Kernels & Probabilistic Modeling

Mengye Ren

(Slides credit to David Rosenberg, He He, et al.)

NYU

October 1, 2024

Slides

Logistics

- Today (Oct 1): Kernels and Probabilistic Modeling
- Oct 8: Guest Lecture
- Oct 15: Homework 2 Due
- Oct 15: Legislative Day No Class
- Oct 22: Midterm, in class, closed-book, covers everything including Oct 8

Expressivity of Hypothesis Space

- For linear models, to grow the hypothesis spaces, we must add features.
- Sometimes we say a larger hypothesis is more expressive.
  - (can fit more relationships between input and action)
- Many ways to create new features.

Handling Nonlinearity with Linear Methods

Example Task: Predicting Health

- General Philosophy: Extract every feature that might be relevant
- Features for medical diagnosis
  - height
  - weight
  - body temperature
  - blood pressure
  - etc...

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Issues for Linear Predictors

- For linear predictors, it's important **how** features are added
  - The relation between a feature and the label may not be linear
  - There may be complex dependence among features
- Three types of nonlinearities can cause problems:
  - Non-monotonicity
  - Saturation
  - Interactions between features

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: The Issue

- Feature Map: $\phi(x) = [1, \text{temperature}(x)]$
- Action: Predict health score $y \in \mathbb{R}$ (positive is good)
- Hypothesis Space $\mathcal{F}$={affine functions of temperature}
- Issue:
  - Health is not an affine function of temperature.
  - Affine function can either say
    - Very high is bad and very low is good, or
    - Very low is bad and very high is good,
    - But here, both extremes are bad.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: Solution 1

- Transform the input:

$$\phi(x) = \left[1, \{\text{temperature(x)-37}\}^2\right],$$

  where 37 is "normal" temperature in Celsius.
- Ok, but requires manually-specified domain knowledge
  - Do we really need that?
  - What does $w^T \phi(x)$ look like?

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: Solution 2

- Think less, put in more:

$$\phi(x) = \left[1, \text{temperature}(x), \{\text{temperature}(x)\}^2\right].$$

- More expressive than Solution 1.

General Rule

Features should be simple building blocks that can be pieced together.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation: The Issue

- Setting: Find products relevant to user's query
- Input: Product x
- Output: Score the relevance of x to user's query
- Feature Map:

$$\phi(x) = [1, N(x)],$$

  where $N(x) =$ number of people who bought x.
- We expect a monotonic relationship between $N(x)$ and relevance, but also expect diminishing return.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation: Solve with nonlinear transform

- Smooth nonlinear transformation:

$$\phi(x) = [1, \log\{1 + N(x)\}]$$

  - $\log(\cdot)$ good for values with large dynamic ranges
- Discretization (a discontinuous transformation):

$$\phi(x) = (\mathbb{1}[0 \leqslant N(x) < 10], \mathbb{1}[10 \leqslant N(x) < 100], \ldots)$$

  - Small buckets allow quite flexible relationship

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: The Issue

- Input: Patient information x
- Action: Health score $y \in \mathrm{R}$ (higher is better)
- Feature Map

$$\phi(x) = [\text{height}(x), \text{weight}(x)]$$

- Issue: It's the weight *relative* to the height that's important.
- Impossible to get with these features and a linear classifier.
- Need some **interaction** between height and weight.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: Approach 1

- Google "ideal weight from height"
- J. D. Robinson's "ideal weight" formula:

$$\text{weight(kg)} = 52 + 1.9\,[\text{height(in)} - 60]$$

- Make score square deviation between height(h) and ideal weight(w)

$$f(x) = (52 + 1.9\,[h(x) - 60] - w(x))^2$$

- WolframAlpha for complicated Mathematics:

$$f(x) = 3.61h(x)^2 - 3.8h(x)w(x) - 235.6h(x) + w(x)^2 + 124w(x) + 3844$$

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: Approach 2

- Just include all second order features:

$$\phi(x) = \left[1, h(x), w(x), h(x)^2, w(x)^2, \underbrace{h(x)w(x)}_{\text{cross term}}\right]$$

- More flexible, no Google, no WolframAlpha.

General Principle

Simpler building blocks replace a single "smart" feature.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Monomial Interaction Terms

Interaction terms are useful building blocks to model non-linearities in features.

- Suppose we start with $x = (1, x_1, \ldots, x_d) \in \mathrm{R}^{d+1} = \mathcal{X}$.
- Consider adding all **monomials** of degree M: $x_1^{p_1} \cdots x_d^{p_d}$, with $p_1 + \cdots + p_d = M$.
  - Monomials with degree 2 in 2D space: x_1^2, x_2^2, $x_1 x_2$
- How many features will we end up with?

$$
\begin{aligned}
x_1^3 &\hookrightarrow \star\,\star\,\star\,|\,| \\
x_1^2 x_2 &\hookrightarrow \star\,\star\,|\,\star\,| \\
x_1^2 x_3 &\hookrightarrow \star\,\star\,|\,|\,\star \\
x_1 x_2^2 &\hookrightarrow \star\,|\,\star\,\star\,| \\
x_1 x_2 x_3 &\hookrightarrow \star\,|\,\star\,|\,\star \\
x_1 x_3^2 &\hookrightarrow \star\,|\,|\,\star\,\star \\
x_2^3 &\hookrightarrow |\,\star\,\star\,\star\,| \\
x_2^2 x_3 &\hookrightarrow |\,\star\,\star\,|\,\star \\
x_2 x_3^2 &\hookrightarrow |\,\star\,|\,\star\,\star \\
x_3^3 &\hookrightarrow |\,|\,\star\,\star\star
\end{aligned}
$$

Big Feature Spaces

This leads to extremely large data matrices

- For $d = 40$ and $M = 8$, we get 314457495 features.

Very large feature spaces have two potential issues:

- Overfitting
- Memory and computational costs

Solutions:

- Overfitting we handle with regularization.
- **Kernel methods** can help with memory and computational costs when we go to high (or infinite) dimensional spaces. »»»> 2fb8a69 (05)

The Kernel Trick

SVM with Explicit Feature Map

- Let $\psi : \mathcal{X} \to \mathrm{R}^d$ be a feature map.
- The SVM objective (with explicit feature map):

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T \psi(x_i)\right).$$

- Computation is costly if d is large (e.g. with high-degree monomials)
- Last time we mentioned an equivalent optimization problem from Lagrangian duality.

SVM Dual Problem

- By Lagrangian duality, it is equivalent to solve the following dual problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{c}{n}\right] \quad \forall i. \end{aligned}$$

- If α^* is an optimal value, then

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i) \qquad \text{and} \qquad \hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)^T \psi(x).$$

- Key observation: $\psi(x)$ only shows up in inner products with another $\psi(x')$ *for both training and inference*.

