DS 1003 Machine Learning Lecture Notes

Long Chen
Center for Data Science, New York University
lc3424@nyu.edu

Spring 2021

Contents

1 Learning Theory

1.1 Decision Theory

We consider relationships between **action**, **input** and **outcome (label)**.

1.1.1 Evaluation Criterion

Decision theory is about finding "optimal" actions, under various definitions of optimality.
Examples:

- Is the classification correct?
- Does text transcription exactly match the spoken words?
  - Should we give partial credit? How?
- How far is the storm from the predicted location? (for point prediction)
- How likely is the storm's location under the predicted distribution? (for density prediction)

1.2 Formalization of ML Problem

Given input $x \in \mathcal{X}$, action $a \in \mathcal{A}$, outcome $\in \mathcal{Y}$, we have **prediction (decision) function** taking input and produces an action:

$$\begin{aligned} f: \ & \mathcal{X} \to \mathcal{A} \\ & x \mapsto f(x). \end{aligned}$$

And a loss function evaluating an action in the context of the outcome:

$$\begin{aligned} \ell: \ & \mathcal{A} \times \mathcal{Y} \to \mathbb{R} \\ & (a, y) \mapsto \ell(a, y) \end{aligned}$$

1.3 Statistical Learning Theory

1.3.1 Setup

Define the following space:

- Assume a data generating distribution $P_{\mathcal{X} \times \mathcal{Y}}$
- All input/output pairs (x, y) are generated i.i.d from $P_{\mathcal{X} \times \mathcal{Y}}$

We want prediction function $f(x)$ that "does well on average", i.e. $\ell(f(x), y)$ is usually small in some sense.

1.3.2 Risk

Definition The **risk** of a prediction function $f : \mathcal{X} \to \mathcal{A}$ is:

$$R(f) = \mathbb{E}_{(x,y)\sim P_{\mathcal{X}\times\mathcal{Y}}} \left[\ell(f(x), y)\right].$$

That is, the **expected loss** of f over $P_{\mathcal{X}\times\mathcal{Y}}$. Since we don't know $P_{\mathcal{X}\times\mathcal{Y}}$, this expectation cannot be computed. In practice, we try to estimate the data generating distribution.

1.3.3 Bayes Prediction Function

Definition A **Bayes prediction function** $f^*\mathcal{X} \to \mathcal{A}$ is a function that achieves the minimal risk among all possible functions:

$$f^* \in \arg\min_f R(f),$$

for all functions f from $\mathcal{X}$ to $\mathcal{A}$. The risk of a Bayes prediction function is called the **Bayes risk**. The Bayes prediction function can be regarded as the **"target function"** as this is essentially the best we can do. The Bayes risk, however, usually cannot be calculated in the absence of data generating distribution.

1.3.4 Empirical Risk

Assume we have sample data $\mathcal{D}_n = ((x_1, y_1), ..., (x_n, y_n))$ be drawn i.i.d from $P_{\mathcal{X}\times\mathcal{Y}}$. From the Strong Law of Large Numbers,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} z_i = \mathbb{E}\left[z\right]$$

with probability 1. So the sample data can `represent` the data generating distribution $P_{\mathcal{X}\times\mathcal{Y}}$.

Definition The **empirical risk** of $f : \mathcal{X} \to \mathcal{A}$ with respect to sample data $\mathcal{D}_n$ is:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)).$$

Which, according to the Strong Law of Large Numbers,

$$\lim_{n\to\infty} \hat{R}_n(f) = R(f),$$

with probability 1.

1.3.5 Empirical Risk Minimization (ERM)

Definition A function $\hat{f}$ is an **empirical risk minimizer** if

$$\hat{f} \in \arg\min_{f} \hat{R}_n(f).$$

1.3.6 Hypothesis Spaces

Definition A **hypothesis space** $\mathcal{F}$ is a set of candidate functions mapping $\mathcal{X} \to \mathcal{A}$ that we are choosing from. We generally want hypothesis spaces that:

- Include only those functions that have desired "regularity", e.g. smoothness, simplicity,
- Easy to work with.

1.3.7 Constrained ERM

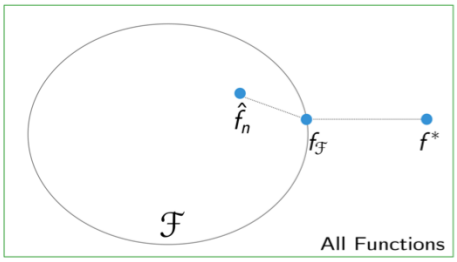
For functions mapping $f : \mathcal{X} \to \mathcal{A} \in \mathcal{F}$, **ERM** in $\mathcal{F}$ is:

$$\hat{f} \in \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)).$$

Risk minimizer in $\mathcal{F}$ is $f^*_{\mathcal{F}} \in \mathcal{F}$, where,

$$f^*_{\mathcal{F}} \in \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(f(x), y)\right].$$

1.4 Excess Risk Decomposition



Definition:

- **Approximation Error** (of $\mathcal{F}$) $= R(f_{\mathcal{F}}) - R(f^*)$
- **Estimation error** (of $\hat{f} \in \mathcal{F}$) $= R(\hat{f}_n) - R(f_{\mathcal{F}})$
- **Optimization Error** $= R(\tilde{f}_n) - R(\hat{f}_n)$

Where,

$$f^* = \arg\min_f \mathbb{E}\left[\ell(f(x), y)\right],$$

$$f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(f(x), y)\right],$$

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)).$$

Definition The **excess risk** compares the risk of f to the Bayes optimal f^*:

$$\textbf{Excess Risk}(f) = R(f) - R(f^*).$$

We show that:

$$\begin{aligned} \textbf{Excess Risk}(\hat{f}_n) &= R(\hat{f}_n) - R(f^*) \\ &= R(\hat{f}_n) - R(f_{\mathcal{F}}) + R(f_{\mathcal{F}}) - R(f^*), \end{aligned}$$

where the first two terms correspond to **estimation error** the the latter two terms correspond to **approximation error**.

