Lecture: Advanced Topics in Machine Learning

Foundations

Markus Lange-Hegermann

Organization

- Exams probably oral
- Very few lectures/exercises
- Switch to interactive reading course
  - you get homework for every session!
- Later: presentations by you
- Sprinkled in: PyTorch tutorials (Andreas, Ruwen)
- Organization via Ilias
- Dates
  - Monday, 8:00-11:15, 1.376 (15 Minute break?)

Math Background

Comments

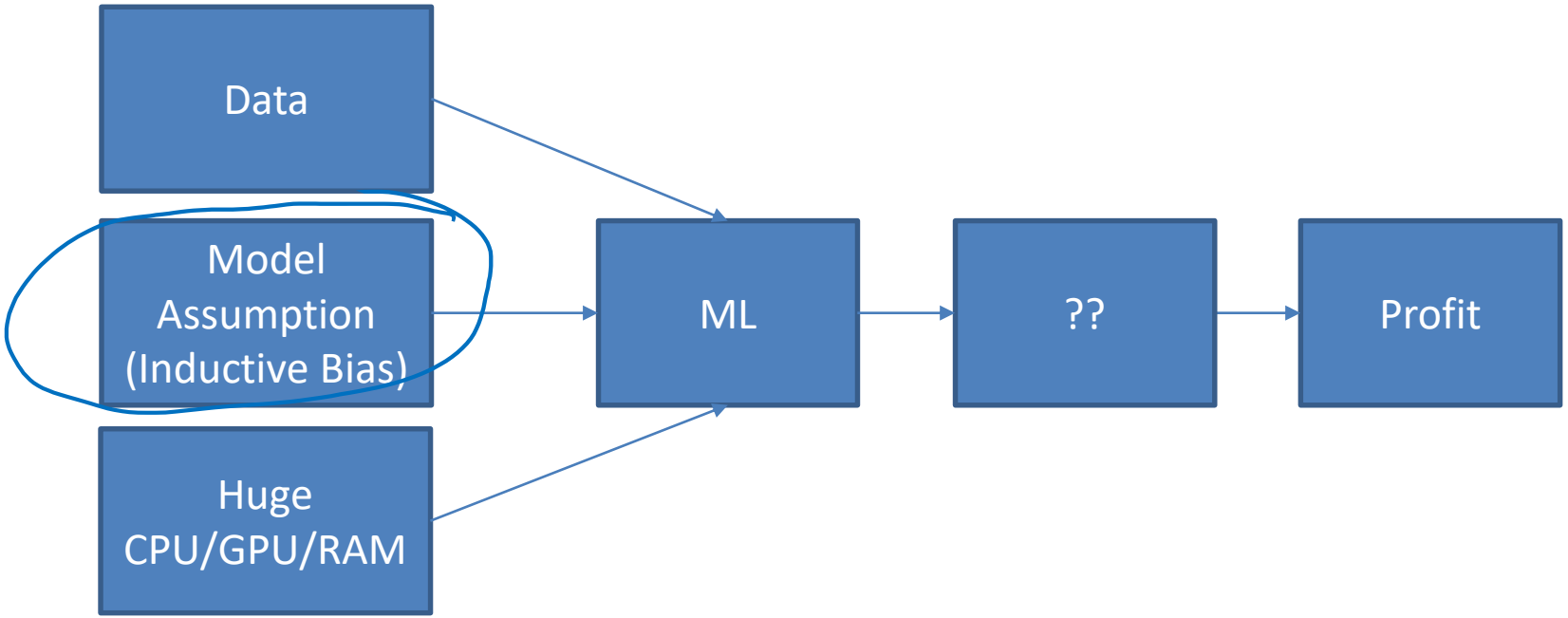
- Last year’s students described the course as hard, but informative, and moaned about too much work.
  - I do not intent to change the difficulty from last year.
- Two years ago I changed the content from Goodfellow’s book Deep Learning to Murphy’s book
  - I expect that you work for yourselves on the topics and not just expose yourselves to the lectures.
- I removed the math part, due to suggestions two years ago. I expect you to fill the gaps yourselves.

- Some lecture slides taken in part from Ian Goodfellow (originally for Deep Learning) and Kevin P. Murphy (Probabilistic Machine Learning - An Introduction)

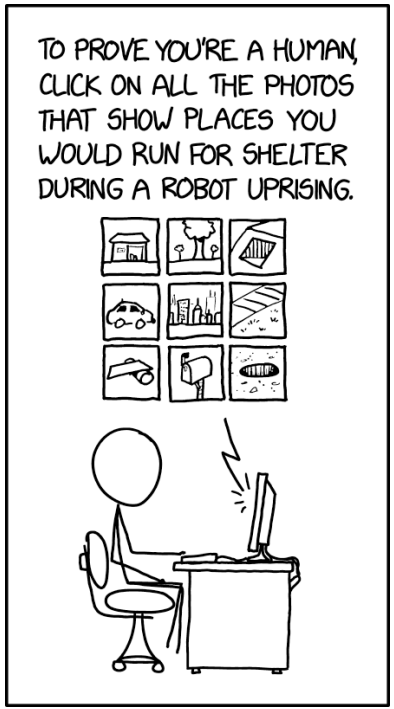
Surrounding activities

- AICommunityOWL
  - Hackathons
  - Talks
- Graduate Seminars
  - my working group, weekly
  - inIT&Fraunhofer, monthly
- Research Projects
- JAII lectures series
- Summer/spring schools

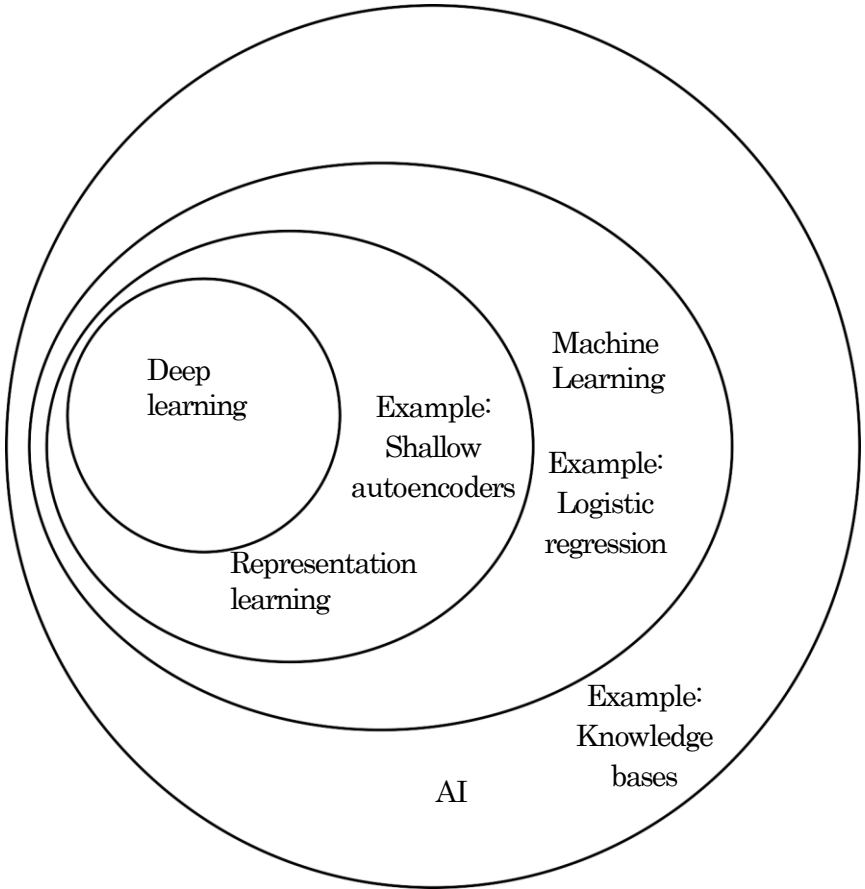
What is ML?


Data
Model
Assumption
(Inductive Bias)
Huge
CPU/GPU/RAM
ML
??
Profit

ML vs AI


TO PROVE YOU'RE A HUMAN, CLICK ON ALL THE PHOTOS THAT SHOW PLACES YOU WOULD RUN FOR SHELTER DURING A ROBOT UPRISING.

Machine Learning and AI



This Course

- Changing field and SOTA. Focus: be able to read paper
- This years reference:
  - Probabilistic Machine Learning - An Introduction, Kevin P. Murphy
  - https://probml.github.io/pml-book/book1.html
- Older References:
  - Deep Learning, Goodfellow&Bengio&Courville
    - https://www.deeplearningbook.org/
  - Gaussian Processes for ML, Rasmussen&William
    - http://www.gaussianprocess.org/gpml/chapters/RW.pdf
- All freely available

ML Basics

- Learn a specific task f from data
- Goal:
  - $f: Example \rightarrow Outcome$
- Both *Example* and *Outcome* are vectors (or more complicated)
- Learn mostly by using data, making errors, and trying to avoid these errors

Tasks

- Supervised (predict label/target/value)
  - Data: (x_i,y_i) (has labels)
  - Classification, Regression
  - Denoising
  - Translation, Speech recognition, image segmentation
- Unsupervised (learn prepresentation/probability)
  - Data: x_i (does not have labels)
  - Density Estimation (e.g. anomaly detection)
  - Data representation, dimension reduction
  - Retrieve missing values
  - Generative models (sampling, art generation, deep fakes, text generation, …)
- Reinforcement learning
- Etc.

Example: Large Language Models

M how does chatgpt work?

ChatGPT is a large language model that works by utilizing a type of artificial intelligence called deep learning. It has been pre-trained on a massive amount of text data using a process called unsupervised learning.

During the pre-training process, ChatGPT was fed a huge amount of text data, such as books, articles, and web pages, and was trained to predict the next word in a sequence of words.
phrases