Compute the Inner Products

Consider 2D data. Let's introduce degree-2 monomials using $\psi : \mathbb{R}^2 \to \mathbb{R}^3$.

$$(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2).$$

The inner product is

$$\begin{aligned}
\psi(x)^T\psi(x') &= x_1^2 x_1'^2 + (\sqrt{2}x_1x_2)(\sqrt{2}x_1'x_2') + x_2^2 x_2'^2 \\
&= (x_1x_1')^2 + 2(x_1x_1')(x_2x_2') + (x_2x_2')^2 \\
&= (x_1x_1' + x_2x_2')^2 \\
&= (x^Tx')^2
\end{aligned}$$

We can calculate the inner product $\psi(x)^T\psi(x')$ in the original input space without accessing the features $\psi(x)$!

Compute the Inner Products

Now, consider monomials up to degree-2:

$$(x_1, x_2) \mapsto (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2).$$

The inner product can be computed by

$$\psi(x)^T\psi(x') = (1 + x^T x')^2 \quad \text{(check)}.$$

$$\begin{aligned}
\psi(x)^T\psi(x') &= 1 + 2x_1x_1' + 2x_2x_2' + x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\
&= 1 + 2(x_1x_1' + x_2x_2') + (x_1x_1' + x_2x_2')^2 \\
&= (1 + x^\top x')^2
\end{aligned}$$

More generally, for features maps producing monomials up to degree-p, we have

$$\psi(x)^T\psi(x') = (1 + x^T x')^p.$$

(Note that the coefficients of each monomial in ψ may not be 1)

Kernel Trick

Kernel trick: we do not need explicit features to calculate inner products.

- Using explicit features: $O(d^p)$
- Using implicit computation: $O(d)$

Kernel Function

The Kernel Function

- **Input space**: $\mathcal{X}$
- **Feature space**: $\mathcal{H}$
- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ is

$$k(x, x') = \langle \psi(x), \psi(x') \rangle,$$

  where $\langle \cdot, \cdot \rangle$ is the inner product associated with $\mathcal{H}$.

Why introduce this new notation $k(x, x')$?

- We can often evaluate $k(x, x')$ without explicitly computing $\psi(x)$ and $\psi(x')$.

When can we use the kernel trick?

Some Methods Can Be "Kernelized"

Definition

A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$. This applies to both the optimization problem and the prediction function.

The SVM Dual is a kernelization of the original SVM formulation.

Optimization:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \forall i. \end{aligned}$$

Prediction:

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)^T \psi(x).$$

The Kernel Matrix

Definition

The **kernel matrix** for a kernel k on $x_1, \ldots, x_n \in \mathcal{X}$ is

$$K = \big(k(x_i, x_j)\big)_{i,j} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \ldots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix} \in \mathrm{R}^{n \times n}.$$

- In ML this is also called a **Gram matrix**, but traditionally (in linear algebra), Gram matrices are defined without reference to a kernel or feature map.

The Kernel Matrix

- The kernel matrix summarizes all the information we need about the training inputs $x_1, \ldots, x_n$ to solve a kernelized optimization problem.
- In the kernelized SVM, we can replace $\psi(x_i)^T \psi(x_j)$ with K_{ij}:

$$\begin{aligned} \text{maximize}_\alpha \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{C}{n}\right] \; i = 1, \ldots, n. \end{aligned}$$

Given a kernelized ML algorithm (i.e. all $\psi(x)$'s show up as $\langle \psi(x), \psi(x') \rangle$),

- Can swap out the inner product for a new kernel function.
- New kernel may correspond to a very high-dimensional feature space.
- Once the kernel matrix is computed, the computational cost depends on number of data points n, rather than the dimension of feature space d.
- Useful when $d >> n$.
- Computing the kernel matrix may still depend on d and the essence of the **trick** is getting around this $O(d)$ dependence.

Example Kernels

Kernels as Similarity Scores

- Often useful to think of the $k(x, x')$ as a **similarity score** for x and x'.
- We can design similarity functions without thinking about the explicit feature map, e.g. "string kernels", "graph kernels".
- How do we know that our kernel functions actually correspond to inner products in some feature space?

How to Get Kernels?

- Explicitly construct $\psi(x) : \mathcal{X} \to \mathrm{R}^d$ (e.g. monomials) and define $k(x, x') = \psi(x)^T \psi(x')$.
- Directly define the kernel function $k(x, x')$ ("similarity score"), and verify it corresponds to $\langle \psi(x), \psi(x') \rangle$ for some ψ.

There are many theorems to help us with the second approach.

Linear Algebra Review: Positive Semidefinite Matrices

Definition

A real, symmetric matrix $M \in \mathrm{R}^{n \times n}$ is **positive semidefinite (psd)** if for any $x \in \mathrm{R}^n$,

$$x^T M x \geqslant 0.$$

Theorem

The following conditions are each necessary and sufficient for a symmetric matrix M to be positive semidefinite:

- *M can be factorized as $M = R^T R$, for some matrix R.*
- *All eigenvalues of M are greater than or equal to 0.*

Positive Definite Kernel

Definition

A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathrm{R}$ is a **positive definite (pd)** kernel on $\mathcal{X}$ if for any finite set $\{x_1, \ldots, x_n\} \in \mathcal{X}$ ($n \in \mathbb{N}$), the kernel matrix on this set

$$K = \big(k(x_i, x_j)\big)_{i,j} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

is a positive semidefinite matrix.

- Symmetric: $k(x, x') = k(x', x)$
- The kernel matrix needs to be positive semidefinite for any finite set of points.
- Equivalent definition: $\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geqslant 0$ given $\alpha_i \in \mathrm{R}\ \forall i$.

Mercer's Theorem

Theorem

A symmetric function $k(x, x')$ can be expressed as an inner product

$$k(x, x') = \langle \psi(x), \psi(x') \rangle$$

for some ψ if and only if $k(x, x')$ is ***positive definite****.*

- Proving a kernel function is positive definite is typically not easy.
- But we can construct new kernels from valid kernels.

Generating New Kernels from Old

- Suppose $k, k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are pd kernels. Then so are the following:

$$
\begin{aligned}
k_{\text{new}}(x, x') &= \alpha k(x, x') \quad \text{for } \alpha \geqslant 0 \quad \text{(non-negative scaling)} \\
k_{\text{new}}(x, x') &= k_1(x, x') + k_2(x, x') \quad \text{(sum)} \\
k_{\text{new}}(x, x') &= k_1(x, x') k_2(x, x') \quad \text{(product)} \\
k_{\text{new}}(x, x') &= k(\psi(x), \psi(x')) \text{ for any function } \psi(\cdot) \quad \text{(recursion)} \\
k_{\text{new}}(x, x') &= f(x) f(x') \text{ for any function } f(\cdot) \quad (f \text{ as 1D feature map})
\end{aligned}
$$

- Lots more theorems to help you construct new kernels from old.

Based on Mark Schmidt's slides: https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L12.5.pdf

Linear Kernel

- Input space: $\mathcal{X} = \mathrm{R}^d$
- Feature space: $\mathcal{H} = \mathrm{R}^d$, with standard inner product
- Feature map

$$\psi(x) = x$$

- Kernel:

$$k(x, x') = x^T x'$$

Quadratic Kernel in R^d

- Input space $\mathcal{X} = \mathrm{R}^d$
- Feature space: $\mathcal{H} = \mathrm{R}^D$, where $D = d + \binom{d}{2} \approx d^2/2$.
- Feature map:

$$\psi(x) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d)^T$$

- Then for $\forall x, x' \in \mathrm{R}^d$

$$\begin{aligned} k(x, x') &= \langle \psi(x), \psi(x') \rangle \\ &= \langle x, x' \rangle + \langle x, x' \rangle^2 \end{aligned}$$

- Computation for inner product with explicit mapping: $O(d^2)$
- Computation for implicit kernel calculation: $O(d)$.

