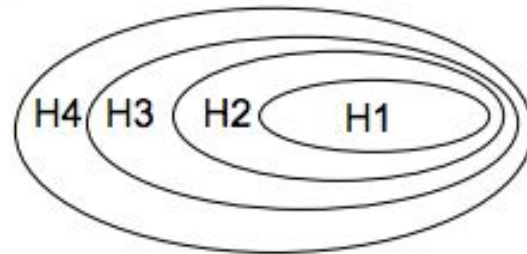
$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$



# Structural Risk Minimization [Vapnik]

Which hypothesis space should we choose?

- Bias / variance tradeoff













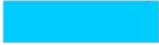
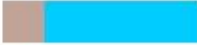






SRM: choose H to minimize bound on true error!

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

\* unfortunately a somewhat loose bound...

# Structural Risk Minimization

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

$i$	$f_i$	TRAINERR	VC-Confidence	Probable upper bound on TESTERR	Choice
1	$f_1$				
2	$f_2$				
3	$f_3$				⊗
4	$f_4$				
5	$f_5$				
6	$f_6$				

# ML Model Class Selection: What You Should Know

- Define training error, generalization error and model selection problem
- Be able to apply training set partitioning to both identify a model class expect to yield good generalization error, and provide an estimation of that generalization error (and explain why this procedure is reasonable)
  - Empirical approach:
    - partition data
- Define VC dimension and be able to
  - Prove the VC dimension of a particular model class
  - Use it to obtain a bound on the generalization error
  - Know how many data points are needed to learn a PAC classifier as a function of VC



# Online Resources

<http://www.autonlab.org/tutorials/vcdim08.pdf>

<http://www.cs.cmu.edu/~awm/10701/slides/PAC-learning-10-25-05.pdf>

<https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Recitations.VCDim>

<http://web.engr.oregonstate.edu/~xfern/classes/cs534/midterm-solutions-07.pdf>

<http://www.cs.cmu.edu/~gustrin/Class/10701/slides/learningtheory-bigpicture.pdf>