Approximation error $R(\hat{f}_n) - R(f_{\mathcal{F}})$ is the penalty for restricting candidate functions to hypothesis space $\mathcal{F}$.

- Bigger $\mathcal{F} \rightarrow$ smaller approx. error,
- A **deterministic** variable.

Estimation error $R(f_{\mathcal{F}}) - R(f^*)$ is the performance hit for 1) using finite training data, and 2) for minimizing empirical risk rather than true risk.

- Smaller $\mathcal{F} \rightarrow$ smaller estimation error,
- A **random** variable, since the sampling procedure is random.

2 Gradient Descent

Let $f : \mathbb{R}^d \rightarrow R$ be differentiable at $x_0 \in \mathbb{R}^d$. The **gradient** of f at x_0, $\nabla_x f(x_0)$, is the direction of to move in for the **fastest increase** in $f(x)$ from x_0. Therefore, to minimize $f(x)$, we move in the opposite direction.

Algorithm 1 Gradient Descent

Initialize $w \leftarrow 0$
while not convergence condition **do**
  $w \leftarrow w - \eta \nabla f(x)$
return w

2.1 Step Size

For a fixed step size that is small enough, gradient descent will converge to optimal. For big step sizes, the algorithm might diverge.

Theorem Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable, and ∇f is **Lipschitz continuous** with $L > 0$, i.e.

$$\|\nabla f(x) - \nabla f(x')\| \leq L\|x - x'\|$$

for any $x, x' \in \mathbb{R}^d$. Then gradient descent with fixed step size $\eta \leq \frac{1}{L}$ converges, with:

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|^2}{2\eta k}.$$

Thus gradient descent is guaranteed to converge in $\mathcal{O}(\frac{1}{k})$.

2.2 Stopping Criterion

- **Theoretically:** wait until $\|\nabla f(x)\|_2 \leq \epsilon$, for some ϵ.
- **In practice:** stop when validation performance is not improving or is dropping.

2.3 Mini-batch and Stochastic Gradient Descent

As calculating full gradient for each epoch is costly, we may take a random sub-sample of size N from data $\mathcal{D}_n$. Thus the **minibatch gradient** is given by:

$$\nabla \hat{R}_N(w) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w \ell(f_w(x_{m_i}), y_{m_i})$$

Stochastic gradient is the minibatch gradient with $N = 1$. Minibatch gradient is an **unbiased estimator** of full gradient, i.e.

$$\mathbb{E}\left[\nabla \hat{R}_N(w)\right] = \nabla \hat{R}_n(w).$$

Also,

$$\frac{1}{N}\, Var\left[\nabla \hat{R}_1(w)\right] = Var\left[\nabla \hat{R}_N(w)\right],$$

implying that the bigger the minibatch, the better the estimate.

Algorithm 2 Minibatch Gradient Descent with batch size N

Initialize $w \leftarrow 0$
while not convergence condition **do**
randomly choose N points $(x_i, y_i)_{i=1}^N \subset \mathcal{D}_n$
$x \leftarrow x - \eta \left[\frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
return w

Intuitively, SGD/minibatch will struggle when close to the optimum. Thus we can 1) use diminishing step sizes (e.g. $\eta_k = \frac{1}{k}$, or 2) reduce learning rate when validation performance stops improving.

3 Loss Functions

For **regression** problems, a loss function is in the form:

$$(\hat{y}, y) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$$

Usually, $\ell(\hat{y}, y)$ concerns residual $r = \hat{y} - y$.

The **classification** loss function depends on a scoring function, $f(x)$. The magnitude of the score represents the confidence of our prediction.

Definition The **margin** for predicted score $\hat{y}$ and true class $y \in \{-1, 1\}$ is $y\hat{y}$, or $yf(x)$.

The margin measures how correct the scoring function is. Prediction is correct if margin is positive, i.e. $y, \hat{y}$ is of the same sign, and vice versa. For a classification problem, we aim to maximize the margin.

4 Regularization

4.1 Complexity of Hypothesis Spaces

With bigger $\mathcal{F}$, we approximate better (i.e. lower approximation error) but can overfit, vice versa.

4.1.1 Complexity Penalty

To balance between complexity of $\mathcal{F}$ and training loss, we introduce **complexity penalty**. We can express the constrained ERM problem in **Tikhonov** and **Ivanov** forms:

Constrained ERM (Tikhonov) For complexity measure $\Omega : \mathcal{F} \to [0, \infty]$ and fixed $\lambda \geq 0$,

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \lambda \Omega(f)$$

Constrained ERM (Ivanov) For complexity measure $\Omega : \mathcal{F} \to [0, \infty]$ and fixed $r \geq 0$,

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$
$$s.t.\ \Omega(f) \leq r$$

In most cases ($\lambda, r > 0$), Tikhonov and Ivanov form yield identical solutions.

4.2 Linear Regression with L2 Regularization

The original linear regression may overfit with hypothesis space of large dimension (e.g. in NLP). We therefore introduce **L2 regularization** that penalizes "large" weights:

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

where $\|w\|_2^2$ is the square of the ℓ_2-norm. This is also known as **ridge regression**. ℓ_2 regularization controls "sensitivity" of the function:

$$\begin{aligned} \left| \hat{f}(x_h) - \hat{f}(x) \right| &= \left| \hat{w}^T (x + h) - \hat{w}^T x \right| \\ &= \left| \hat{w}^T h \right| \\ &\leq \|\hat{w}\|_2 \|h\|_2 \end{aligned}$$

Where the last inequality comes from Cauchy-Schwarz inequality. We also have **lasso regression**, with ℓ_1-norm:

$$\hat{w} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \{w^T x_i - y_i\}^2 + \lambda \|w\|_1.$$

ℓ_1 regularization gives **feature sparsity** (check that derivative of objective function w.r.t. w "penalizes" a weight by a constant), which can be used as a feature selection tool. ℓ_2 regularization does **NOT** give a sparse solution (note that the derivative "penalizes" a weight proportional to x, thus usually the weight will not be reduced to 0). **Elastic net** combines both ℓ_2 and ℓ_1 penalties.