Quadratic Kernel

$$
\begin{aligned}
k(x, x') &= \langle \psi(x), \psi(x') \rangle \\
&= \sum_{ij} x_i x_j' + \sum_{ij} x_i^2 x_j'^2 + 2 \sum_{ijkl} x_i x_j x_k' x_l' \\
&= \sum_{ij} x_i x_j' + \left(\sum_{ij} x_i x_j' \right)^2 \\
&= \langle x, x' \rangle + \langle x, x' \rangle^2
\end{aligned}
$$

Polynomial Kernel in R^d

- Input space $\mathcal{X} = \mathrm{R}^d$
- Kernel function:

$$k(x, x') = \left(1 + \langle x, x' \rangle\right)^M$$

- Corresponds to a feature map with all monomials up to degree M.
- For any M, computing the kernel has same computational cost
- Cost of explicit inner product computation grows rapidly in M.

Radial Basis Function (RBF) / Gaussian Kernel

Input space $\mathcal{X} = \mathrm{R}^d$

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right),$$

where σ^2 is known as the bandwidth parameter.

- Probably the most common nonlinear kernel.
- Does it act like a similarity score?
- Have we departed from our "inner product of feature vector" recipe?
  - Yes and no: corresponds to an infinite dimensional feature vector

Remaining Questions

Our current recipe:

- Recognize kernelized problem: $\psi(x)$ only occur in inner products $\psi(x)^T\psi(x')$
- Pick a kernel function ("similarity score")
- Compute the kernel matrix (n by n where n is the dataset size)
- Optimize the model and make predictions by accessing the kernel matrix

Next: When can we apply kernelization?

SVM solution is in the "span of the data"

- We found the SVM dual problem can be written as:

$$\begin{aligned} \sup_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \in \left[0, \frac{c}{n}\right] \; i = 1, \ldots, n. \end{aligned}$$

- Given dual solution α^*, primal solution is $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$.
- Notice: w^* is a linear combination of training inputs $x_1, \ldots, x_n$.
- We refer to this phenomenon by saying "w^* is in the **span of the data**."
  - Or in math, $w^* \in \text{span}(x_1, \ldots, x_n)$.

Ridge regression solution is in the "span of the data"

- The ridge regression solution for regularization parameter $\lambda > 0$ is

$$w^* = \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- This has a closed form solution:

$$w^* = \left(X^T X + \lambda I \right)^{-1} X^T y,$$

where X is the design matrix, with $x_1, \ldots, x_n$ as rows.

Ridge regression solution is in the "span of the data"

- Rearranging $w^* = \left(X^T X + \lambda I\right)^{-1} X^T y$, we can show that:

$$\begin{aligned} w^* &= X^T \underbrace{\left(\frac{1}{\lambda} y - \frac{1}{\lambda} X w^*\right)}_{\alpha^*} \\ &= X^T \alpha^* = \sum_{i=1}^{n} \alpha_i^* x_i. \end{aligned}$$

- So w^* is in the span of the data.
  - i.e. $w^* \in \text{span}\left(x_1, \ldots, x_n\right)$

Derivation

Ridge regression solution: $w^* = \left(X^T X + \lambda I\right)^{-1} X^T y$.
Lemma: If A and $A+B$ are non-singular, then $(A+B)^{-1} = A^{-1} - A^{-1}B(A+B)^{-1}$.

Let $\lambda I = A$, $X^T X = B$,

$$
\begin{aligned}
w^* &= \left(X^T X + \lambda I\right)^{-1} X^T y \\
&= (\lambda^{-1} - \lambda^{-1} X^T X (X^T X + \lambda I)^{-1}) X^T y \\
&= X^T \lambda^{-1} y - \lambda^{-1} X^T X (X^T X + \lambda I)^{-1} X^T y \\
&= X^T \lambda^{-1} y - \lambda^{-1} X^T X w^* \\
&= X^T \underbrace{\left(\frac{1}{\lambda} y - \frac{1}{\lambda} X w^*\right)}_{\alpha^*} \\
&= X^T \alpha^* = \sum_{i=1}^{n} \alpha_i^* x_i.
\end{aligned}
$$

Matrix Sum Inverse Lemma Derivation

Woodbury identity: $(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$

Let $C, V = I,\ U = B$,

$$\begin{aligned}(A+B)^{-1} &= A^{-1} - A^{-1}B(I+A^{-1}B)^{-1}A^{-1} \\ &= A^{-1} - A^{-1}B(A(I+A^{-1}B))^{-1} \\ &= A^{-1} - A^{-1}B(A+B)^{-1}.\end{aligned}$$

If solution is in the span of the data, we can reparameterize

- The ridge regression solution for regularization parameter $\lambda > 0$ is

$$w^* = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- We now know that $w^* \in \text{span}\,(x_1, \ldots, x_n) \subset \mathbb{R}^d$.
- So rather than minimizing over all of $\mathbb{R}^d$, we can minimize over $\text{span}\,(x_1, \ldots, x_n)$.

$$w^* = \arg\min_{w \in \text{span}(x_1, \ldots, x_n)} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- Let's reparameterize the objective by replacing w as a linear combination of the inputs.

If solution is in the span of the data, we can reparameterize

- Note that for any $w \in \text{span}(x_1, \ldots, x_n)$, we have $w = X^T \alpha$, for some $\alpha \in \mathbb{R}^n$.
- So let's replace w with $X^T \alpha$ in our optimization problem:

$$\begin{aligned} \text{[original]} \ w^* &= \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^n \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2 \\ \text{[reparameterized]} \ \alpha^* &= \underset{\alpha \in \mathbb{R}^n}{\arg\min} \frac{1}{n} \sum_{i=1}^n \left\{ \left(X^T \alpha \right)^T x_i - y_i \right\}^2 + \lambda \|X^T \alpha\|_2^2. \end{aligned}$$

- To get w^* from the reparameterized optimization problem, we just take $w^* = X^T \alpha^*$.
- We changed the dimension of our optimization variable from d to n. Is this useful?

Consider very large feature spaces

- Suppose we have a 300-million dimension feature space [very large]
  - (e.g. using high order monomial interaction terms as features, as described last lecture)
- Suppose we have a training set of 300,000 examples [fairly large]
- In the original formulation, we solve a 300-million dimension optimization problem.
- In the reparameterized formulation, we solve a 300,000-dimension optimization problem.
- This is why we care about when the solution is in the span of the data.
- This reparameterization is interesting when we have more features than data ($d \gg n$).

More General

- For SVM and ridge regression, we found that the solution is in the span of the data.
- The Representer Theorem shows that this "span of the data" result occurs far more generally.