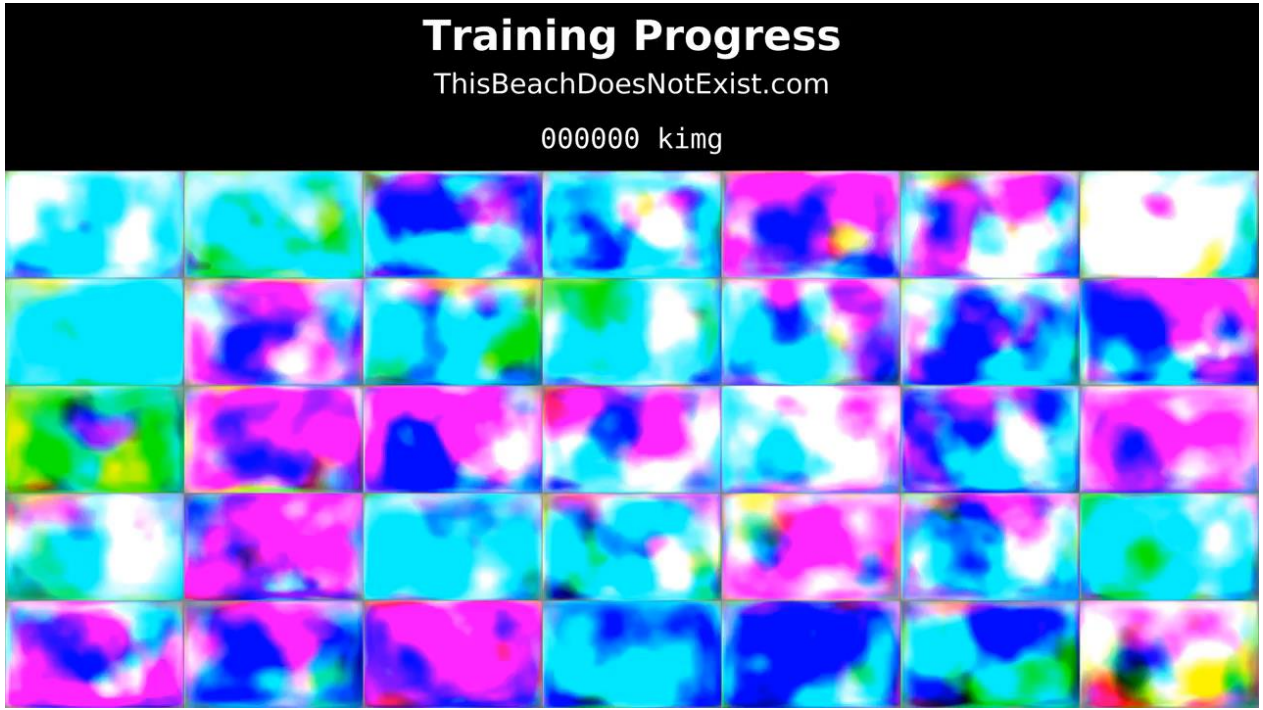
When y
words,
underst
efficien

ChatGP
interact
respons

M answer shorter

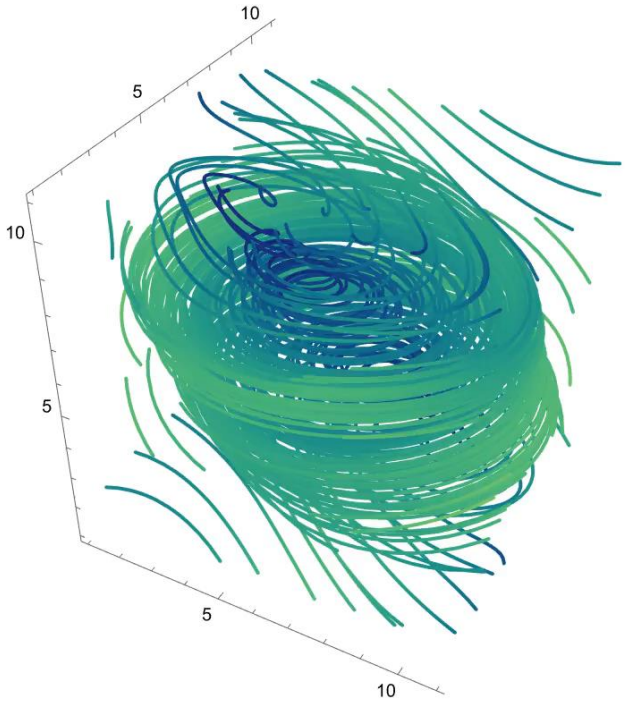
ChatGPT is an artificial intelligence language model that uses deep learning to understand language and generate responses based on input. It has been pre-trained on a massive amount of text data, and constantly improves through its interactions with users.

thisbeachdoesnotexist.com/

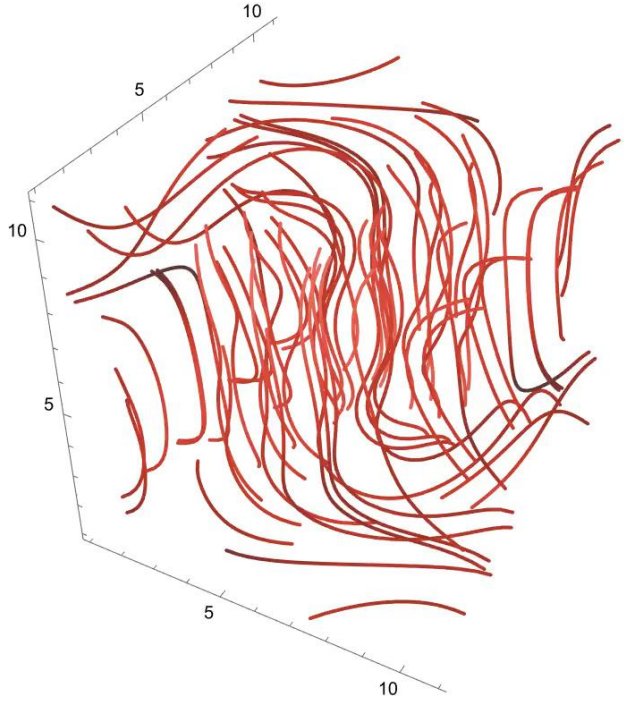


GAN Latent Space Interpolation

Gaussian Process Priors for Systems of Linear Partial Differential Equations with Constant Coefficients



E <-> B



Main Idea: Probabilities

f : data → prob. distr.

The outputs of models parametrize probability distributions

- Construct a probability distribution of the data
- Sample the probability distribution
- Not only learn predictions, but also variances
- Do not classify, but predict class probabilities
- Condition probability distributions on prior knowledge
- Initialize neural networks randomly to generate a suitable random function
- Stochastic optimization algorithms
- Etc.

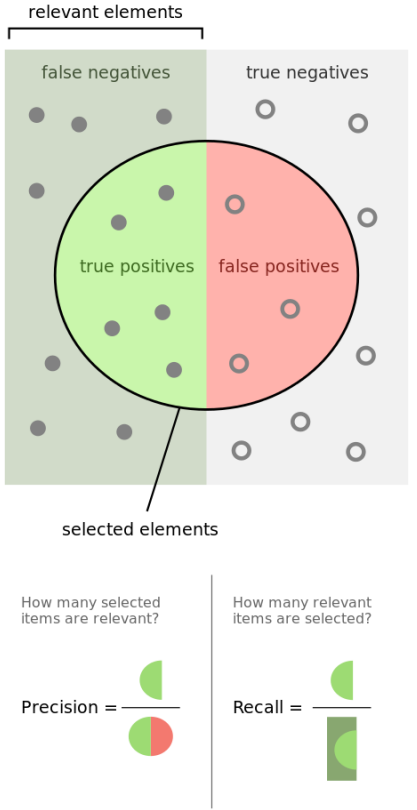
old: $y = f(x)$
now: $p(y \mid x)$

Performance Measure

- L1, L2=least squares, $||f(x_i) - y_i||$ (for regression)
- Likelihood (important!)
- Maximum posterior (important!)
- True Bayes (the best, but hard)
- Accuracy (for classification)
- Special case: error rate = 0-1-loss
- F-score (for classification)
  - harmonic mean of precision&recall
- Etc.

Test sets! Validation sets!

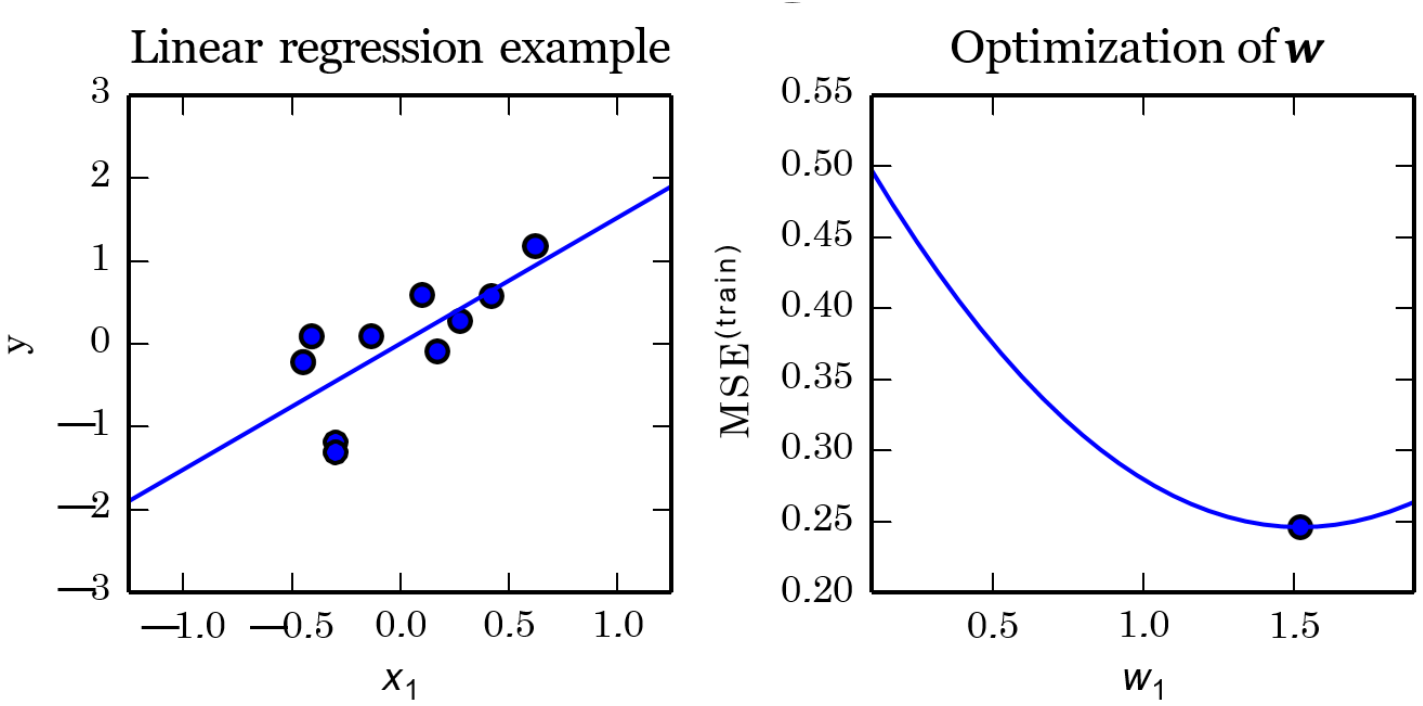
Use many performance measure!