4.3 Feature Selection with Regularization

- For identical features
  - ℓ_1 spreads weight s arbitrarily with same signs
  - ℓ_2 spreads weight evenly
- For linearly dependent features
  - ℓ_1 chooses one variable **with larger scale** and assign 0 to the others
  - ℓ_2 spreads weight **proportional to scale**

5 Support Vector Machine

5.1 Perceptron

Definition For a **linearly separable** dataset $\mathcal{D}$, we can find a separating hyperplane such that $y_i(w^T x_i) > 0$ for all (x_i, y_i). The set $\{v \in \mathbb{R}^d | w^T v + b = 0\}$ is the **separating hyperplane**.

We can use the perceptron algorithm to find the separating hyperplane:

Algorithm 3 Perceptron Algorithm

```
Initialize w = 0
while exists misclassified examples do
    for (x_i, y_i) ∈ D do
        if y_i w^T x_i < 0 then
            update x ← w + y_i x_i
return w
```

Definition Let H be a separating hyperplane for dataset $\mathcal{D} = \{(x_i, y_i) | i = 1, ..., n\}$, the **geometric margin** of the hyperplane is the distance from the hyperplane to the closest data point:

$$\min_i d(x_i, H) = \min_i \frac{y_i(w^T x_i + b)}{\|w\|_2}$$

Intuitively, we have infinitely many satisfying, separating hyperplane from the perceptron algorithm given linearly separable dataset. We want to find the one that returns **maximum geometric margin**, thus:

$$\begin{aligned} &\text{maximize} \quad M \\ &\text{subject to} \quad \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq M \quad \text{for all } i \end{aligned}$$

Note that the solution to this problem is not unique, as we can scale w and b by any constant α. We thus fix the norm $\|w\|_2$ to $\frac{1}{M}$, and a convex optimization problem will form as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \quad \text{for all } i \end{aligned}$$

5.2 Soft Margin SVM

Introduce **slack variables (ξ)** to allow non-linearly separable dataset:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{for all } i \\ & \xi_i \geq 0 \text{for all } i \end{aligned}$$

We note that:

$$d(x_i, H) = \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq \frac{1 - \xi_i}{\|w\|_2}.$$

Thus $\xi_i > 0$ means that the point is wrongly classified and has a violation by multiples of the geometric margin. For example, $\xi_i = 1$ indicates the point lies on the separating hyperplane, while $\xi_i = 2$ indicates the point lies on the opposite margin line. Hyperparameter **C** indicates how much "penalty" the algorithm opposes to a misclassified datapoint. The larger the C, the narrower the margin.

5.3 Loss

For $m = yw^T x$, consider **perceptron loss:**

$$\ell(x, y, w) = max(0, -m)$$

The function is NOT differentiable at 0 and the derivative is 0 otherwise. This is not optimal for convex optimization. We introduce **hinge loss:**

$$\ell_{\text{Hinge}} = max\{1 - m, 0\} = (1 - m)_+$$

Hinge in **convex, upper bound** of perceptron loss, but is not differentiable at $m = 1$. We represent SVM solution using ERM with ℓ_2 regularization and hinge loss:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \xi_i \geq 1 - y_i(w^T x_i + b) \quad \text{for all } i \\ & \xi_i \geq 0 \quad \text{for all } i \end{aligned}$$

5.4 Subgradient

Consider convex and differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Then we for any $x, y \in \mathbb{R}^d$, we have first-order condition:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Naturally, the RHS linear approximation is a global underestimator of f. This implies that if $\nabla f(x) = 0$, x is a global minimizer of f. We utilize this intuition and define **subgradient** for a convex function that is not differentialbe everywhere:

Definition A vector $g \in \mathbb{R}^d$ is a **subgradient** of a convex function $f : \mathbb{R}^d \to \mathbb{R}$ at x if for all z,

$$f(z) \geq f(x) + g^T (z - x)$$

Definition The set of all subgradients at x is called the **subdifferential** $\partial f(x)$. Function f is **subdifferentialbe** at x if $\exists$ at least one subgradient at x.

For convex functions:

- f is differentiable at x iff $\partial f(x) = \{\nabla f(x)\}$
- Subdifferential is non-empty
- x is the global optimum iff $0 \in \partial f(x)$

For non-convex functions, subdifferential may be an empty set.

5.4.1 Subgradient Descent

We may exploit gradient descent using subgradient of a function:

$$x^{t+1} \leftarrow x^t - \eta g,$$

where $g \in \partial f(x^t)$ and $\eta > 0$. This can increase objective but will be geometrically closer to the minimizer if f is convex and η is small enough:

$$\|x^{t+1} - x^*\| < \|x^t - x^*\|.$$

We need to decreasing step sizes w.r.t epoch. Subgradient descent is slower than gradient descent. We will introduce stochastic subgradient descent with **Pegasos algorithm:**

Algorithm 4 Pegasos Algorithm

Input: $\lambda > 0$
Initialize $w_1 = 0, t = 0$
for $j = 1, ..., n$ **do**
  $t = t + 1$
  $\eta_t = 1/(t\lambda)$
  if $y_j w_t^T x_j < 1$ **then**
    $w_{t+1} \leftarrow (1 - \eta_t \lambda) w_t + \eta_t y_i x_i$
  else
    $w_{t+1} \leftarrow (1 - \eta_t \lambda) w_t$

5.5 Dual Solution

Definition The **Lagrange dual function** is:

$$g(\lambda) = \inf_x L(x, \lambda) = \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) \right),$$

such that $\lambda_i \geq 0,\ f_i(x) \leq 0$ for $\forall i$. For any optimization problem in primal form:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \quad \text{for all } i, \end{aligned}$$

we have a corresponding dual problem in lagrangian form. The dual function follows **lower bound property**, i.e. if $\lambda \succeq 0$, $g(\lambda) \leq p^*$, where p^* is the optimal value of the optimization problem.