The Representer Theorem (Optional)

- **Generalized objective**:

$$w^* = \operatorname*{arg\,min}_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

- Representer theorem tells us we can look for w^* in the span of the data:

$$w^* = \operatorname*{arg\,min}_{w \in \operatorname{span}(x_1, \ldots, x_n)} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle).$$

- So we can reparameterize as before:

$$\alpha^* = \operatorname*{arg\,min}_{\alpha \in \mathbb{R}^n} R\left(\left\|\sum_{i=1}^n \alpha_i x_i\right\|\right) + L\left(\left\langle \sum_{i=1}^n \alpha_i x_i, x_1 \right\rangle, \ldots, \left\langle \sum_{i=1}^n \alpha_i x_i, x_n \right\rangle\right).$$

- Our reparameterization trick applies much more broadly than SVM and ridge.

Summary

- We formualte the kernelized verions of SVM and ridge regression.
- Many other algorithms can be kernelized.
- Our principled tool for kernelization is reparameterization by the representer theorem.
- Representer theorem says that all norm-regularized linear models can be kernelized.
- Once kernelized, we can apply the kernel trick: doesn't need to represent $\phi(x)$ explicitly.

Probabilistic Modeling: Overview

Why probabilistic modeling?

- A unified framework that covers many models, e.g., linear regression, logistic regression
- Learning as **statistical inference**
- Principled ways to incorporate your belief on the data generating distribution (inductive biases)

Two ways of generating data

- Two ways to model how the data is generated:
  - **Conditional**: $p(y \mid x)$
  - **Generative**: $p(x, y)$
- How to estimate the parameters of our model? Maximum likelihood estimation.
- Compare and contrast conditional and generative models.

Conditional models

Linear regression

Linear regression is one of the most important methods in machine learning and statistics.

Goal: Predict a real-valued **target** y (also called response) from a vector of **features** x (also called covariates).

Examples:

- Predicting house price given location, condition, build year etc.
- Predicting medical cost of a person given age, sex, region, BMI etc.
- Predicting age of a person based on their photos.

Problem setup

Data Training examples $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

Model A *linear* function h (parametrized by θ) to predict y from x:

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x, \tag{1}$$

where $\theta \in \mathbb{R}^d$ are the **parameters** (also called weights).

Note that

- We incorporate the **bias term** (also called the intercept term) into x (i.e. $x_0 = 1$).
- We use superscript to denote the example id and subscript to denote the dimension id.

Parameter estimation

Loss function We estimate θ by minimizing the **squared loss** (the least square method):

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - \theta^T x^{(n)} \right)^2. \quad \text{(empirical risk)} \tag{2}$$

Matrix form

- Let $X \in \mathbb{R}^{N \times d}$ be the **design matrix** whose rows are input features.
- Let $\mathrm{y} \in \mathbb{R}^N$ be the vector of all targets.
- We want to solve

$$\hat{\theta} = \arg\min_{\theta} (X\theta - \mathrm{y})^T (X\theta - \mathrm{y}). \tag{3}$$

Solution Closed-form solution: $\hat{\theta} = (X^T X)^{-1} X^T \mathrm{y}$.

Review questions

- Derive the solution for linear regression.
- What if $X^T X$ is not invertible?

Review

We've seen

- Linear regression: response is a linear function of the inputs
- Estimate parameters by minimize the squared loss

But...

- Why squared loss is a reasonable choice for regression problems?
- What assumptions are we making on the data? (inductive bias)

Next,

- Derive linear regression from a probabilistic modeling perspective.

Assumptions in linear regression

- x and y are related through a linear function:

$$y = \theta^T x + \epsilon, \tag{4}$$

  where ϵ is the **residual error** capturing all unmodeled effects (e.g., noise).
- The errors are distributed *iid* (independently and identically distributed):

$$\epsilon \sim \mathcal{N}(0, \sigma^2). \tag{5}$$

What's the distribution of $Y \mid X = x$?

$$p(y \mid x; \theta) = \mathcal{N}(\theta^T x, \sigma^2). \tag{6}$$

Imagine putting a Gaussian bump around the output of the linear predictor.

Maximum likelihood estimation (MLE)

Given a probabilistic model and a dataset $\mathcal{D}$, how to estimate the model parameters θ?

The **maximum likelihood principle** says that we should maximize the (conditional) likelihood of the data:

$$L(\theta) \stackrel{\text{def}}{=} p(\mathcal{D}; \theta) \tag{7}$$

$$= \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; \theta). \qquad \text{(examples are distributed } \textit{iid}\text{)} \tag{8}$$

In practice, we maximize the **log likelihood** $\ell(\theta)$, or equivalently, minimize the negative log likelihood (NLL).

MLE for linear regression

Let's find the MLE solution for our model. Recall that $Y \mid X = x \sim \mathcal{N}(\theta^T x, \sigma^2)$.

$$\ell(\theta) \stackrel{\text{def}}{=} \log L(\theta) \tag{9}$$

$$= \log \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; \theta) \tag{10}$$

$$= \sum_{n=1}^{N} \log p(y^{(n)} \mid x^{(n)}; \theta) \tag{11}$$

$$= \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(n)} - \theta^T x^{(n)}\right)^2}{2\sigma^2}\right) \tag{12}$$

$$= N \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(y^{(n)} - \theta^T x^{(n)}\right)^2 \tag{13}$$

Gradient of the likelihood

Recall that we obtained the normal equation by setting the derivative of the squared loss to zero. Now let's compute the derivative of the likelihood w.r.t. the parameters.

$$\ell(\theta) = N \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(y^{(n)} - \theta^T x^{(n)} \right)^2 \tag{14}$$

$$\frac{\partial \ell}{\partial \theta_i} = -\frac{1}{\sigma^2} \sum_{n=1}^{N} (y^{(n)} - \theta^T x^{(n)}) x_i^{(n)}. \tag{15}$$

Review

We've seen

- Linear regression assumes that $Y \mid X = x$ follows a Gaussian distribution
- MLE of linear regression is equivalent to the least square method

However,

- Sometimes Gaussian distribution is not a reasonable assumption, e.g., classification
- Can we use the same modeling approach for other prediction tasks?

Next,

- Derive logistic regression for classification.

Assumptions in logistic regression

Consider binary classification where $Y \in \{0, 1\}$. What should be the distribution $Y \mid X = x$?

We model $p(y \mid x)$ as a Bernoulli distribution:

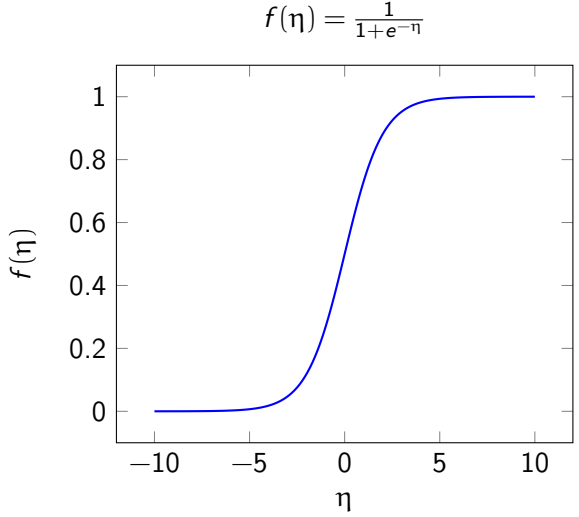
$$p(y \mid x) = h(x)^y (1 - h(x))^{1-y}. \tag{16}$$

How should we parameterize $h(x)$?

