CS 480/680 Winter 2024:
Lecture Notes

Lecture notes taken, unless otherwise specified, by myself during section 002 of the Winter 2024 offering of CS 480/680, taught by Hongyang Zheng.

Lectures

Chapter 1

Classic Machine Learning

1.1 Introduction

Lecture 1
Jan 9

There have been three historical AI booms:

1. 1950s–1970s: search-based algorithms (e.g., chess), failed when they realized AI is actually a hard problem
2. 1980s–1990s: expert systems
3. 2012 – present: deep learning

Machine learning is the subset of AI where a program can learn from experience.

Major learning paradigms of machine learning:

- Supervised learning: teacher/human labels answers (e.g., classification, ranking, etc.)
- Unsupervised learning: without labels (e.g., clustering, representation, generation, etc.)
- Reinforcement learning: rewards given for actions (e.g., gaming, pricing, etc.)
- Others: semi-supervised, active learning, etc.

Active focuses in machine learning research:

- Representation: improving the encoding of data into a space
- Generalization: improving the use of the model on new distributions
- Interpretation: understanding how deep learning actually works
- Complexity: improving time/space requirements
- Efficiency: reducing the amount of samples required
- Privacy: respecting legal/ethical concerns of data sourcing
- Robustness: gracefully failing under errors or malicious attack
- Applications

Lecture 2
Jan 11

A machine learning algorithm has three phases: training, prediction, and evaluation.

Definition 1.1.1 (dataset)

A dataset consists of a list of features $\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}'_1, \dots, \mathbf{x}'_m \in \mathbb{R}^d$ which are d-dimensional vectors and a label vector $\mathbf{y}^\top \in \mathbb{R}^n$.

Each training sample $\mathbf{x}_i$ is associated with a label y_i. A test sample $\mathbf{x}'_i$ may or may not be labelled.

Example 1.1.2 (email filtering). Suppose we have a list D of d English words.

Define the training set $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ and $\mathbf{y} = [y_1, \dots, y_n] \in \{\pm 1\}^n$ such that $\mathbf{x}_{ij} = 1$ if the word $j \in D$ appears in email i (this is the bag-of-words representation):

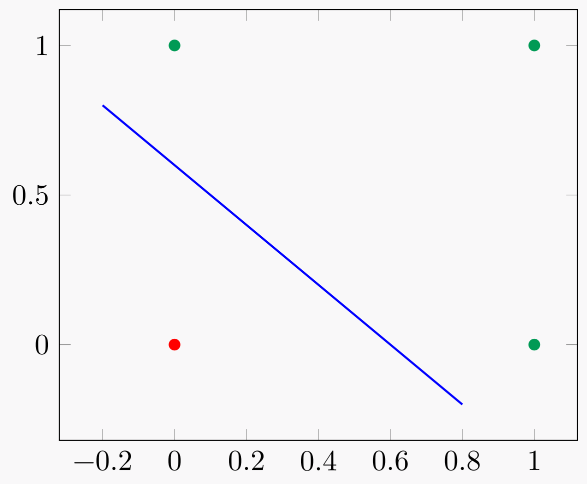
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}'$
and	1	0	0	1	1	1	1
viagra	1	0	1	0	0	0	1
the	0	1	1	0	1	1	0
of	1	1	0	1	0	1	0
nigeria	1	0	0	0	1	0	0
y	$+$	$-$	$+$	$-$	$+$	$-$	?

Then, given a new email $\mathbf{x}'_1$, we must determine if it is spam or not.

Example 1.1.3 (OR dataset). We want to train the OR function:

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
y	$-$	$+$	$+$	$+$

This can be represented graphically by finding a line dividing the points:



1.2 Perceptron

Definition 1.2.1

The inner product of vectors $\langle \mathbf{a}, \mathbf{b} \rangle$ is the sum of the element-wise product $\sum_j a_j b_j$.

A linear function is a function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that for all $\alpha, \beta \in \mathbb{R}$, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, $f(\alpha\mathbf{x}+\beta\mathbf{z}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{z})$.

Theorem 1.2.2 (linear duality)

A function is linear if and only if there exists $\mathbf{w} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$.

Proof. ($\Rightarrow$) Suppose f is linear. Let $\mathbf{w} := [f(\mathbf{e}_1), \dots, f(\mathbf{e}_d)]$ where $\mathbf{e}_i$ are coordinate vectors. Then:

$$\begin{aligned} f(\mathbf{x}) &= f(x_1\mathbf{e}_1 + \cdots + x_d\mathbf{e}_d) \\ &= x_1 f(\mathbf{e}_1) + \cdots + x_d f(\mathbf{e}_d) \\ &= \langle \mathbf{x}, \mathbf{w} \rangle \end{aligned}$$

by linearity of f.