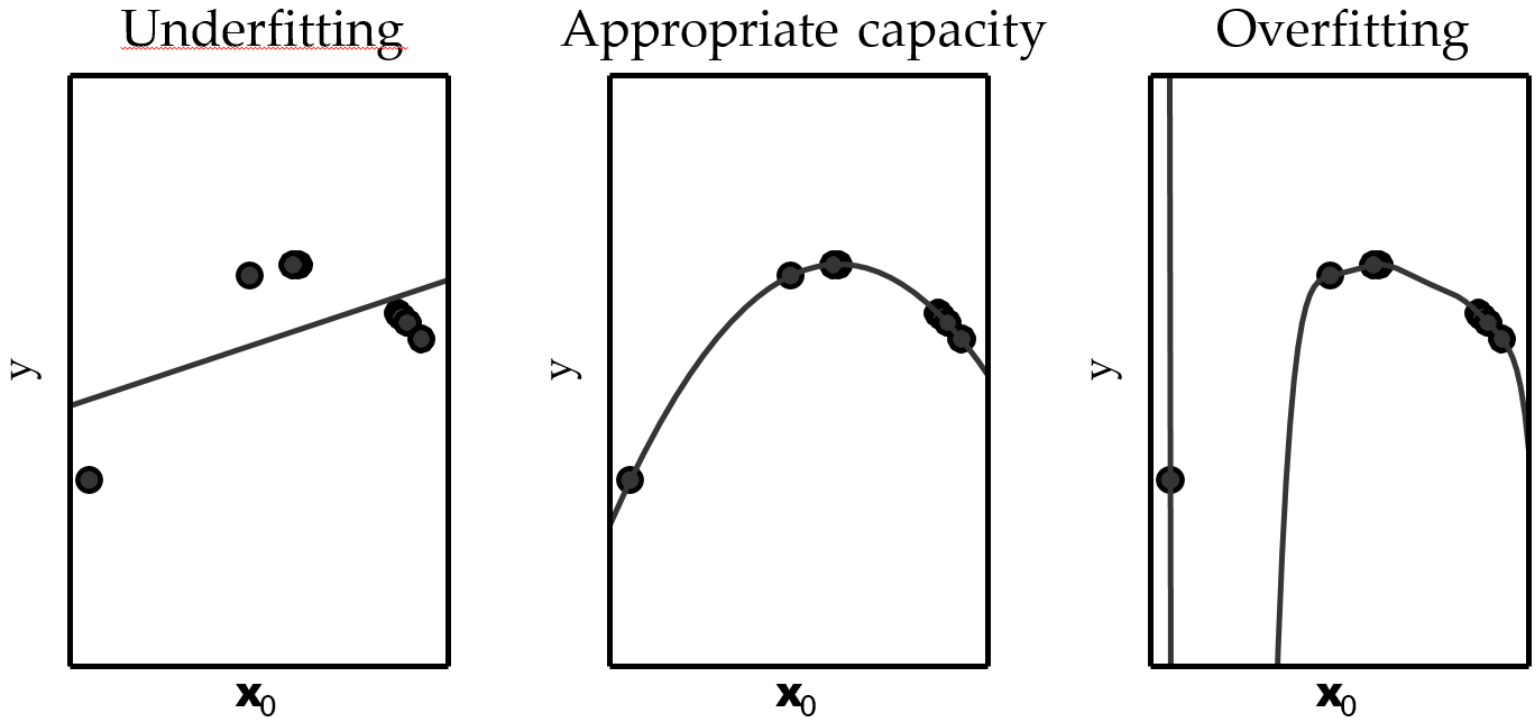
Interesting Challanges

- Missing inputs (marginalization?)
- Noise data/labels (default)
- Uneven data distribution (ignore, sampling, …)
- Real time (implementation, model capacity)
- Not enough data (other model)
- Interpretability
- Include expert knowledge
- autoML
- Bias (racism, sexism, …), ethics, alignment
- Etc.

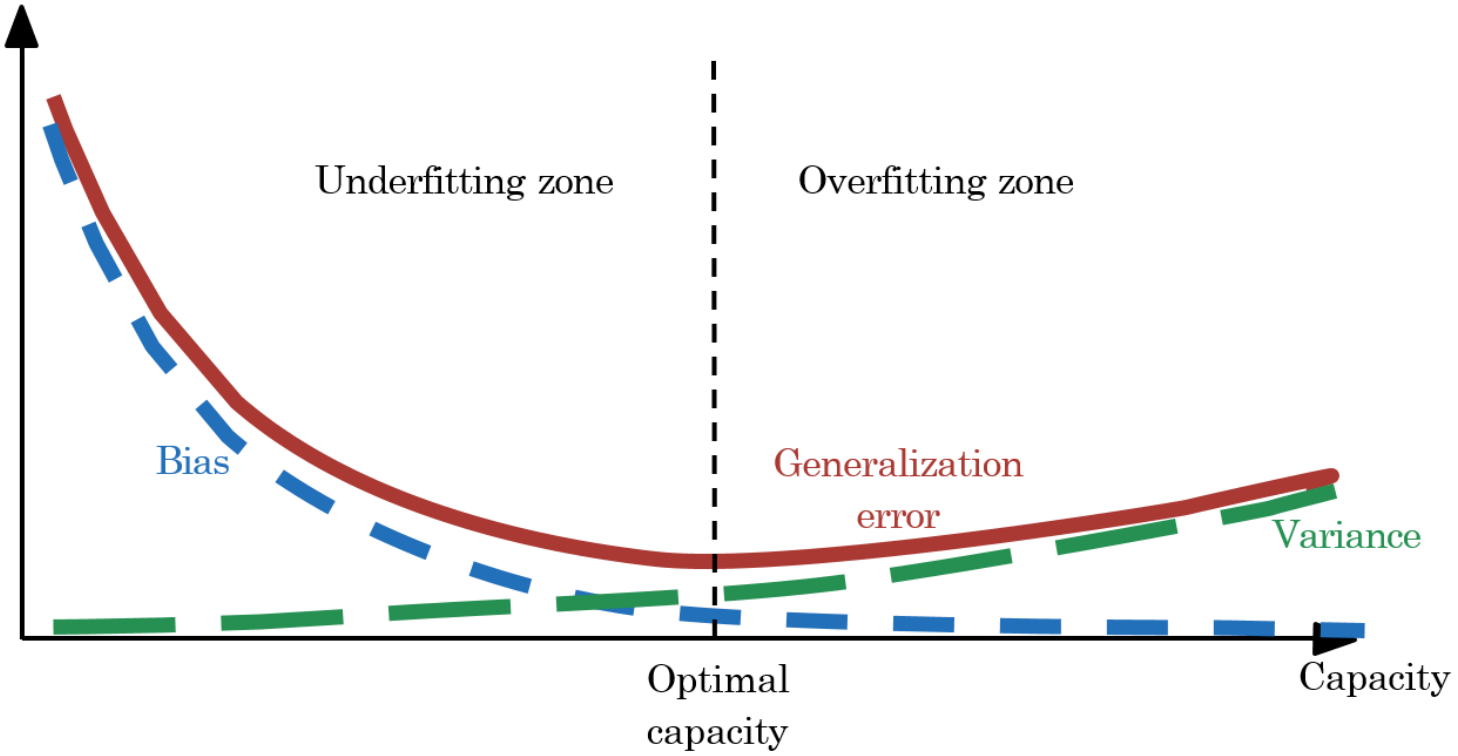
Linear Regression



Underfitting and Overfitting in Polynomial Estimation

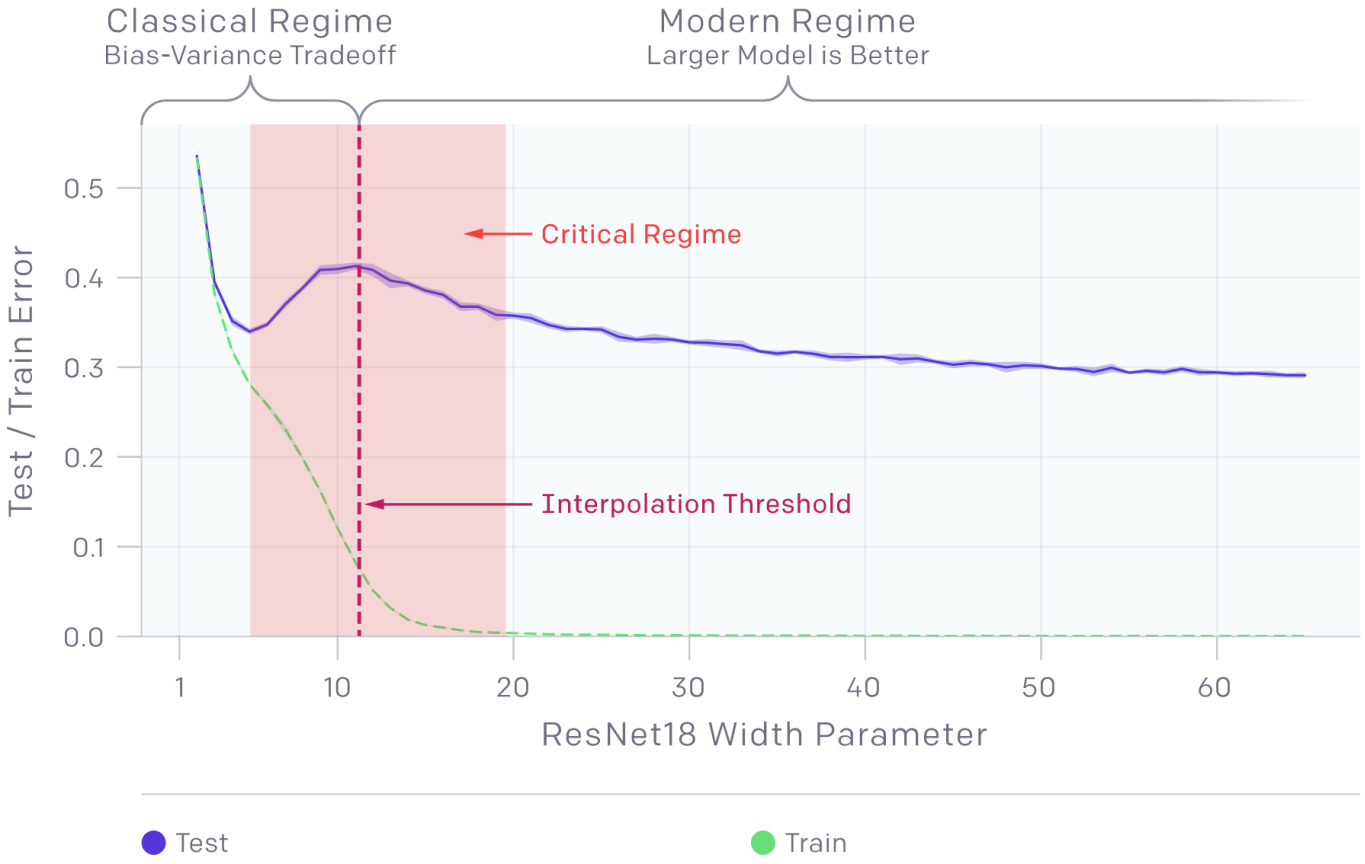


Bias and Variance



Bias and Variance: a new view

Bias and Variance: a new view



Maximum Likelihood

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}).$$

$$D_{\text{KL}}\left(\hat{p}_{\text{data}} \| p_{\text{model}}\right) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}} \left[\log \hat{p}_{\text{data}}(\boldsymbol{x}) - \log p_{\text{model}}(\boldsymbol{x})\right].$$

$$- \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}} \left[\log p_{\text{model}}(\boldsymbol{x})\right]$$