5.5.1 SVM Dual Problem

We can express SVM optimization problem,

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & 1 - y_i(w^T x_i + b) - \xi_i \leq 0 \quad \text{for all } i \\ & -\xi_i \leq 0 \quad \text{for all } i \end{aligned}$$

in Lagrangian dual form. Let α_i be the lagrange multiplier for the first constraint and λ_i the multiplier for the second constraint. Thus we have:

$$L(w,b,\xi,\alpha,\lambda) = \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - y_i(w^T x_i + b) - \xi_i) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

We obtain dual optimal $d^* = \sup_{\alpha,\lambda\succeq 0}\inf_{w,b,\xi} L(w,b,\xi,\alpha,\lambda)$. This can be solved by using first order conditions:

$$\frac{\partial}{\partial w}L = 0 \Longleftrightarrow w = \sum_{i=1}^{n}\alpha_i y_i x_i$$

$$\frac{\partial}{\partial b}L = 0 \Longleftrightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\frac{\partial}{\partial \xi_i}L = 0 \Longleftrightarrow \alpha_i + \lambda_i = \frac{c}{n} \text{ for } \forall i$$

Using these results, we can transform SVM dual problem into the following:

$$\begin{aligned} \sup_{\alpha,\lambda} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_j^T x_i \\ s.t. \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n} \\ & \alpha_i, \lambda_i \geq 0 \text{ for } \forall i \end{aligned}$$

The SVM dual problem satisfies **Slater's constraint qualification**, i.e. Convex problem + affine constraints $\Longrightarrow$ strong duality iff problem is feasible. The SVM dual problem is feasible with at least one feasible point $(w,b,\xi) = (0,0,1)$. Therefore, we have strong duality in SVM.

5.5.2 Karush-Kuhn-Tucker (KKT) Conditions

Definition For convex problems, if Slater's condition is satisfied, then KKT condition provide necessary and sufficient conditions for the optimal solution:

- Primal feasibility: $f_i(x) \leq 0, \forall i$
- Dual feasibility: $\lambda \succeq 0$
- Complementary slackness: $\lambda_i f_i(x) = 0$
- First-order condition: $\frac{\partial}{\partial x}L(x,\lambda) = 0$

5.5.3 Complementary Slackness Condition (CSC)

Let x^* be primal optimal and λ^* be dual optimal. Assume **strong duality**, i.e. $x^* = \lambda^*$. Then:

$$\begin{aligned} f_0(x^*) = g(\lambda^*) &= \inf_x L(x, \lambda^*) \\ &\leq L(x^*, \lambda^*) \\ &= f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* f_i(x^*) \\ &\leq f_0(x^*). \end{aligned}$$

The last inequality holds since that $\lambda_i^* f_i(x^*) \leq 0$ for $\forall i$. Therefore, we further have $\lambda_i^* f_i(x^*) = 0$ for $\forall i$. We then have **complementary slackness Condition.**

$$\lambda_i > 0 \implies f_i(x^*) = 0 \text{ and } f_i(x^*) < 0 \implies \lambda_i = 0 \text{ for } \forall i.$$

This criterion guarantees that the values of the primal and dual are the same. Consider SVM dual problem, we derive the following equalities from CSC (also note that from first-order condition we have $\lambda_i^* = \frac{c}{n} - \alpha_i^*$ and $\alpha_i \in \left[0, \frac{c}{n}\right]$):

$$\begin{aligned} \alpha_i^*(1 - y_i f^*(x_i) - \xi_i^*) &= 0 \\ \lambda_i^* \xi_i^* = (\frac{c}{n} - \alpha_i^*)\xi_i^* &= 0 \end{aligned}$$

Recall that $\xi_i^* = \max(0, 1 - y_i f^*(x_i))$ is the hinge loss. We thus have the following consequences:

$$\begin{aligned} y_i f^*(x_i) > 1 &\implies \alpha_i^* = 0 \\ y_i f^*(x_i) = 1 &\implies \alpha_i^* \in \left[0, \frac{c}{n}\right] \\ y_i f^*(x_i) < 1 &\implies \alpha_i^* = \frac{c}{n} \\ \alpha_i^* = 0 &\implies y_i f^*(x) \geq 1 \\ \alpha_i^* \in \left(0, \frac{c}{n}\right) &\implies y_i f^*(x_i) = 1 \\ \alpha_i^* = \frac{c}{n} &\implies y_i f^*(x) \leq 1 \end{aligned}$$

Definition **Support Vectors** are x_i's corresponding to $\alpha_i > 0$.

Bias Term b* can be calculated using any (x_i, y_i) s.t. $\alpha_i \in \left(0, \frac{c}{n}\right)$ (note that in this case $\xi_i^* = 0$, also we have $y_i \in \{-1, 1\}$):

$$\begin{aligned} & y_i \left[x_i^T w^* + b^*\right] = 1 \\ \iff & x_i^T w^* + b^* = y_i \\ \iff & b^* = y_i - x_i^T w^* \end{aligned}$$

6 Feature Map

Definition Mapping an input from $\mathcal{X}$ to a vector in $\mathbb{R}^d$ is called **feature extraction** or **featurization**, $\phi : \mathcal{X} \to \mathbb{R}^d$. There are issues regarding linear features:

6.1 Non-monotonicity

Consider that a target variable may not be an affine function of features (e.g. temperature vs. fever). We might resolve this problem by 1) transforming features into ones using domain knowledge (e.g. transforming temperature into a variable of absolute difference from 37 degrees); or 2) put in more expressivity by expanding the spanning set (e.g. add square, square root).

6.2 Saturation

We might expect diminishing of return for a feature in larger scale. We may use non-linear transforms (e.g. log transformation, discretization).

6.3 Interaction

Some features may be inter-dependent on each other (e.g. height and weight). We have certain solutions to the problem:

- Feature engineering with domain knowledge. For example, if we build a health checker w.r.t height and weight, we may transform the features into BMI index.
- Adding higher-order monomial interaction terms (including cross terms).