- What is $p(y = 1 \mid x)$ and $p(y = 0 \mid x)$? $h(x) \in (0, 1)$.
- What is the mean of $Y \mid X = x$? $h(x)$. (Think how we parameterize the mean in linear regression)
- Need a function f to map the linear predictor $\theta^T x$ in $\mathbb{R}$ to $(0, 1)$:

$$f(\eta) = \frac{1}{1 + e^{-\eta}} \qquad \text{logistic function} \tag{17}$$

Logistic regression



- $p(y \mid x) = \text{Bernoulli}(f(\theta^T x))$.
- When do we have $p(y = 1 \mid x) = 1$ and $p(y = 0 \mid x) = 1$?
- Exercise: show that the **log odds** is

$$\log \frac{p(y = 1 \mid x)}{p(y = 0 \mid x)} = \theta^T x. \tag{18}$$

$$\implies \text{linear decision boundary} \tag{19}$$

- How do we extend it to multiclass classification? (more on this later)

MLE for logistic regression

Similar to linear regression, let's estimate θ by maximizing the conditional log likelihood.

$$\ell(\theta) = \sum_{n=1}^{N} \log p(y^{(n)} \mid x^{(n)}; \theta) \tag{20}$$

$$= \sum_{n=1}^{N} y^{(n)} \log f(\theta^T x^{(n)}) + (1 - y^{(n)}) \log(1 - f(\theta^T x^{(n)})) \tag{21}$$

- Closed-form solutions are not available.
- But, the likelihood is concave—gradient ascent gives us the unique optimal solution.

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta). \tag{22}$$

Gradient descent for logistic regression

Math review: Chain rule

If z depends on y which itself depends on x, e.g., $z = (y(x))^2$, then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$.

Likelihood for a single example: $\ell^n = y^{(n)} \log f(\theta^T x^{(n)}) + (1 - y^{(n)}) \log(1 - f(\theta^T x^{(n)}))$.

$$\frac{\partial \ell^n}{\partial \theta_i} = \frac{\partial \ell^n}{\partial f^n}\frac{\partial f^n}{\partial \theta_i} \tag{23}$$

$$= \left(\frac{y^{(n)}}{f^n} - \frac{1 - y^{(n)}}{1 - f^n}\right)\frac{\partial f^n}{\partial \theta_i} \qquad \frac{d}{dx}\ln x = \frac{1}{x} \tag{24}$$

$$= \left(\frac{y^{(n)}}{f^n} - \frac{1 - y^{(n)}}{1 - f^n}\right)\left(f^n(1 - f^n)x_i^{(n)}\right) \qquad \text{Exercise: apply chain rule to } \frac{\partial f^n}{\partial \theta_i} \tag{25}$$

$$= (y^{(n)} - f^n)x_i^{(n)} \qquad \text{simplify by algebra} \tag{26}$$

The full gradient is thus $\frac{\partial \ell}{\partial \theta_i} = \sum_{n=1}^{N}(y^{(n)} - f(\theta^T x^{(n)}))x_i^{(n)}$.

A closer look at the gradient

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{n=1}^{N} (y^{(n)} - f(\theta^T x^{(n)})) x_i^{(n)} \tag{27}$$

- Does this look familiar?
- Our derivation for linear regression and logistic regression are quite similar...
- Next, a more general family of models.

Compare linear regression and logistic regression

	linear regression	logistic regression
Combine the inputs	$\theta^T x$ (linear)	$\theta^T x$ (linear)
Output	real	categorical
Conditional distribution	Gaussian	Bernoulli
Transfer function $f(\theta^T x)$	identity	logistic
Mean $\mathbb{E}(Y \mid X = x; \theta)$	$f(\theta^T x)$	$f(\theta^T x)$

- x enters through a linear function.
- The main difference between the formulations is due to different conditional distributions.
- Can we generalize the idea to handle other output types, e.g., positive integers?

Construct a generalized regression model

Task: Given x, predict $p(y \mid x)$

Modeling:

- Choose a parametric family of distributions $p(y; \theta)$ with parameters $\theta \in \Theta$
- Choose a transfer function that maps a linear predictor in $\mathbb{R}$ to Θ

$$\underbrace{x}_{\in \mathrm{R}^d} \mapsto \underbrace{w^T x}_{\in \mathrm{R}} \mapsto \underbrace{f(w^T x)}_{\in \Theta} = \theta, \tag{28}$$

Learning: MLE: $\hat{\theta} \in \arg\max_\theta \log p(\mathcal{D}; \hat{\theta})$

Inference: For prediction, use $x \to f(w^T x)$

Example: Construct Poisson regression

Say we want to predict the number of people entering a restaurant in New York during lunch time.

- What features would be useful?
- What's a good model for number of visitors (the output distribution)?

Math review: Poisson distribution

Given a random variable $Y \in 0, 1, 2, \ldots$ following Poisson(λ), we have

$$p(Y = k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{29}$$

where $\lambda > 0$ and $\mathbb{E}[Y] = \lambda$.

The Poisson distribution is usually used to model the number of events occurring during a fixed period of time.

Example: Construct Poisson regression

We've decided that $Y \mid X = x \sim \text{Poisson}(\eta)$, what should be the transfer function f?

x enters linearly:

$$x \mapsto \underbrace{w^T x}_{\mathbb{R}} \mapsto \lambda = \underbrace{f(w^T x)}_{(0,\infty)}$$

Standard approach is to take

$$f(w^T x) = \exp\left(w^T x\right).$$

Likelihood of the full dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$:

$$\log p(y_i; \lambda_i) = [y_i \log \lambda_i - \lambda_i - \log(y_i!)] \tag{30}$$

$$\log p(\mathcal{D}; w) = \sum_{i=1}^{n} \left[y_i \log\left[\exp\left(w^T x_i\right)\right] - \exp\left(w^T x_i\right) - \log(y_i!)\right] \tag{31}$$

$$= \sum_{i=1}^{n} \left[y_i w^T x_i - \exp\left(w^T x_i\right) - \log(y_i!)\right] \tag{32}$$

Example: multinomial logistic regression

How to extend logistic regression to multiclass classification? Output: Bernoulli distribution $\rightarrow$ categorical distribution

- Parametrized by a probability vector $\theta = (\theta_1, \ldots, \theta_k) \in \mathbb{R}^k$:
  - $\sum_{i=1}^{k} \theta_i = 1$ and $\theta_i \geqslant 0$ for $i = 1, \ldots, k$
  - So $\forall y \in \{1, \ldots, k\}$, $p(y) = \theta_y$.
- From each x, we compute a linear score function for each class:

$$x \mapsto (\langle w_1, x \rangle, \ldots, \langle w_k, x \rangle) \in \mathbb{R}^k,$$

- What's the transfer function that maps this $\mathbb{R}^k$ vector into a probability? The **softmax function**:

$$(s_1, \ldots, s_k) \mapsto \theta = \left(\frac{e^{s_1}}{\sum_{i=1}^{k} e^{s_i}}, \ldots, \frac{e^{s_k}}{\sum_{i=1}^{k} e^{s_i}} \right).$$

Review

Recipe for contructing a conditional distribution for prediction:

1. Define input and output space (as for any other model).
2. Choose the output distribution $p(y \mid x; \theta)$ based on the task that belongs to the exponential family.
3. Choose the transfer function that maps $w^T x$ to a Θ.
4. The formal family is called "generalized linear models".