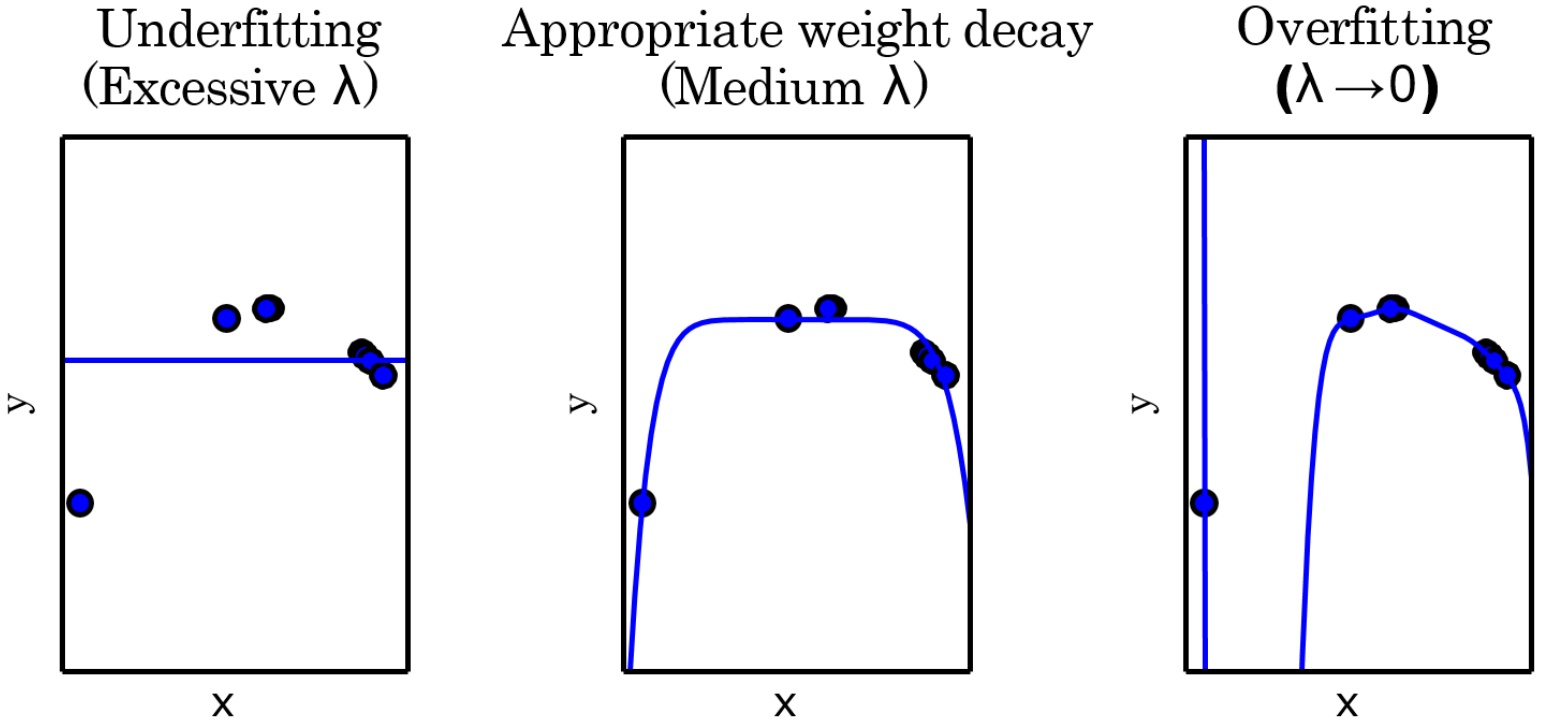
Max A Posteriori (MAP)

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}).$$

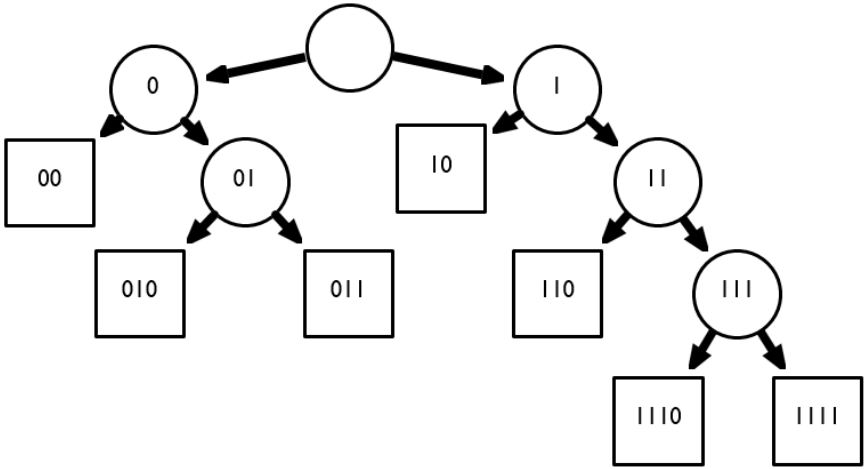
regularization

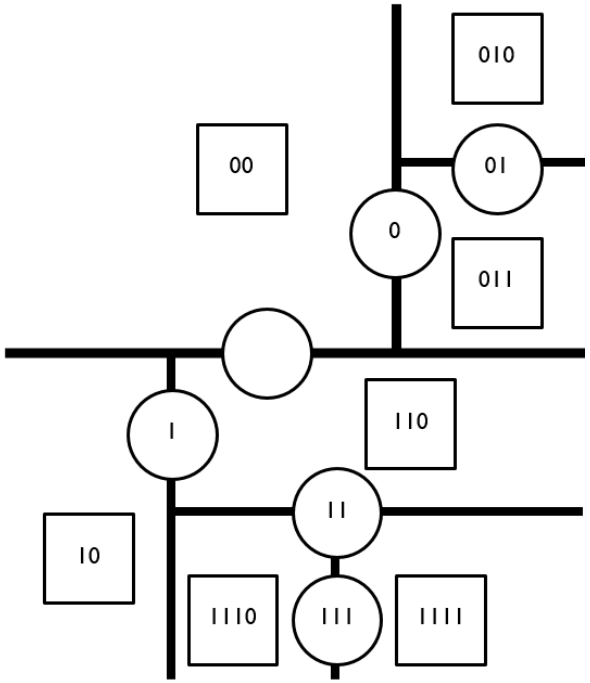
$$p(\theta \mid x) = \frac{p(x \mid \theta)}{p(x)} \cdot p(\theta)$$