7 Kernelization

We consider SVM objective with explicit feature map $psi : \mathcal{X} \to \mathbb{R}^d$:

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|_2^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T \psi(x_i)\right).$$

This can be computational costly for large d. Recall that this SVM objective can be transformed into a Lagrangian dual optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i) \\ s.t. \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n} \\ & \alpha_i, \lambda_i \geq 0 \text{ for } \forall i \end{aligned}$$

We also observe that:

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i),$$

and,

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

Intuitively, $\psi(x)$ is only present in inner products at training/inference time. We introduce **kernel** method. When calculating inner products for explicit features, we only need to access the original features. For example, for feature map $\psi(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)$, we have:

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

Note that the last expression does not have any explicit features. This is called the **"kernel trick"**. Another example will be monomials with degree-p:

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

We have a massive speed up using kernelization than using explicit features ($\mathcal{O}(d)$ vs. $\mathcal{O}(d)^p$).

Definition For an input space $\mathcal{X}$, feature space $\mathcal{H}$ and feature map $\psi : \mathcal{X} \to \mathcal{H}$, the **kernel function** corresponding to ψ is:

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

Definition The **kernel matrix (Gram matrix)** for a kernel k on $x_1, ..., x_n \in \mathcal{X}$ is :

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

We can thus **kernelize** SVM dual using a kernel matrix:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ij} \\ s.t. \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n} \\ & \alpha_i, \lambda_i \geq 0 \text{ for } \forall i \end{aligned}$$

Kernelization can be useful:

- Can swap to new kernel functions easily.
- New kernel may represent very high-dimensional feature space, which can be useful in cases.
- Time complexity after kernel function computed depends on the number of data points rather than the dimension of feature space, thus can be computationally efficient when $d >> n$.

7.1 Kernel Functions

We here discuss some kernel functions. We start with mathematical definitions:

7.1.1 Math for Kernelization

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

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

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

- M can be decomposed as $M = R^T R$ for some matrix R.
- M has non-negative eigenvalues.

Definition A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **positive definite (pd)** kernel on $\mathcal{X}$ if for **any** finite set $\{x_1, \ldots, x_n\} \in \mathcal{X}$, the kernel matrix on this set is a positive semidefinite matrix. We can also have alternative definition:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0 \text{ for } \alpha_i \in \mathbb{R} \ \forall i.$$

y

Mercer's Theorem A symmetric function $k(x, x')$ can be expressed as an inner product for some ψ iff $k(x, x')$ is positive definite.

With the aforementioned conditions, we need a positive definite kernel function for the inner product to be mathematically feasible. Some example kernels are:

- **Linear kernel:** $k(x, x') = x^T x'$
- **Polynormial kernel:** $k(x, x') = (1 + \langle x, x' \rangle)^M$
- **Radial Basis Function (RBF) / Gaussian Kernel:** $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$

In practice, proving a kernel function to be pd can be difficult, but kernels can be constructed using operations. For pd kernels $k, k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the followings are also positive definite:

- **Non-negative scaling:** $k_{\text{new}}(x, x') = \alpha k(x, x')$ for $\alpha \geq 0$
- **Sum:** $k_{\text{new}}(x, x') = k_1(x, x') + k_2(x, x')$
- **Product:** $k_{\text{new}}(x, x') = k_1(x, x') k_2(x, x')$
- **Recursion:** $k_{\text{new}}(x, x') = k(\psi(x), \psi(x'))$ for any $\psi(\cdot)$
- **1D feature map:** $k_{\text{new}}(x, x') = f(x) f(x')$ for any function $f(\cdot)$

7.2 Representer Theorem

Representer Theorem Let $J(w) = R(\|w\|) + L(\langle w, x_1\rangle, \ldots, \langle w, x_n\rangle)$, where,

- $w, x_1, \ldots, x_n \in \mathcal{H}$ for some Hilbert space $\mathcal{H}$
- $\|\cdot\|$ the norm corresponding to the inner product of $\mathcal{H}$
- Regularizatijon term $R : [0, \infty) \to R$ is non-decreasing
- Loss term $L : \mathbb{R}^n \to \mathbb{R}$ is arbitrary.

Then the objective function has a minimizer of the form:

$$w^* = \sum_{i=1}^{n} \alpha_i x_i.$$

Proof: For any $w \in \mathcal{H}$, let $w_M = Proj_M w$ for $\mathbf{M} = \mathbf{span}(\mathbf{x_1}, \ldots, \mathbf{x_n})$. Then $w - w_M$ is orthogonal to x for $\forall x \in M$. Thus:

$$\langle w, x_i\rangle = \langle w_M + w - w_M, x_i\rangle = \langle w_M, x_i\rangle + \langle w - w_M, x_i\rangle = \langle w_M, x_i\rangle \text{ for } \forall i.$$

We thus have:

$$L\left(\langle w, x_1\rangle, \ldots, \langle w, w_n\rangle\right) = L\left(\langle w_M, x_1\rangle, \ldots, \langle w_M, x_N\rangle\right).$$

Since projection decreases norm, i.e. $\|w_M\| \leq \|w\|$, and that R is non-decreasing,

$$R\left(\|w_M\|\right) \leq R\left(\|w\|\right).$$

Thus we conclude that:

$$J(w_M) \leq J(w).$$

Thus for any w^* that minimizes $J(w)$, $w_M^* = Proj_M w^*$ is also a minimizer of $J(w)$. Therefore, since that $\mathbf{M} = \mathbf{span}(\mathbf{x_1}, \ldots, \mathbf{x_n})$, $\exists\ \alpha$ s.t. $w_M^* = \sum_{i=1}^{n} \alpha_i x_i$ is a minimizer of $J(w)$. **(Q.E.D.)**

Theorem Let

$$J(w) = R(\|w\|) + L(\langle w, x_1\rangle, \ldots, \langle w, x_n\rangle),$$

and let $M = span(x_1, \ldots, x_n)$. Then under the same conditions given in the Representer theorem, if w_M^* minimizes $J(w)$ over set M, w_M^* minimizes $J(w)$ over all $\mathcal{H}$.