Learning:

- Fit the model by maximum likelihood estimation.
- Closed solutions do not exist in general, so we use gradient ascent.

Generative models

Review

We've seen

- Model the conditional distribution $p(y \mid x; \theta)$ using generalized linear models.
- (Previously) Directly map x to y, e.g., perceptron.

Next,

- Model the joint distribution $p(x, y; \theta)$.
- Predict the label for x as $\arg\max_{y \in \mathcal{Y}} p(x, y; \theta)$.

Generative modeling through the Bayes rule

Training:

$$p(x, y) = p(x \mid y)p(y) \tag{33}$$

Testing:

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} \qquad \text{Bayes rule} \tag{34}$$

$$\arg\max_y p(y \mid x) = \arg\max_y p(x \mid y)p(y) \tag{35}$$

Naive Bayes (NB) models

Let's consider binary text classification (e.g., fake vs genuine review) as a motivating example.

Bag-of-words representation of a document

- ["machine", "learning", "is", "fun", "."]
- $x_i \in \{0, 1\}$: whether the i-th word in our vocabulary exists in the input

$$x = [x_1, x_2, \ldots, x_d] \quad \text{where } d = \text{vocabulary size} \tag{36}$$

What's the probability of a document x?

$$p(x \mid y) = p(x_1, \ldots, x_d \mid y) \tag{37}$$

$$= p(x_1 \mid y)p(x_2 \mid y, x_1)p(x_3 \mid y, x_2, x_1) \ldots p(x_d \mid y, x_{d-1}, \ldots, x_1) \quad \text{chain rule} \tag{38}$$

$$= \prod_{i=1}^{d} p(x_i \mid y, x_{<i}) \tag{39}$$

Naive Bayes assumption

Challenge: $p(x_i \mid y, x_{<i})$ is hard to model (and estimate), especially for large i.

Solution:

Naive Bayes assumption

Features are **conditionally independent** given the label:

$$p(x \mid y) = \prod_{i=1}^{d} p(x_i \mid y). \tag{40}$$

A strong assumption in general, but works well in practice.

Parametrize $p(x_i \mid y)$ and $p(y)$

For binary x_i, assume $p(x_i \mid y)$ follows Bernoulli distributions.

$$p(x_i = 1 \mid y = 1) = \theta_{i,1}, \;\; p(x_i = 1 \mid y = 0) = \theta_{i,0}. \tag{41}$$

Similarly,

$$p(y = 1) = \theta_0. \tag{42}$$

Thus,

$$\begin{aligned} p(x, y) &= p(x \mid y) p(y) && (43)\\ &= p(y) \prod_{i=1}^{d} p(x_i \mid y) && \text{NB assumption} \quad (44)\\ &= p(y) \prod_{i=1}^{d} \theta_{i,y} \mathbb{I}\{x_i = 1\} + (1 - \theta_{i,y}) \mathbb{I}\{x_i = 0\} && (45) \end{aligned}$$

Indicator function $\mathbb{I}\{\text{condition}\}$ evaluates to 1 if "condition" is true and 0 otherwise.

MLE for our NB model

We maximize the likelihood of the data $\prod_{n=1}^{N} p_\theta(x^{(n)}, y^{(n)})$ (as opposed to the *conditional* likelihood we've seen before).

Exercise:

Set $\frac{\partial}{\partial \theta_{j,1}} \ell$ to zero, show that $\theta_{j,1} = \frac{\sum_{n=1}^{N} \mathbb{I}\left\{y^{(n)}=1 \wedge x_j^{(n)}=1\right\}}{\sum_{n=1}^{N} \mathbb{I}\left\{y^{(n)}=1\right\}}$.

In practice, count words:

$$\frac{\text{number of fake reviews containing ``absolutely''}}{\text{number of fake reviews}}$$

Similarly, $\theta_{j,0} = \frac{\sum_{n=1}^{N} \mathbb{I}\left\{y^{(n)}=0 \wedge x_j^{(n)}=1\right\}}{\sum_{n=1}^{N} \mathbb{I}\left\{y^{(n)}=0\right\}}, \theta_0 = \frac{\sum_{n=1}^{N} \mathbb{I}\left\{y^{(n)}=1\right\}}{N}$.

Review

NB assumption: conditionally independent features given the label

Recipe for learning a NB model:

1. Choose $p(x_i \mid y)$, e.g., Bernoulli distribution for binary x_i.
2. Choose $p(y)$, often a categorical distribution.
3. Estimate parameters by MLE (same as the strategy for conditional models) .

Next, NB with continuous features.

NB with continuous inputs

Let's consider a multiclass classification task with continuous inputs.

$$p(x_i \mid y) \sim \mathcal{N}(\mu_{i,y}, \sigma^2_{i,y}). \tag{46}$$

$$p(y = k) = \theta_k. \tag{47}$$

Likelihood of the data:

$$p(\mathcal{D}) = \prod_{n=1}^{N} p(y^{(n)}) \prod_{i=1}^{d} p(x_i^{(n)} \mid y^{(n)}) \tag{48}$$

$$= \prod_{n=1}^{N} \theta_{y^{(n)}} \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_{i,y^{(n)}}} \exp\left(-\frac{1}{2\sigma^2_{i,y^{(n)}}} \left(x_i^{(n)} - \mu_{i,y^{(n)}} \right)^2 \right). \tag{49}$$

MLE for Gaussian NB

Log likelihood:

$$\ell = \sum_{n=1}^{N} \log \theta_{y^{(n)}} + \sum_{n=1}^{N} \sum_{i=1}^{d} \log \frac{1}{\sqrt{2\pi}\sigma_{i,y^{(n)}}} - \frac{1}{2\sigma_{i,y^{(n)}}^2} \left(x_i^{(n)} - \mu_{i,y^{(n)}} \right)^2. \tag{50}$$

$$\frac{\partial}{\partial \mu_{j,k}} \ell = \frac{\partial}{\partial \mu_{j,k}} \sum_{n: y^{(n)}=k} -\frac{1}{2\sigma_{j,k}^2} \left(x_j^{(n)} - \mu_{j,k} \right)^2 \qquad \text{ignore irrelevant terms} \tag{51}$$

$$= \sum_{n: y^{(n)}=k} \frac{1}{\sigma_{j,k}^2} \left(x_j^{(n)} - \mu_{j,k} \right). \tag{52}$$