Weight Decay $\ell_2(\theta) = -\|\theta\|_2$

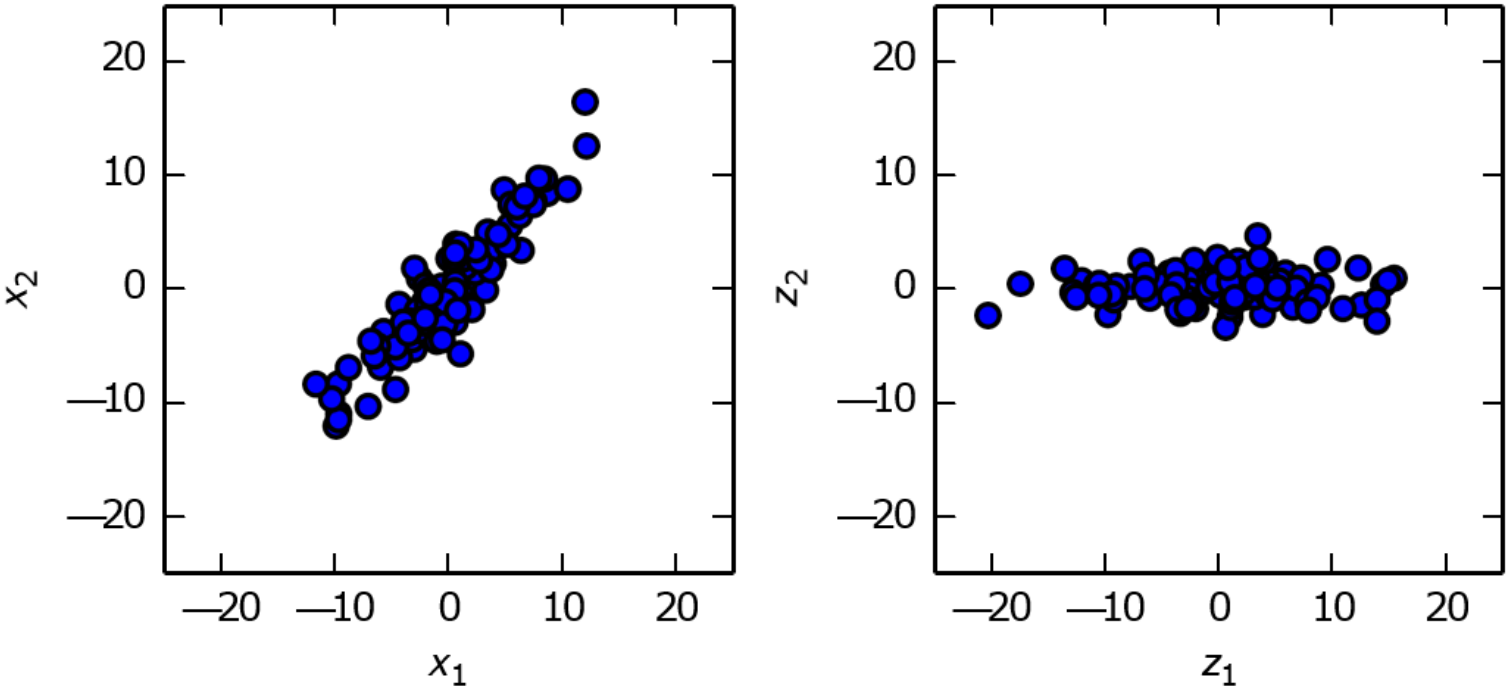


Decision Trees

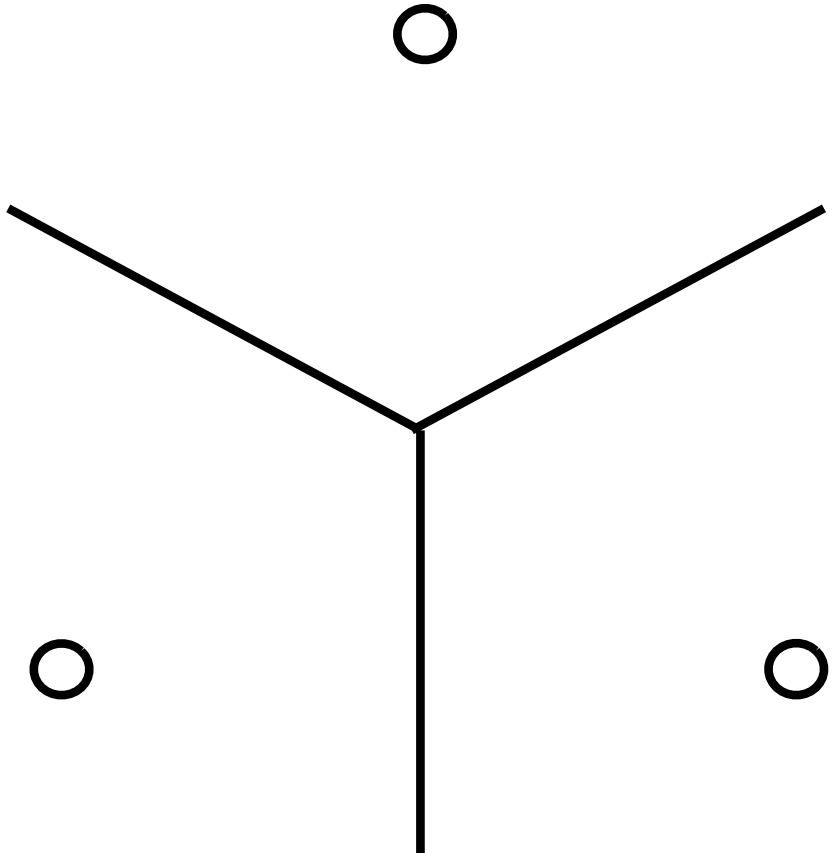




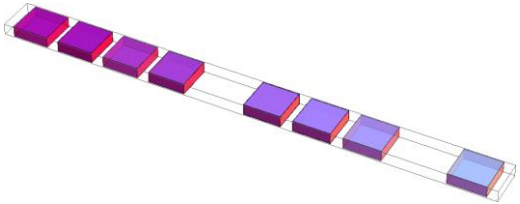
Principal Components Analysis

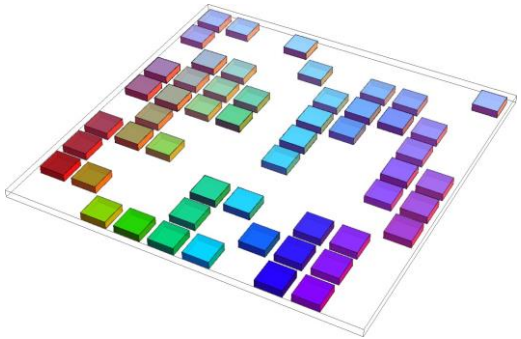


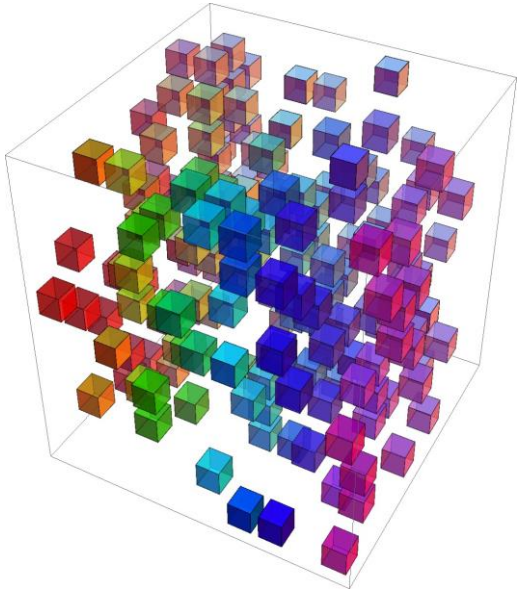
Nearest Neighbor



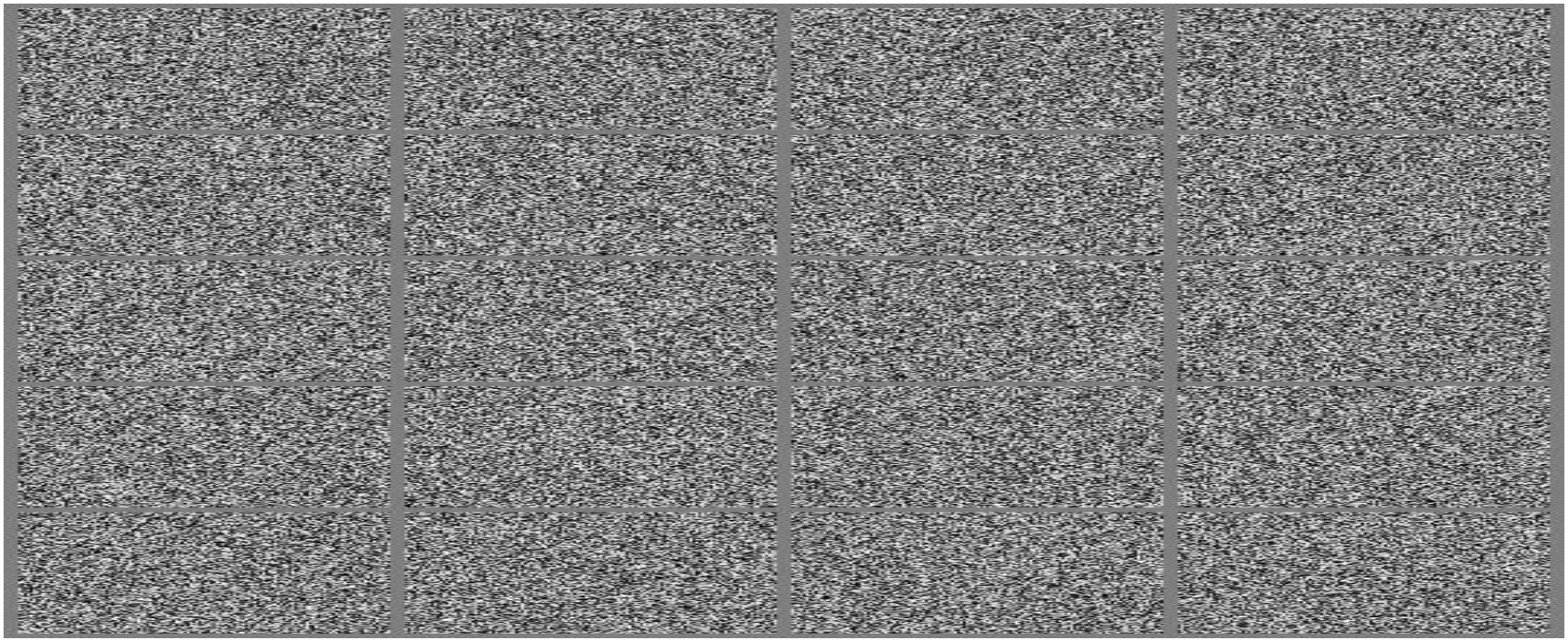
Curse of Dimensionality



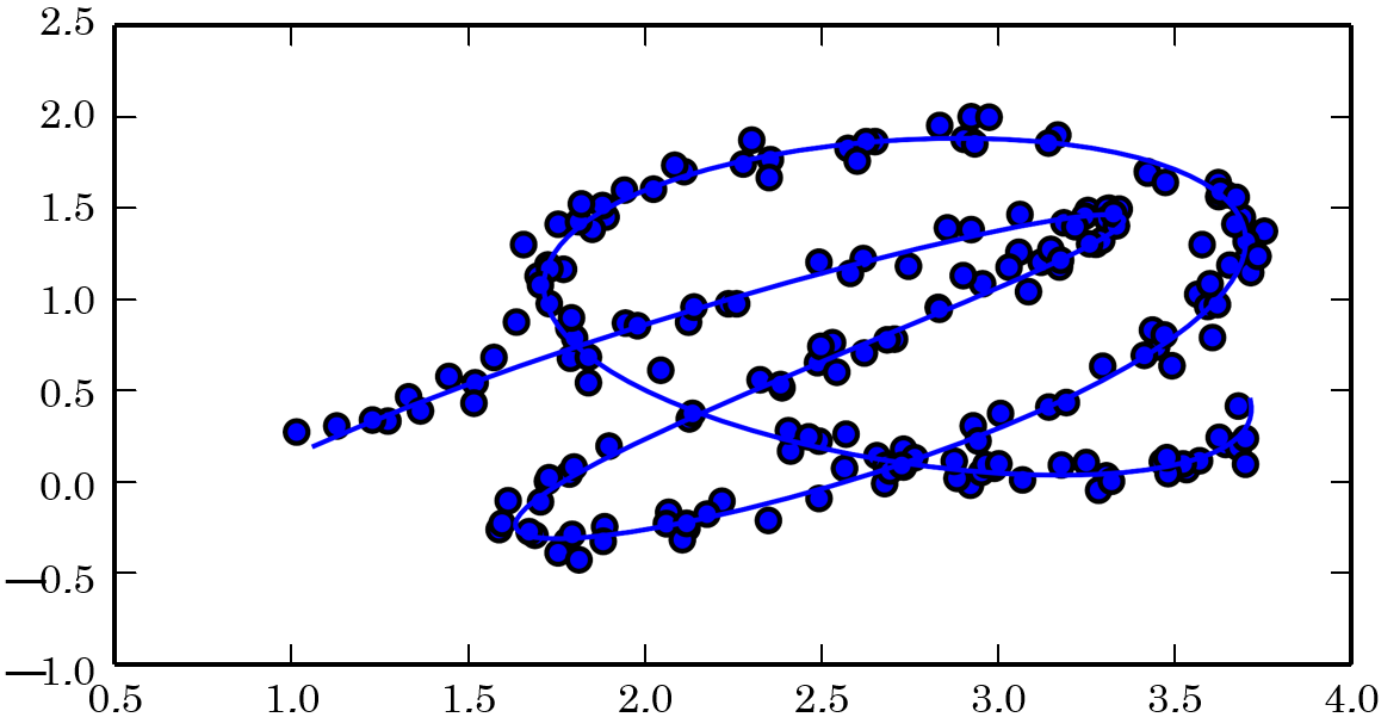




Uniformly Sampled Images

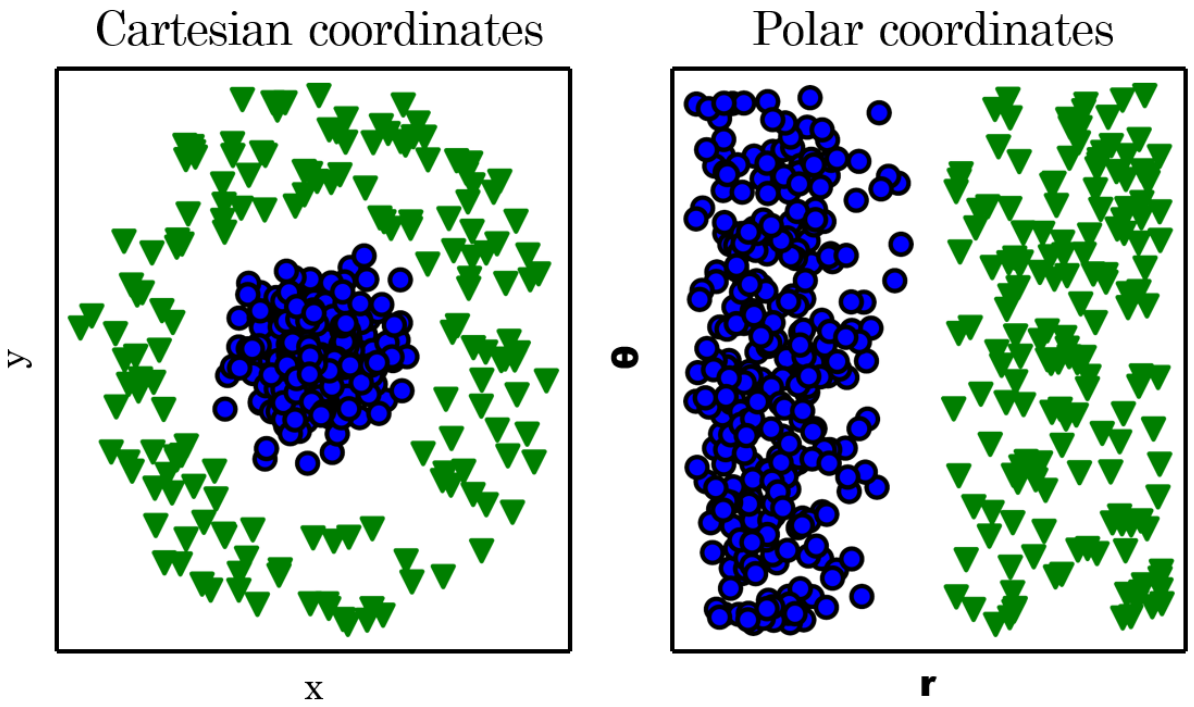


Manifold Learning

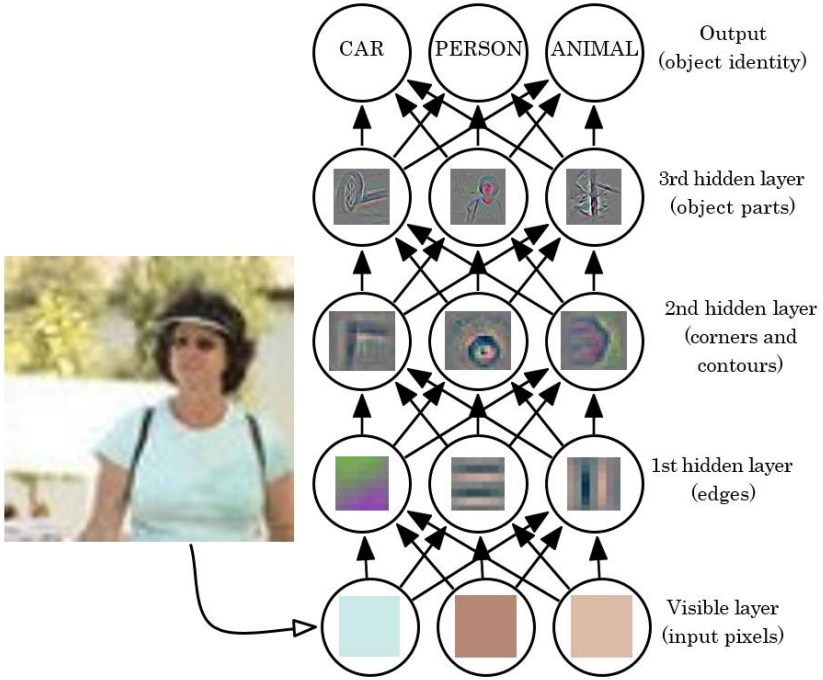


QMUL Dataset

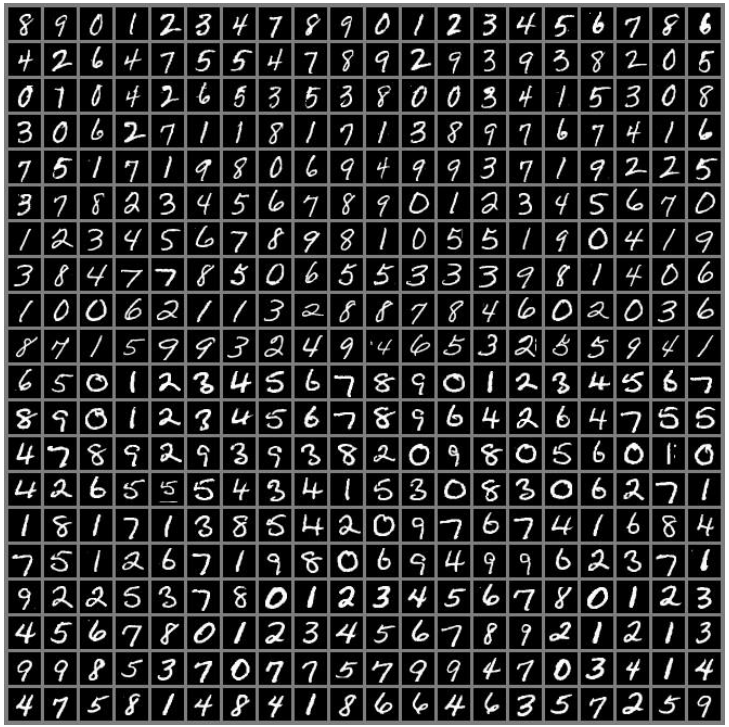
Representations Matter



Depth: Repeated Composition



The MNIST Dataset



What is possible by computer?

- In the 60s, Marvin Minsky assigned a couple of undergrads to spend the summer programming a computer to use a camera to identify objects in a scene. He figured they'd have the problem solved by the end of the summer. Half a century later, we're still working on it.
- Comic published: Sep. 2014

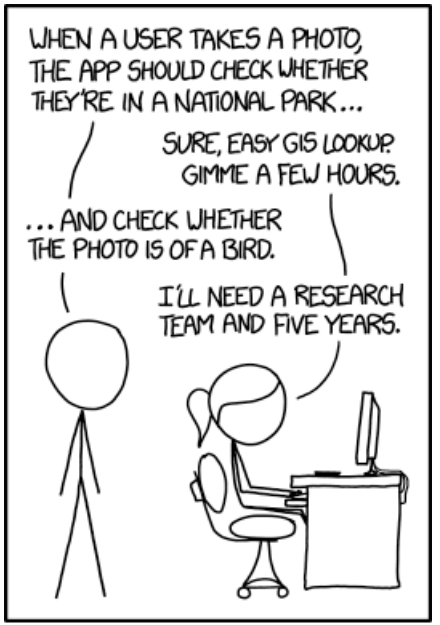
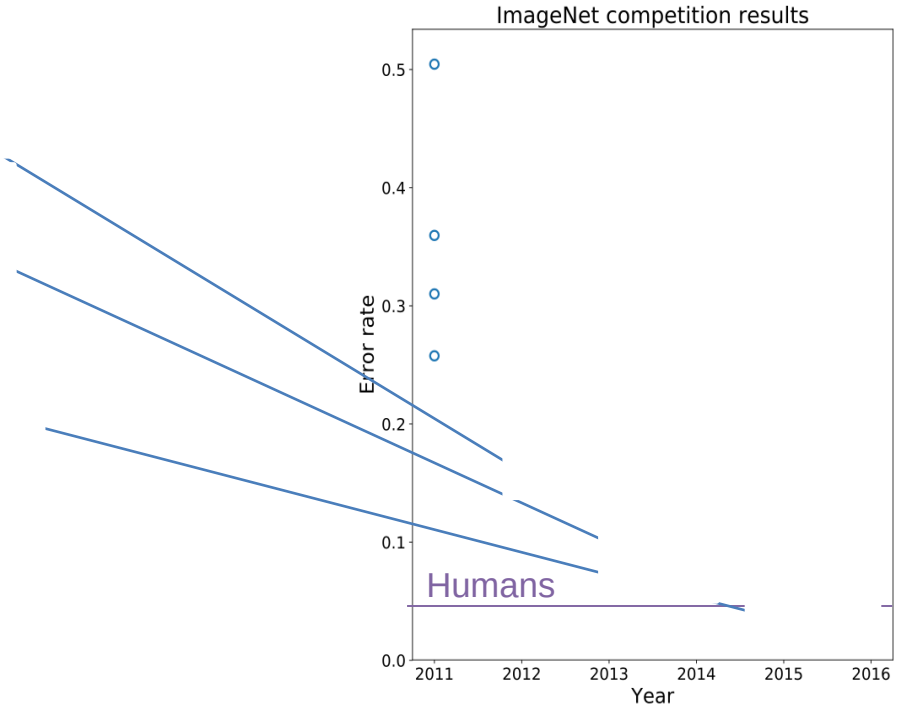


Image recognition



https://en.wikipedia.org/wiki/File:ImageNet_error_rate_history_(just_systems).svg, CC BY-SA 4.0

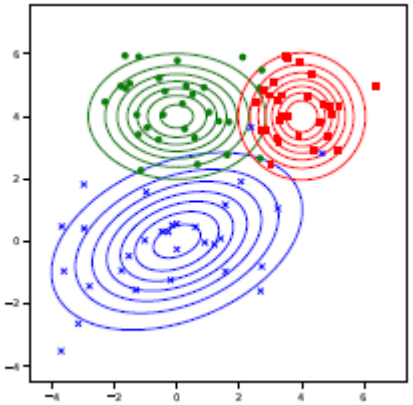
ML Algorithm Anatomy

- Dataset
- Model
  - Linear model, neural net model, Gaussian process model, …
- Cost function
  - Neg. log Likelihood, regularization, priors
- Optimization procedure
  - SGD, GD, Newton, grid search, evolutionary algorithm, closed form, guess…

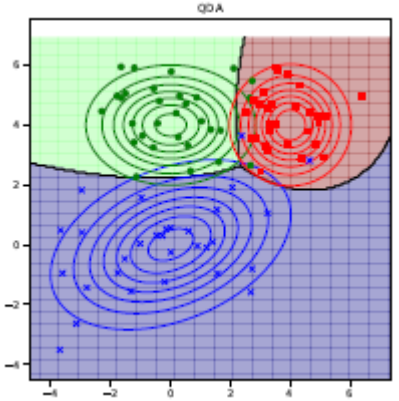
Can mostly be combined independently
Sometimes it is necessary to approximate parts
Always make sure you know which parts you are using

Exampe Session: Linear Discriminant Analysis

- We read Section 9 except FLDA, 11 pages
- Idea: probabilistic model (often Gaussian) for each class
- Apply Bayes theorem to get posterior densities



$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|y = c; \boldsymbol{\theta})p(y = c; \boldsymbol{\theta})}{\sum_{c'} p(\boldsymbol{x}|y = c'; \boldsymbol{\theta})p(y = c'; \boldsymbol{\theta})}$$



9 Linear Discriminant Analysis

$$p(y|x) \overset{\text{Bayes}}{=} \frac{p(x|y)}{p(x)} p(y)$$

$$= \frac{p(x|y)}{\sum_{c'} p(x|y=c') \cdot p(y=c')} \cdot p(y)$$

9.1 Introduction

In this chapter, we consider classification models of the following form:

$$p(y = c|x; \theta) \overset{\text{Bayes}}{=} \frac{p(x|y = c; \theta)p(y = c; \theta)}{\sum_{c'} p(x|y = c'; \theta)p(y = c'; \theta)} \tag{9.1}$$