Proof of this theorem is trivial.

As a consequence of the Representer Theorem, we can reparameterize the objective function such that the minimizer is in the span of inputs:

$$J_0(\alpha) = R\left(\|\sum_{i=1}^{n} \alpha_i x_i\|\right) + L\left(s\left(\sum_{i=1}^{n} \alpha_i x_i\right)\right).$$

We have weight norm:

$$\begin{aligned}
\|w\|^2 &= \langle w, w \rangle \\
&= \langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{i=1}^{n} \alpha_i x_i \rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j x_i x_j \\
&= \alpha^T K \alpha,
\end{aligned}$$

and scoring function:

$$\begin{aligned}
s(w, x) &= s\left(\sum_{i=1}^{n} \alpha_i x_i\right) \\
&= \begin{pmatrix} \sum_{i=1}^{n} \alpha_i \langle x_i, x_1 \rangle \\ \vdots \\ \sum_{i=1}^{n} \alpha_i \langle x_i, x_n \rangle \end{pmatrix} \\
&= \begin{pmatrix} \alpha_1 \langle x_1, x_1 \rangle + \ldots + \alpha_n \langle x_n, x_1 \rangle \\ \vdots \\ \alpha_1 \langle x_1, x_n \rangle + \ldots + \alpha_n \langle x_n, x_n \rangle \end{pmatrix} \\
&= \begin{pmatrix} \langle x_1, x_1 \rangle & \ldots & \langle x_n, x_1 \rangle \\ \vdots & & \\ \langle x_1, x_n \rangle & \ldots & \langle x_n, x_n \rangle \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \\
&= K\alpha,
\end{aligned}$$

where K is the typical Kernel (Gram) matrix aforementioned. Therefore we have reparameterized, kernelized objective function:

$$J_0(\alpha) = R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha),$$

and we thus minimize over $\alpha \in \mathbb{R}^n$. Note that we are now minimizing $\mathbb{R}^n$ rather than $\mathbb{R}^d$ (when we are trying to minimize w in non-parameterized form), which can be computationally advantageous when $d >> n$ (e.g. in NLP). We can also make predictions with optimal α:

$$\begin{aligned}\hat{f}(x) = \langle w^*, x\rangle &= \sum_{i=1}^{n} \alpha_i^* \langle x_i, x\rangle \\ &= K_x^T \alpha^*,\end{aligned}$$

where K_x is the column kernel vector for $x \in \mathcal{H}$. Note that we need to use all training inputs $x_1, \ldots, x_n$ to make prediction.

Definition A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$.

We can kernelize many methods (e.g. SVM in dual form, ridge regression). We do not need to explicitly compute $\phi(x)$, as previously mentioned as the **"kernel trick"**.

8 Probabilistic Models

So far we assume that data is drawn i.i.d from a distribution $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$. Intuitively, this may not be the case for datasets in practice. We will therefore look into relaxing the constraint.

8.1 Probability Distribution

Let

- $p(y)$ be a probability distribution on $\mathcal{Y}$. Assume that $p(y)$ is either a probability density function (for a continuous space $\mathcal{Y}$) or a probability mass function (for a discrete space $\mathcal{Y}$).
- $\hat{p}(y)$ be an estimate of the probability distribution.
- $\mathcal{D} = (y_1, \ldots, y_n$ sampled i.i.d from true distribution $p(y)$.

Definition The **likelihood** of $\hat{p}$ for the data $\mathcal{D}$ is

$$\hat{p}(\mathcal{D}) = \prod_{i=1}^{n} \hat{p}(y_i).$$

8.2 Parametric Families of Distributions

Definition A **parametric model** is a set of probability distributions indexed by a parameter $\theta \in \Theta$, denoted as

$$\{p(y;\theta) | \theta \in \Theta\},$$

where θ is the **parameter** and Θ is the **parameter space**.

We will see examples of families of distributions for probability models:

Poisson Family Let $\mathcal{Y} = \{0, 1, 2, 3, \ldots\}$ and parameter space $\{\lambda \in \mathbb{R} | \lambda > 0\}$. Then the probablity mass function on $k \in \mathcal{Y}$ is:

$$p(k; \lambda) = \lambda^k \frac{e^{-\lambda}}{k!}.$$

Beta Family Let $\mathcal{Y} = \{0, 1\}$ and parameter space $\{\theta = (\alpha, \beta) | \alpha, \beta > 0\}$. Then the probability density function on $y \in \mathcal{Y}$ is:

$$p(y; \alpha, \beta) = \frac{y^{\alpha-1}(1-y)^{\beta-1}}{B(\alpha, \beta)}.$$

Gamma Family Let $\mathcal{Y} = \mathbb{R}^+$ and parameter space $\{\theta = (k, \theta) | k, \theta > 0\}$. Then the probability density function on $y \in \mathcal{Y}$ is:

$$p(y; k, \theta) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{\frac{-y}{\theta}}.$$

8.3 Maximum Likelihood Estimation

For a parametric model $\{p(y; \theta) | \theta \in \Theta\}$ and a sample $\mathcal{D} = (y_1, \ldots, y_n)$, the **likelihood** of parameter estimate $\hat{\theta} \in \Theta$ for sample $\mathcal{D}$ is:

$$p(\mathcal{D}; \hat{\theta}) = \prod_{i=1}^{n} p(y_i; \hat{\theta}).$$

In practice, we work with the **log-likelihood** to avoid computationally costly product operations:

$$\log p(\mathcal{D}; \hat{\theta}) = \sum_{i=1}^{n} \log p(y_i; \hat{\theta}).$$

Definition Let $\mathcal{D} = (y_1, \ldots, y_n)$ be an i.i.d sample from some distribution. Then a **maximum likelihood estimator (MLE)** for θ in the parametric model $\{p(y; \theta) | \theta \in \Theta\}$ is:

$$\begin{aligned} \hat{\theta} &\in \arg\max_{\theta \in \Theta} \log p(\mathcal{D}, \hat{\theta}) \\ &= \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log p(y_i; \theta). \end{aligned}$$

We can solve MLE as an optimization problem with closed form derivation (for certain models), or use numerical methods (e.g. SGD).