Set $\frac{\partial}{\partial \mu_{j,k}} \ell$ to zero:

$$\mu_{j,k} = \frac{\sum_{n: y^{(n)}=k} x_j^{(n)}}{\sum_{n: y^{(n)}=k} 1} = \text{sample mean of } x_j \text{ in class } k. \tag{53}$$

MLE for Gaussian NB

Show that

$$\sigma_{j,k}^2 = \frac{\sum_{n:y^{(n)}=k} \left(x_j^{(n)} - \mu_{j,k}\right)^2}{\sum_{n:y^{(n)}=k} 1} = \text{sample variance of } x_j \text{ in class } k. \tag{54}$$

$$\theta_k = \frac{\sum_{n:y^{(n)}=k} 1}{N}. \qquad \text{(class prior)} \tag{55}$$

Decision boundary of the Gaussian NB model

Is the Gaussian NB model a linear classifier?

$$\log \frac{p(y=1 \mid x)}{p(y=0 \mid x)} = \log \frac{p(x \mid y=1)p(y=1)}{p(x \mid y=0)p(y=0)} \tag{56}$$

$$= \log \frac{\theta_0}{1-\theta_0} + \sum_{i=1}^{d} \left(\log \sqrt{\frac{\sigma_{i,0}^2}{\sigma_{i,1}^2}} + \left(\frac{(x_i - \mu_{i,0})^2}{2\sigma_{i,0}^2} - \frac{(x_i - \mu_{i,1})^2}{2\sigma_{i,1}^2} \right) \right) \quad \text{quadratic} \tag{57}$$

assume that $\sigma_{i,0} = \sigma_{i,1} = \sigma_i, \ (\theta_0 = 0.5)$ (58)

$$= \sum_{i=1}^{d} \frac{1}{2\sigma_i^2} \left((x_i - \mu_{i,0})^2 - (x_i - \mu_{i,1})^2 \right) \tag{59}$$

$$= \sum_{i=1}^{d} \frac{\mu_{i,1} - \mu_{i,0}}{\sigma_i^2} x_i + \frac{\mu_{i,0}^2 - \mu_{i,1}^2}{2\sigma_i^2} \quad \text{linear} \tag{60}$$

Decision boundary of the Gaussian NB model

Assuming the variance of each feature is the same for both classes, we have

$$\log \frac{p(y=1 \mid x)}{p(y=0 \mid x)} = \sum_{i=1}^{d} \frac{\mu_{i,1} - \mu_{i,0}}{\sigma_i^2} x_i + \frac{\mu_{i,0}^2 - \mu_{i,1}^2}{2\sigma_i^2} \tag{61}$$

$$= \theta^T x \qquad \text{where else have we seen it?} \tag{62}$$

(63)

$$\theta_i = \frac{\mu_{i,1} - \mu_{i,0}}{\sigma_i^2} \qquad \text{for } i \in [1, d] \tag{64}$$

$$\theta_0 = \sum_{i=1}^{d} \frac{\mu_{i,0}^2 - \mu_{i,1}^2}{2\sigma_i^2} \qquad \text{bias term} \tag{65}$$

Naive Bayes vs logistic regression

	logistic regression	Gaussian naive Bayes
model type	conditional/discriminative	generative
parametrization	$p(y \mid x)$	$p(x \mid y)$, $p(y)$
assumptions on Y	Bernoulli	Bernoulli
assumptions on X	—	Gaussian
decision boundary	$\theta_{\text{LR}}^T x$	$\theta_{\text{GNB}}^T x$

Given the same training data, is $\theta_{\text{LR}} = \theta_{\text{GNB}}$?

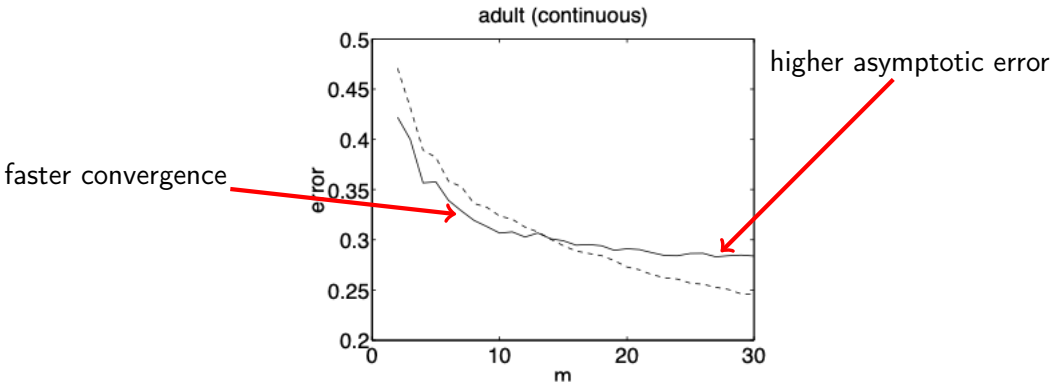
Naive Bayes vs logistic regression

Logistic regression and Gaussian naive Bayes converge to the same classifier asymptotically, assuming the GNB assumption holds.

- Data points are generated from Gaussian distributions for each class
- Each dimension is independently generated
- Shared variance

Generative vs discriminative classifiers

Ng, A. and Jordan, M. (2002). On discriminative versus generative classifiers: A comparison of logistic regression and naive Bayes. In Advances in Neural Information Processing Systems 14.

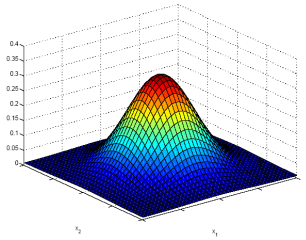


Solid line: naive Bayes; dashed line: logistic regression.

Multivariate Gaussian Distribution

- $x \sim \mathcal{N}(\mu, \Sigma)$, a Gaussian (or normal) distribution defined as

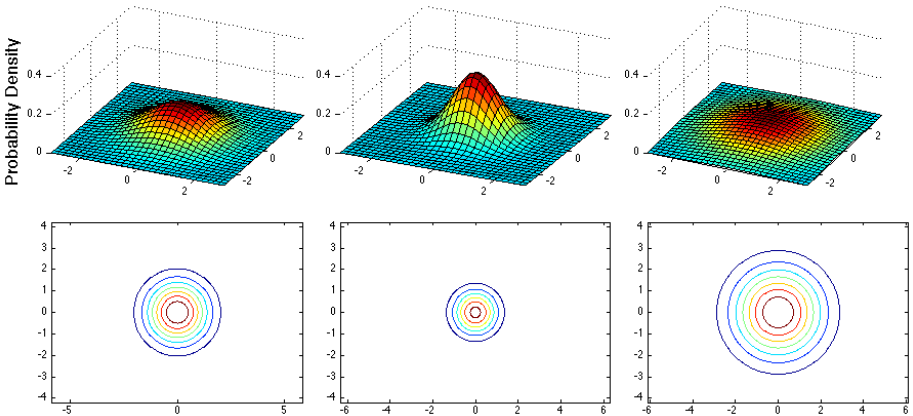
$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$