The term $p(y = c; \theta)$ is the prior over class labels, and the term $p(x|y = c; \theta)$ is called the **class conditional density** for class c.

The overall model is called a **generative classifier**, since it specifies a way to *generate* the features x for each class c, by sampling from $p(x|y = c; \theta)$. By contrast, a **discriminative classifier** directly models the class posterior $p(y|x; \theta)$. We discuss the pros and cons of these two approaches to classification in Section 9.4.

If we choose the class conditional densities in a special way, we will see that the resulting posterior over classes is a linear function of x, i.e., $\log p(y = c|x; \theta) = w^\mathsf{T} x + \text{const}$, where w is derived from θ. Thus the overall method is called **linear discriminant analysis** or **LDA**.[1]

9.2 Gaussian discriminant analysis

In this section, we consider a generative classifier where the class conditional densities are multivariate Gaussians:

$$p(\boldsymbol{x}|y=c,\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{9.2}$$

The corresponding class posterior therefore has the form

$$p(y=c|\boldsymbol{x},\boldsymbol{\theta}) \propto \pi_c \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{9.3}$$

where $\pi_c = p(y=c)$ is the prior probability of label c. (Note that we can ignore the normalization constant in the denominator of the posterior, since it is independent of c.) We call this model **Gaussian discriminant analysis** or **GDA**.

$\log p(y=c|x,\theta) = \log\left(\frac{\pi_c \cdot p(x|y=c,\theta)}{\text{const}}\right) = \log(\pi_c) + \log p(x|y=c,\theta) - \log(\text{const}')$

$= -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(x-\mu_c)^T\Sigma_c^{-1}(x-\mu_c)$

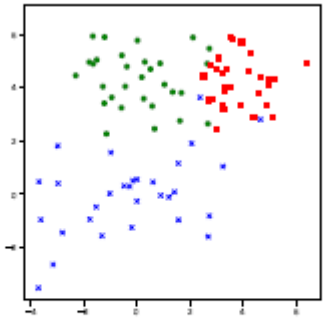
9.2.1 Quadratic decision boundaries

From Equation (9.3), we see that the log posterior over class labels is given by

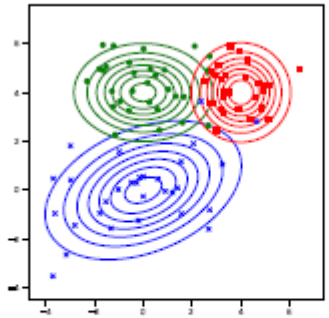
$$\log p(y=c|\boldsymbol{x},\boldsymbol{\theta}) = \log \pi_c - \frac{1}{2}\log|2\pi\boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_c)^\mathsf{T}\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_c) + \text{const} \tag{9.4}$$

This is called the **discriminant function**. We see that the decision boundary between any two classes, say c and c', will be a quadratic function of $\boldsymbol{x}$. Hence this is known as **quadratic discriminant analysis** (QDA).