8.4 Conditional Distribution Estimation with Linear Probabilistic Classifiers

Task Given x, predict probability distribution $p(y|x)$. We achieve this by representing the objective as parametric families of distribution and train MLE of the model. Examples of such task are:

- Logistic / Probit regression (Bernoulli distribution)
- Poisson regression (Poisson distribution)
- Linear regression (Normal distribution with fixed variance)
- Gradient Boosting Machines (GBM)
- Many neural network models

8.4.1 Bernoulli Regression

Consider a generalized linear model where we have a linear predictor $x \mapsto x^T x \in \mathbb{R}$. If we want to convert such function into a probabilistic model, we want to map the result to $[0, 1]$ using **transfer function**:

$$x \mapsto x^T x \mapsto f(w^T x) = \theta.$$

We call $f : \mathbb{R} \to [0, 1]$ a **transfer function**. For example, the following two are transfer functions for Bernoulli distribution:

- Logistic function: $f(\eta) = \frac{1}{1+e^{-\eta}}$ for logistic regression
- Normal CDF $f(\eta) = \int_{-\infty}^{\eta} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ for probit regression

With such setting, we can use MLE to find w that maximizes $\log p(\mathcal{D}; w$. Equivalently, we can minimize the **negative log-likelihood**. For example in the case of Bernoulli family model, the objective function[1] is:

$$J(w) = -\left[\sum_{i=1}^{n} y_i \log f(w^T x_i) + (1 - y_i) \log \left[1 - f(w^T x_i)\right]\right],$$

which is differentiable given differentiable transfer function f and thus can be solved using closed form or gradient descent.

[1] Note that we have applied the probability "trick" to the objective function, or otherwise the function will be separately defined for $y = 0$ and $y = 1$.

8.4.2 Poisson Regression

Poisson regression differs from Bernoulli regression in that the output and action space is $\mathcal{Y} = \{0, 1, 2, 3, \ldots\}$. Let $\lambda \in (0, \infty)$ be the mean parameter for Poisson distribution. Then we have a transfer function:

$$x \mapsto w^T x \mapsto \lambda = f(w^T x),$$

for $f : \mathbb{R} \to (0, \infty)$. A standard transfer function is exponential. Thus for $lambda_i = f(w^T x_i) = \exp(w^T x_i)$, we have:

$$\log p(y_i; \lambda_i) = [y_i \log \lambda_i - \lambda_i - \log(yi!)].$$

Thus the likelihood for w on dataset $\mathcal{D}$ is:

$$\begin{aligned}
\log p(\mathcal{D}; w) &= \sum_{i=1}^{n} [y_i \log \lambda_i - \lambda_i - \log(yi!) \\
&= \sum_{i=1}^{n} [y_i \log \exp(w^T x_i) - \exp(w^T x_i) - \log(yi!) \\
&= \sum_{i=1}^{n} [y_i w^T x_i - \exp(w^T x_i) - \log(yi!).
\end{aligned}$$

Unlike Bernoulli distribution, this objective function does not have closed form. However, since it's concave, it's very easy to optimize.

8.4.3 Conditional Gaussian Regression

Consider Gaussian linear regression, where the output space is $y = \mathbb{R}$. We aim to find a prediction function that produces a distribution $\mathcal{N}(\mu, \sigma^2)$. If we assume σ to be known, the objective can be parameterized by the mean parameter $\mu \in \mathbb{R}$. Thus we have a mapping of input $x \in \mathbb{R}^d$:

$$x \mapsto w^T x \mapsto \mu = f(w^T x).$$

Since $\mu \in \mathbb{R}$, we can use identity transfer function $f(w^T x) = w^T x$. Then we have a likelihood for dataset $\mathcal{D}$:

$$p(\mathcal{D}; w) = \prod_{i=1}^{n} p(y_i | x_i; w).$$

We can solve this by maximizing log-likelihood:

$$
\begin{aligned}
&\sum_{i=1}^{n} \log p(y_i|x_i; w) \\
=&\sum_{i=1}^{n} \log \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2} \right) \right] \\
=&\sum_{i=1}^{n} \log \left[\frac{1}{\sigma\sqrt{2\pi}} \right] + \sum_{i=1}^{n} \left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2} \right)
\end{aligned}
$$

Note that σ is invariant w.r.t w. Thus we have a MLE as following:

$$
w^* = \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - w^T x_i)^2,
$$

which is essentially the objective function for least square.

8.4.4 Multinomial Logistic Regression

In this setting, we want to produce a distribution on k classes. We have a categorical distribution by a probability vector $\theta = (\theta_1, \ldots, \theta_k) \in \mathbb{R}^k$, where the ℓ_1 norm of the vector is 1 and all entries are non-negative. To do this, we introduce the **softmax function**:

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

Thus, for any $y \in \{1, \ldots, k\}$, we have:

$$
p(y|x; w) = \frac{\exp(w_y^T x)}{\sum_{i=1}^{k} \exp(w_i^T x)}.
$$

We thus aim to find $w's$ that maximize the log-likelihood on $\mathcal{D}$.

8.5 Maximum Likelihood as ERM

Consider a MLE for dataset $\mathcal{D}$:

$$
\hat{f}_{\text{MLE}} \in \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \log \left[f(x_i) y_i \right].
$$

We define loss for a predicted PDF or PMF $p(y)$ and outcome y as :

$$
\ell(p, y) = -\log p(y).
$$

Then the risk of decision function is:

$$
R(f) = -\mathbb{E}_{x,y} \log \left[f(x) y \right].
$$

And the empirical risk of f (or the negative conditional log-likelihood) is:

$$\hat{R}(f) = -\frac{1}{n}\sum_{i=1}^{n} \log\left[f(x_i)y_i\right].$$

We therefore observe equivalence between negative log-likelihood, ERM and MLE.