- Mahalanobis distance $(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)$ measures the distance from x to μ.
- It normalizes for difference in variances and correlations

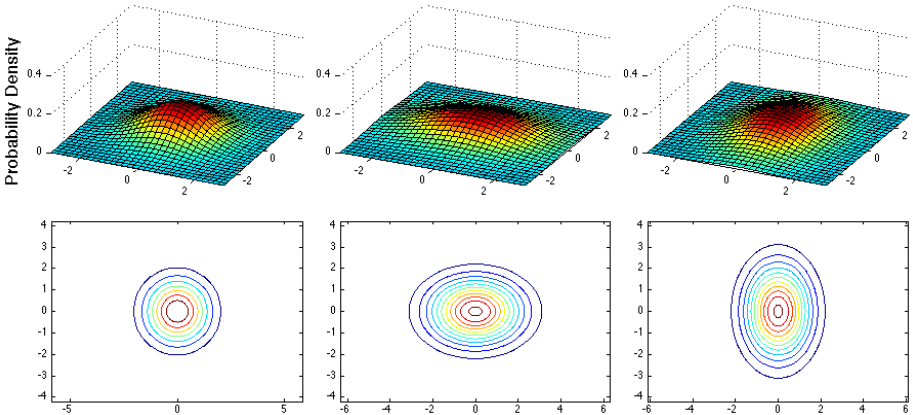
Bivariate Normal

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = 0.5 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



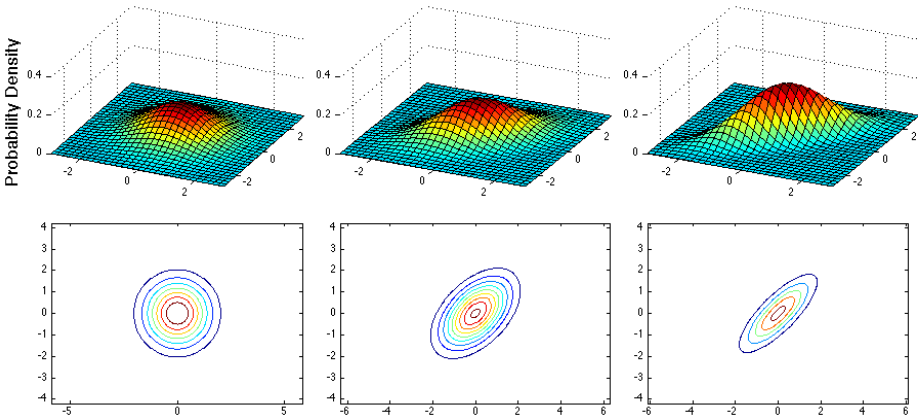
Bivariate Normal

$var(x_1) = var(x_2)$ $var(x_1) > var(x_2)$ $var(x_1) < var(x_2)$



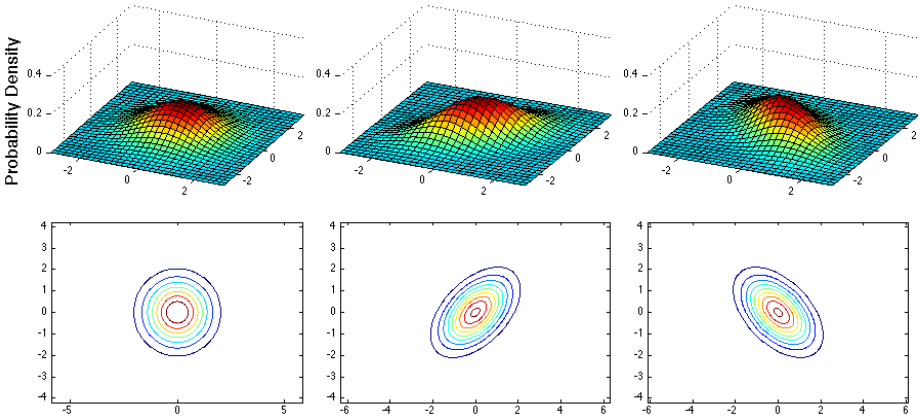
Bivariate Normal

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$



Bivariate Normal

$Cov(x_1, x_2) = 0$ $Cov(x_1, x_2) > 0$ $Cov(x_1, x_2) < 0$



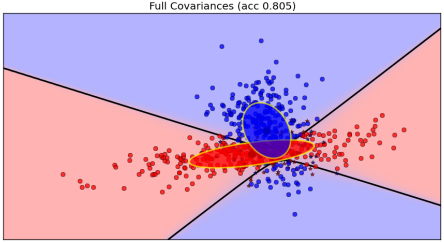
Gaussian Bayes Classifier

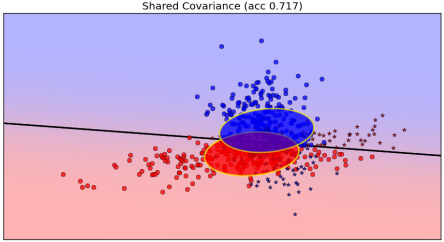
- Gaussian Bayes Classifier in its general form assumes that $p(\mathrm{x}|y)$ is distributed according to a multivariate normal (Gaussian) distribution
- Multivariate Gaussian distribution:

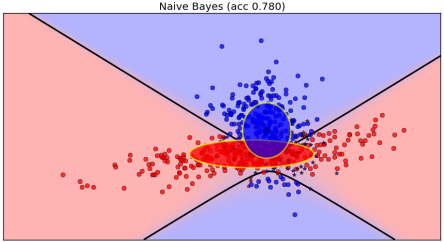
$$p(\mathrm{x}|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathrm{x}-\boldsymbol{\mu}_k)\right]$$

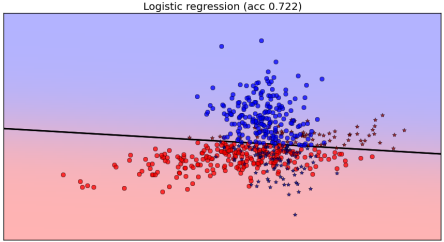
  where $|\Sigma_k|$ denotes the determinant of the matrix, and d is dimension of x
- Each class k has associated mean vector $\boldsymbol{\mu}_k$ and covariance matrix Σ_k
- Σ_k has $\mathcal{O}(d^2)$ parameters - could be hard to estimate

Example


Full Covariances (acc 0.805)


Shared Covariance (acc 0.717)


Naive Bayes (acc 0.780)


Logistic regression (acc 0.722)

Gaussian Bayes Binary Classifier Cases

Different cases on the covariance matrix:

- Full covariance: Quadratic decision boundary
- Shared covariance: Linear decision boundary
- Naive Bayes: Diagonal covariance matrix, quadratic decision boundary

GBC vs. Logistic Regression:

- If data is truly Gaussian distributed, then shared covariance = logistic regression.
- But logistic regression can learn other distributions.

Summary

- Probabilistic framework of using maximum likelihood as a more principled way to derive loss functions.
- Conditional vs. generative
- Generative models the joint distribution, and may lead to more assumption on the data.
- When there is very few data point, it may be hard to model the distribution.
- Is there an equivalent "regularization" in a probabilistic framework?