For example, consider the 2d data from 3 different classes in Figure 9.1a. We fit full covariance Gaussian class-conditionals (using the method explained in Section 9.2.4), and plot the results in Figure 9.1b. We see that the features for the blue class are somewhat correlated, whereas the features for the green class are independent, and the features for the red class are independent and isotropic (spherical covariance). In Figure 9.2a, we see that the resulting decision boundaries are quadratic functions of $\boldsymbol{x}$.

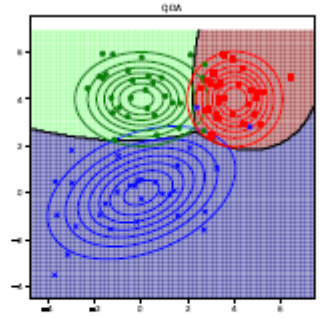


(a)

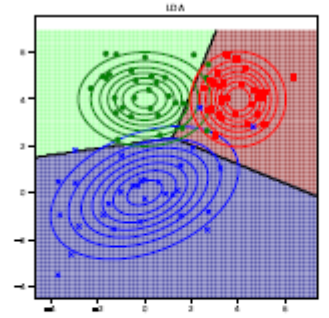


(b)

Figure 9.1: (a) Some 2d data from 3 different classes. (b) Fitting 2d Gaussians to each class. Generated by discrim_analysis_dboundaries_plot2.ipynb.



(a)



(b)

Figure 9.2: Gaussian discriminant analysis fit to data in Figure 9.1. (a) Unconstrained covariances induce quadratic decision boundaries. (b) Tied covariances induce linear decision boundaries. Generated by discrim_analysis_dboundaries_plot2.ipynb.

9.2.2 Linear decision boundaries

Now we consider a special case of Gaussian discriminant analysis in which the covariance matrices are **tied** or **shared** across classes, so $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$. If $\boldsymbol{\Sigma}$ is independent of c, we can simplify Equation (9.4) as follows:

$$\log p(y=c|\boldsymbol{x},\boldsymbol{\theta}) = \log \pi_c - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_c)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_c) + \text{const} \tag{9.5}$$

$$= \underbrace{\log \pi_c - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c}_{\gamma_c} + \boldsymbol{x}^\mathsf{T}\underbrace{\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c}_{\boldsymbol{\beta}_c} + \underbrace{\text{const} - \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}}_{\kappa} \tag{9.6}$$

$$= \gamma_c + \boldsymbol{x}^\mathsf{T}\boldsymbol{\beta}_c + \kappa \tag{9.7}$$

The final term is independent of c, and hence is an irrelevant additive constant that can be dropped. Hence we see that the discriminant function is a linear function of $\boldsymbol{x}$, so the decision boundaries will be linear. Hence this method is called **linear discriminant analysis** or **LDA**. See Figure 9.2b for an example.

9.2.4 Model fitting

We now discuss how to fit a GDA model using maximum likelihood estimation. The likelihood function is as follows

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N_{\mathcal{D}}} \left(\text{Cat}(y_n|\boldsymbol{\pi}) \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)^{\mathbb{I}(y_n=c)} \right) \tag{9.17}$$

Hence the log-likelihood is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N_{\mathcal{D}}} \sum_{c=1}^{C} \mathbb{I}\left(y_n = c\right) \log \pi_c \right] + \sum_{c=1}^{C} \left[\sum_{n:y_n=c} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \right] \tag{9.18}$$

Thus we see that we can optimize $\boldsymbol{\pi}$ and the $(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ terms separately.

From Section 4.2.4, we have that the MLE for the class prior is $\hat{\pi}_c = \frac{N_c}{N}$. Using the results from Section 4.2.6, we can derive the MLEs for the Gaussians as follows:

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_{\mathcal{D}c}} \sum_{n:y_n=c} \boldsymbol{x}_n \tag{9.19}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_{\mathcal{D}c}} \sum_{n:y_n=c} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^{\mathsf{T}} \tag{9.20}$$

Unfortunately the MLE for $\hat{\boldsymbol{\Sigma}}_c$ can easily overfit (i.e., the estimate may not be well-conditioned) if $N_{\mathcal{D}c}$ is small compared to D, the dimensionality of the input features. We discuss some solutions to this below.

$\mathbb{I}(A) = \begin{cases} 1 & : A \text{ true} \\ 0 & : A \text{ false} \end{cases}$

9.2.4.1 Tied covariances

If we force $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$ to be tied, we will get linear decision boundaries, as we have seen. This also usually results in a more reliable parameter estimate, since we can pool all the samples across classes:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N_{\mathcal{D}}} \sum_{c=1}^{C} \sum_{n:y_n=c} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^{\mathsf{T}} \tag{9.21}$$

9.2.4.2 Diagonal covariances

If we force $\boldsymbol{\Sigma}_c$ to be diagonal, we reduce the number of parameters from $O(CD^2)$ to $O(CD)$, which avoids the overfitting problem. However, this loses the ability to capture correlations between the features. (This is known as the naive Bayes assumption, which we discuss further in Section 9.3.) Despite this approximation, this approach scales well to high dimensions.

We can further restrict the model capacity by using a shared (tied) diagonal covariace matrix. This is called "diagonal LDA" [BL04].

9.2.4.3 MAP estimation

Forcing the covariance matrix to be diagonal is a rather strong assumption. An alternative approach is to perform MAP estimation of a (shared) full covariance Gaussian, rather than using the MLE. Based on the results of Section 4.5.2, we find that the MAP estimate is

$$\hat{\boldsymbol{\Sigma}}_{\text{map}} = \lambda \text{diag}(\hat{\boldsymbol{\Sigma}}_{\text{mle}}) + (1 - \lambda)\hat{\boldsymbol{\Sigma}}_{\text{mle}} \tag{9.22}$$

where λ controls the amount of regularization. This technique is known as **regularized discriminant analysis** or RDA [HTF09, p656].

9.2.5 Nearest centroid classifier

If we assume a uniform prior over classes, we can compute the most probable class label as follows:

$$\hat{y}(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} \log p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \underset{c}{\operatorname{argmin}} (\boldsymbol{x} - \boldsymbol{\mu}_c)^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_c) \tag{9.23}$$

This is called the **nearest centroid classifier**, or **nearest class mean classifier** (**NCM**), since we are assigning $\boldsymbol{x}$ to the class with the closest $\boldsymbol{\mu}_c$, where distance is measured using (squared) Mahalanobis distance.

We can replace this with any other distance metric to get the decision rule

$$\hat{y}(\boldsymbol{x}) = \underset{c}{\operatorname{argmin}}\, d^2(\boldsymbol{x}, \boldsymbol{\mu}_c) \tag{9.24}$$

We discuss how to learn distance metrics in Section 16.2, but one simple approach is to use

$$d^2(\boldsymbol{x}, \boldsymbol{\mu}_c) = ||\boldsymbol{x} - \boldsymbol{\mu}_c||_{\mathbf{W}}^2 = (\boldsymbol{x} - \boldsymbol{\mu}_c)^{\mathsf{T}} (\mathbf{W}\mathbf{W}^{\mathsf{T}}) (\boldsymbol{x} - \boldsymbol{\mu}_c) = ||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_c)||^2 \tag{9.25}$$

The corresponding class posterior becomes

$$p(y = c|\boldsymbol{x}, \boldsymbol{\mu}, \mathbf{W}) = \frac{\exp(-\frac{1}{2}||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_c)||_2^2)}{\sum_{c'=1}^{C} \exp(-\frac{1}{2}||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_{c'})||_2^2)} \tag{9.26}$$

We can optimize $\mathbf{W}$ using gradient descent applied to the discriminative loss. This is called **nearest class mean metric learning** [Men+12]. The advantage of this technique is that it can be used for **one-shot learning** of new classes, since we just need to see a single labeled prototype $\boldsymbol{\mu}_c$ per class (assuming we have learned a good $\mathbf{W}$ already).

9.3 Naive Bayes classifiers

In this section, we discuss a simple generative approach to classification in which we assume the features are conditionally independent given the class label. This is called the **naive Bayes assumption**. The model is called "naive" since we do not expect the features to be independent, even conditional on the class label. However, even if the naive Bayes assumption is not true, it often results in classifiers that work well [DP97]. One reason for this is that the model is quite simple (it only has $O(CD)$ parameters, for C classes and D features), and hence it is relatively immune to overfitting.

More precisely, the naive Bayes assumption corresponds to using a class conditional density of the following form:

$$p(\boldsymbol{x}|y=c,\boldsymbol{\theta}) = \prod_{d=1}^{D} p(x_d|y=c,\boldsymbol{\theta}_{dc}) \tag{9.46}$$

where $\boldsymbol{\theta}_{dc}$ are the parameters for the class conditional density for class c and feature d. Hence the posterior over class labels is given by

$$p(y=c|\boldsymbol{x},\boldsymbol{\theta}) = \frac{p(y=c|\boldsymbol{\pi})\prod_{d=1}^{D} p(x_d|y=c,\boldsymbol{\theta}_{dc})}{\sum_{c'} p(y=c'|\boldsymbol{\pi})\prod_{d=1}^{D} p(x_d|y=c',\boldsymbol{\theta}_{dc'})} \tag{9.47}$$

where π_c is the prior probability of class c, and $\boldsymbol{\theta} = (\boldsymbol{\pi}, \{\boldsymbol{\theta}_{dc}\})$ are all the parameters. This is known as a **naive Bayes classifier** or **NBC**.

9.3.1 Example models

We still need to specify the form of the probability distributions in Equation (9.46). This depends on what type of feature x_d is. We give some examples below:

- In the case of binary features, $x_d \in \{0, 1\}$, we can use the Bernoulli distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \text{Ber}(x_d|\theta_{dc})$, where θ_{dc} is the probability that $x_d = 1$ in class c. This is sometimes called the **multivariate Bernoulli naive Bayes** model. For example, Figure 9.6 shows the estimated parameters for each class when we fit this model to a binarized version of MNIST. This approach does surprisingly well, and has a test set accuracy of 84.3%. (See Figure 9.7 for some sample predictions.)
- In the case of categorical features, $x_d \in \{1, \ldots, K\}$, we can use the categorical distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \text{Cat}(x_d|\boldsymbol{\theta}_{dc})$, where θ_{dck} is the probability that $x_d = k$ given that $y = c$.
- In the case of real-valued features, $x_d \in \mathbb{R}$, we can use the univariate Gaussian distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_{dc}, \sigma_{dc}^2)$, where μ_{dc} is the mean of feature d when the class label is c, and σ_{dc}^2 is its variance. (This is equivalent to Gaussian discriminant analysis using diagonal covariance matrices.)

9.3.2 Model fitting

In this section, we discuss how to fit a naive Bayes classifier using maximum likelihood estimation.