8.6 Bayesian Method

8.6.1 Prior Distribution

So far we assume parameter to be fixed with one model. With Bayesian statistics, we relax the condition by giving parameters a **prior distribution**, $p(\theta)$. With a parametric family of densities defined as in previous section, and a prior distribution on parameter space Θ, we get a joint density on θ and $\mathcal{D}$:

$$p(\mathcal{D}, \theta) = p(\mathcal{D}|\theta)p(\theta).$$

8.6.2 Posterior Distribution

The **posterior distribution** for θ is $p(\theta|\mathcal{D})$, representing a **rationally "updated" belief** about θ *after* observing $\mathcal{D}$. We have the expression for posterior distribution from Bayes rule:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}.$$

Definition Let π be a family of prior distributions on Θ, and P be a parametric family of distribution with parameter space Θ. Then a family of distribution π is **conjugate** to parametric model P if for any prior in π, the posterior is always in π.

Definition **Maximum a posteriori (MAP)** estimate is:

$$\hat{\theta} = \arg\max_{\theta} p(\theta|\mathcal{D}).$$

8.6.3 Bayesian Decision Theory

We consider the following settings:

- Parameter space Θ;
- Prior distribution $p(\theta)$ on Θ;
- Action space $\mathcal{A}$;
- Loss function $\ell : \mathcal{A} \times \Theta \to \mathbb{R}$.

Definition A **point estimation** is a statistic $\hat{\theta} = \hat{\theta}(\mathcal{D} \in \Theta$. A good point estimator will have $\hat{\theta} \approx \theta$ and has following properties:

- **Consistency:** $\hat{\theta}_n \to \theta$ as $n \to \infty$, where n is the size of dataset.
- **Efficiency:** $\hat{\theta}_n$ is as accurate as we can get from a sample of size n.

Intuitively, MLE are consistent and efficient under reasonable conditions.

Beyesian Point Estimation Let **prior** $p(\theta)$ on $\Theta = \mathbb{R}$ and **loss function** $\ell(\hat{\theta}, \theta)$. For a data $\mathcal{D}$ generated by $p(y|\theta)$ for unknown $\theta \in \Theta$, we aim to produce a point estimate for θ by finding an action $\theta \in \Theta$ that minimizes posterior risk:

$$\begin{aligned} r(\hat{\theta}) &= \mathbb{E}\left[\ell(\hat{\theta}, \theta)|\mathcal{D}\right] \\ &= \int \ell(\hat{\theta}, \theta) p(\theta|\mathcal{D}) d\theta. \end{aligned}$$

Loss Function and MLE Outcome For different losses, we will get different results from MLE algorithm:

- Square loss $\ell(\hat{\theta}, \theta) = (\theta - \hat{\theta})^2$ leads to **posterior mean**.
- 1-0 loss $\ell(\hat{\theta}, \theta) = \mathbb{1}(\theta \neq \hat{\theta})$ leads to **posterior mode**.
- Absolute loss $\ell(\hat{\theta}, \theta) = \left|\theta - \hat{\theta}\right|$ leads to **posterior median**.

Definition The $\hat{\theta}$ is called the **maximum a posteriori (MAP)** estimate.

8.7 Bayesian Regression

In the absence of observed data, we can produce a prediction function with Bayesian setting. The **prior predictive distribution** is:

$$x \mapsto p(y|x) = \int p(y|x;\theta) p(\theta) d\theta.$$

That is, an average of all conditional densities weighted by the prior. Such average is called a **mixture distribution.**

The **posterior predictive distribution** is:

$$x \mapsto p(y|x, \mathcal{D}) = \int p(y|x;\theta) p(\theta|\mathcal{D}) d\theta.$$

Empirically:

- $x \mapsto \mathbb{E}[y|x, \mathcal{D}]$ to minimize expected square error.
- $x \mapsto$ median $[y|x, \mathcal{D}]$ to minimize expected absolute error.
- $x \mapsto \arg\max_{y \in \mathcal{Y}} [y|x, \mathcal{D}]$ to minimize expected 0/1 loss.

9 Multi-Label Classification

9.1 Reduction to Binary Classification

9.1.1 One-vs-All (OvA)

For an input space $\mathcal{X}$ and output space $y = \{1, \ldots, k\}$, we train k binary classifiers, one for each class $h_1, \ldots, h_k : \mathcal{X} \to \mathbb{R}$. Intuitively, classifier h_i distinguishes class i from the rest. We then make a prediction by majority vote:

$$h(x) = \arg\max_{i \in \{1,\ldots,k\}} \; h_i(x).$$

9.1.2 All vs All (AvA)

For an input space $\mathcal{X}$ and output space $y = \{1, \ldots, k\}$, we train multiple binary classifiers such that each pair of classes have one binary classifier. Therefore, classifier h_{ij} distinguishes class i from class j. We then make a prediction by majority vote:

$$h(x) = \arg\max_{i \in \{1,\ldots,k\}} \sum_{j \neq i} h_{ij}(x)\mathbb{1}(i < j) - h_{ji}(x)\mathbb{1}(j < i).$$

Classifier h_{ij} has class i as $+1$, where h_{ij} has class i as -1. The expression is equivalent with $j = \{i + 1, \ldots, k\}$ under summation.

9.2 Multiclass Perceptron

Algorithm 5 Multiclass Perceptron

Input: multiclass dataset $\mathcal{D} = \{(x, y)\}$
Initialize $w \leftarrow 0$
for $iter = 1, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in Y} w_{y'}^T x$
    if $\hat{y} \neq y$ **then**
      $w_y \leftarrow w_y + x$
      $w_{\hat{y}} \leftarrow w_{\hat{y}} - x$
return w