We can write the likelihood as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N_{\mathcal{D}}} \mathrm{Cat}(y_n|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_{nd}|y_n, \boldsymbol{\theta}_d) \tag{9.48}$$

$$= \prod_{n=1}^{N_{\mathcal{D}}} \mathrm{Cat}(y_n|\boldsymbol{\pi}) \prod_{d=1}^{D} \prod_{c=1}^{C} p(x_{nd}|\boldsymbol{\theta}_{d,c})^{\mathbb{I}(y_n=c)} \tag{9.49}$$

so the log-likelihood is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N_{\mathcal{D}}} \sum_{c=1}^{C} \mathbb{I}\left(y_n = c\right) \log \pi_c\right] + \sum_{c=1}^{C} \sum_{d=1}^{D} \left[\sum_{n:y_n=c} \log p(x_{nd}|\boldsymbol{\theta}_{dc})\right] \tag{9.50}$$

We see that this decomposes into a term for $\boldsymbol{\pi}$, and CD terms for each $\boldsymbol{\theta}_{dc}$:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log p(\mathcal{D}_y|\boldsymbol{\pi}) + \sum_{c} \sum_{d} \log p(\mathcal{D}_{dc}|\boldsymbol{\theta}_{dc}) \tag{9.51}$$

where $\mathcal{D}_y = \{y_n : n = 1 : N\}$ are all the labels, and $\mathcal{D}_{dc} = \{x_{nd} : y_n = c\}$ are all the values of feature d for examples from class c. Hence we can estimate these parameters separately.

In Section 4.2.4, we show that the MLE for $\boldsymbol{\pi}$ is the vector of empirical counts, $\hat{\pi}_c = \frac{N_c}{N}$. The MLEs for $\boldsymbol{\theta}_{dc}$ depend on the choice of the class conditional density for feature d. We discuss some common choices below.

- In the case of discrete features, we can use a categorical distribution. A straightforward extension of the results in Section 4.2.4 gives the following expression for the MLE:

$$\hat{\theta}_{dck} = \frac{N_{dck}}{\sum_{k'=1}^{K} N_{dck'}} = \frac{N_{dck}}{N_c} \tag{9.52}$$

  where $N_{dck} = \sum_{n=1}^{N} \mathbb{I}(x_{nd} = k, y_n = c)$ is the number of times that feature d had value k in examples of class c.

- In the case of binary features, the categorical distribution becomes the Bernoulli, and the MLE becomes

$$\hat{\theta}_{dc} = \frac{N_{dc}}{N_c} \tag{9.53}$$

  which is the empirical fraction of times that feature d is on in examples of class c.

- In the case of real-valued features, we can use a Gaussian distribution. A straightforward extension of the results in Section 4.2.5 gives the following expression for the MLE:

$$\hat{\mu}_{dc} = \frac{1}{N_c} \sum_{n:y_n=c} x_{nd} \tag{9.54}$$

$$\hat{\sigma}^2_{dc} = \frac{1}{N_c} \sum_{n:y_n=c} (x_{nd} - \hat{\mu}_{dc})^2 \tag{9.55}$$

Thus we see that fitting a naive Bayes classifier is extremely simple and efficient.

9.3.3 Bayesian naive Bayes

In this section, we extend our discussion of MLE estimation for naive Bayes classifiers from Section 9.3.2 to compute the posterior distribution over the parameters. For simplicity, let us assume we have categorical features, so $p(x_d|\boldsymbol{\theta}_{dc}) = \text{Cat}(x_d|\boldsymbol{\theta}_{dc})$, where $\theta_{dck} = p(x_d = k|y = c)$. In Section 4.6.3.2, we show that the conjugate prior for the categorical likelihood is the Dirichlet distribution, $p(\boldsymbol{\theta}_{dc}) = \text{Dir}(\boldsymbol{\theta}_{dc}|\boldsymbol{\beta}_{dc})$, where β_{dck} can be interpereted as a set of "**pseudo counts**", corresponding to counts N_{dck} that come from prior data. Similarly we use a Dirichlet prior for the label frequencies, $p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$. By using a conjugate prior, we can compute the posterior in closed form, as we explain in Section 4.6.3. In particular, we have

$$p(\boldsymbol{\theta}|\mathcal{D}) = \text{Dir}(\boldsymbol{\pi}|\,\widehat{\boldsymbol{\alpha}}) \prod_{d=1}^{D} \prod_{c=1}^{C} \text{Dir}(\boldsymbol{\theta}_{dc}|\,\widehat{\boldsymbol{\beta}}_{dc}) \tag{9.56}$$

where $\widehat{\alpha}_c = \breve{\alpha}_c + N_c$ and $\widehat{\beta}_{dck} = \breve{\beta}_{dck} + N_{dck}$.

Using the results from Section 4.6.3.4, we can derive the posterior predictive distribution as follows. The prior over the label is given by $p(y|\mathcal{D}) = \text{Cat}(y|\overline{\pi})$, where $\overline{\pi}_c = \widehat{\alpha}_c / \sum_{c'} \widehat{\alpha}_{c'}$. For the features, we have $p(x_d = k|y = c, \mathcal{D}) = \overline{\theta}_{dck}$, where

$$\overline{\theta}_{dck} = \frac{\widehat{\beta}_{dck}}{\sum_{k'} \widehat{\beta}_{dck'}} = \frac{\breve{\beta}_{dck} + N_{dck}}{\sum_{k'} \breve{\beta}_{dck'} + N_{dck'}} \tag{9.57}$$

is the posterior mean of the parameters.

If $\breve{\beta}_{dck} = 0$, this reduces to the MLE in Equation (9.52). By contrast, if we set $\breve{\beta}_{dck} = 1$, we add 1 to all the empirical counts before normalizing. This is called **add-one smoothing** or **Laplace smoothing**. For example, in the binary case, this gives

$$\overline{\theta}_{dc} = \frac{\breve{\beta}_{dc1} + N_{dc1}}{\breve{\beta}_{dc0} + N_{dc0} + \breve{\beta}_{dc1} + N_{dc1}} = \frac{1 + N_{dc1}}{2 + N_c} \tag{9.58}$$

Once we have estimated the parameter posterior, we can compute the predicted distribution over the label as follows:

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_d p(x_d|y = c, \mathcal{D}) = \overline{\pi}_c \prod_d \prod_k \overline{\theta}_{dck}^{\mathbb{I}(x_d = k)} \tag{9.59}$$

This gives us a fully Bayesian form of naive Bayes, in which we have integrated out all the parameters. (In this case, the predictive distribution can be obtained merely by plugging in the posterior mean parameters.)

9.4 Generative vs discriminative classifiers

A model of the form $p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x}|y)$ is called a **generative classifier**, since it can be used to generate examples $\boldsymbol{x}$ from each class y. By contrast, a model of the form $p(y|\boldsymbol{x})$ is called a **discriminative classifier**, since it can only be used to discriminate between different classes. Below we discuss various pros and cons of the generative and discriminative approaches to classification. (See also [BT04; UB05; LBM06; BL07a; Rot+18].)

9.4.1 Advantages of discriminative classifiers

The main advantages of discriminative classifiers are as follows:

- **Better predictive accuracy**. Discriminative classifiers are often much more accurate than generative classifiers [NJ02]. The reason is that the conditional distribution $p(y|\boldsymbol{x})$ is often much simpler (and therefore easier to learn) than the joint distribution $p(y, \boldsymbol{x})$, as illustrated in Figure 9.8. In particular, discriminative models do not need to "waste effort" modeling the distribution of the input features.
- **Can handle feature preprocessing**. A big advantage of discriminative methods is that they allow us to preprocess the input in arbitrary ways. For example, we can perform a polynomial expansion of the input features, and we can replace a string of words with embedding vectors (see Section 20.5). It is often hard to define a generative model on such pre-processed data, since the new features can be correlated in complex ways which are hard to model.
- **Well-calibrated probabilities**. Some generative classifiers, such as naive Bayes (described in Section 9.3), make strong independence assumptions which are often not valid. This can result in very extreme posterior class probabilities (very near 0 or 1). Discriminative models, such as logistic regression, are often better calibrated in terms of their probability estimates, although they also sometimes need adjustment (see e.g., [NMC05]).

9.4.2 Advantages of generative classifiers

The main advantages of generative classifiers are as follows:

- **Easy to fit.** Generative classifiers are often very easy to fit. For example, in Section 9.3.2, we show how to fit a naive Bayes classifier by simple counting and averaging. By contrast, logistic regression requires solving a convex optimization problem (see Section 10.2.3 for the details), and neural nets require solving a non-convex optimization problem, both of which are much slower.

- **Can easily handle missing input features.** Sometimes some of the inputs (components of $\boldsymbol{x}$) are not observed. In a generative classifier, there is a simple method for dealing with this, as we show in Section 1.5.5. However, in a discriminative classifier, there is no principled solution to this problem, since the model assumes that $\boldsymbol{x}$ is always available to be conditioned on.

- **Can fit classes separately.** In a generative classifier, we estimate the parameters of each class conditional density independently (as we show in Section 9.3.2), so we do not have to retrain the model when we add more classes. In contrast, in discriminative models, all the parameters interact, so the whole model must be retrained if we add a new class.

- **Can handle unlabeled training data.** It is easy to use generative models for semi-supervised learning, in which we combine labeled data $\mathcal{D}_{xy} = \{(\boldsymbol{x}_n, y_n)\}$ and unlabeled data, $\mathcal{D}_x = \{\boldsymbol{x}_n\}$. However, this is harder to do with discriminative models, since there is no uniquely optimal way to exploit $\mathcal{D}_x$.

- **May be more robust to spurious features**. A discriminative model $p(y|\boldsymbol{x})$ may pick up on features of the input $\boldsymbol{x}$ that can discriminate different values of y in the training set, but which are not robust and do not generalize beyond the training set. These are called **spurious features** (see e.g., [Arj21; Zho+21]). By contrast, a generative model $p(\boldsymbol{x}|y)$ may be better able to capture the causal mechanisms of the underlying data generating process; such causal models can be more robust to distribution shift (see e.g., [Sch19; LBS19; LN81]).

9.4.3 Handling missing features

Sometimes we are missing parts of the input $\boldsymbol{x}$ during training and/or testing. In a generative classifier, we can handle this situation by marginalizing out the missing values. (We assume that the missingness of a feature is not informative about its potential value.) By contrast, when using a discriminative model, there is no unique best way to handle missing inputs, as we discuss in Section 1.5.5.

For example, suppose we are missing the value of x_1. We just have to compute

$$p(y=c|\boldsymbol{x}_{2:D},\boldsymbol{\theta}) \propto p(y=c|\boldsymbol{\pi})p(\boldsymbol{x}_{2:D}|y=c,\boldsymbol{\theta}) \tag{9.63}$$

$$= p(y=c|\boldsymbol{\pi})\sum_{x_1} p(x_1,\boldsymbol{x}_{2:D}|y=c,\boldsymbol{\theta}) \tag{9.64}$$

In Gaussian discriminant analysis, we can marginalize out x_1 using the equations from Section 3.2.3.

If we make the naive Bayes assumption, things are even easier, since we can just ignore the likelihood term for x_1. This follows because

$$\sum_{x_1} p(x_1,\boldsymbol{x}_{2:D}|y=c,\boldsymbol{\theta}) = \left[\sum_{x_1} p(x_1|\boldsymbol{\theta}_{1c})\right]\prod_{d=2}^{D} p(x_d|\boldsymbol{\theta}_{dc}) = \prod_{d=2}^{D} p(x_d|\boldsymbol{\theta}_{dc}) \tag{9.65}$$

where we exploited the fact that $\sum_{x_1} p(x_1|y=c,\boldsymbol{\theta}_{1c}) = 1$.

Homework for next week

- Read the previous slides, which we could not cover due to time issues.
- Read Logistic Regression (10 full, 30 Pages)
- Read the start of Linear Regression (11-11.2, 10 Pages)

(There will always be a homework. Check ILIAS if in doubt.)