($\Leftarrow$) Suppose there exists $\mathbf{w}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$. Then:

$$\begin{aligned} f(\alpha\mathbf{x} + \beta\mathbf{z}) &= \langle \alpha\mathbf{x} + \beta\mathbf{z}, \mathbf{w}, \alpha\mathbf{x} + \beta\mathbf{z}, \mathbf{w} \rangle \\ &= \alpha \langle \mathbf{x}, \mathbf{w} \rangle + \beta \langle \mathbf{x}, \mathbf{w} \rangle \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{z}) \end{aligned}$$

since inner products are linear in the first argument. □

Definition 1.2.3 (affine function)

A function $f(\mathbf{x})$ where there exist $\mathbf{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$.

Definition 1.2.4 (sign function)

$$\operatorname{sgn}(t) = \begin{cases} +1 & t > 0 \\ -1 & t \leq 0 \end{cases}$$

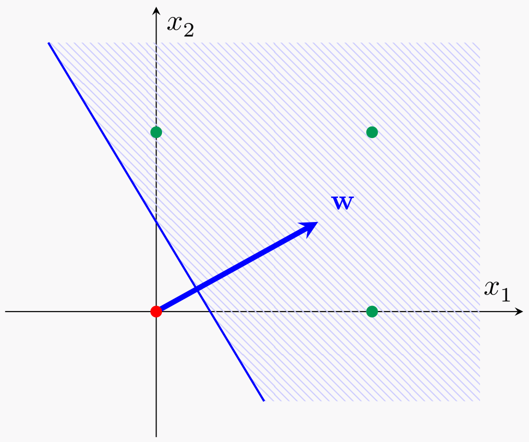
It does not matter what $\operatorname{sgn}(0)$ is defined as.

Definition 1.2.5 (linear classifier)

$\hat{y} = \operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$

The parameters $\mathbf{w}$ and b will uniquely determine the linear classifier.

Example 1.2.6 (geometric interpretation)**.** We can interpret $\hat{y} > 0$ as a halfspace (see CO 250). Then, we can draw something like:



Proposition 1.2.7

The vector $\mathbf{w}$ is orthogonal to the decision boundary H.

Proof. Let $\mathbf{x}, \mathbf{x}' \in H$ be vectors on the boundary $H = \{x : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$. Then, we must show $\mathbf{x}' - \mathbf{x} = \overrightarrow{\mathbf{x}\mathbf{x}'} \perp \mathbf{w}$.

We can calculate $\langle \mathbf{w}, \mathbf{x}' - \mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}, \mathbf{x}' \rangle = -b - (-b) = 0$. □

Originally, the inventor of the perceptron thought it could do anything. He was (obviously) wrong.

Algorithm 1 Training Perceptron

```
Require: Dataset $(\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\}$, initialization $\mathbf{w}_0 \in \mathbb{R}^d$, $b_0 \in \mathbb{R}$.
Ensure: $\mathbf{w}$ and $b$ for linear classifier $\operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$
  for $t = 1, 2, \dots$ do
      receive index $I_t \in \{1, \dots, n\}$
      if $\mathsf{y}_{I_t}\big(\langle \mathbf{x}_{I_t}, \mathbf{w} \rangle + b\big) \le 0$ then
          $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}_{I_t}\mathbf{x}_{I_t}$
          $b \leftarrow b + \mathsf{y}_{I_t}$
```

In a perceptron, we train by adjusting $\mathbf{w}$ and b whenever a training data feature is classified "wrong" (i.e., $\mathsf{score}_{\mathbf{w},b}(\mathbf{x}) := \mathsf{y}\hat{y} < 0 \iff$ the signs disagree).

The perceptron solves the feasibility problem

$$\text{Find } \mathbf{w} \in \mathbb{R}^d,\, b \in \mathbb{R} \text{ such that } \forall i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) > 0$$

by iterating one-by-one. It will converge "faster" (with fewer t-iterations) if the data is "easy".

Consider what happens when there is a "wrong" classification. Let $\mathbf{w}_{k+1} = w_k + \mathsf{y}\mathbf{x}$ and $b_{k+1} = b_k + \mathsf{y}$.

Then, the updated score is:

$$\begin{aligned}\mathsf{score}_{\mathbf{w}_{k+1},b_{k+1}}(\mathbf{x}) &= \mathsf{y}\cdot(\langle \mathbf{x},\mathbf{w}_{k+1}\rangle + b_{k+1}) \\ &= \mathsf{y}\cdot(\langle \mathbf{x},\mathbf{w}_k+\mathsf{y}\mathbf{x}\rangle + b_k + \mathsf{y}) \\ &= \mathsf{y}\cdot(\langle \mathbf{x},\mathbf{w}_k\rangle + b_k) + \langle \mathbf{x},\mathbf{x}\rangle + 1 \\ &= \mathsf{y}\cdot(\langle \mathbf{x},\mathbf{w}_k\rangle + b_k) + \underbrace{\|\mathbf{x}\|_2^2 + 1}_{\text{always positive}}\end{aligned}$$

which is always an increase over the previous "wrong" score.

↓ *Lectures 3 and 4 taken slides and Neysa since I was sick* ↓ *Lecture 3 Jan 16*

Instead of writing the affine function $\langle \mathbf{x},\mathbf{w}\rangle + b$, write $\langle \mathbf{x},\mathbf{w}\rangle = \left\langle \begin{pmatrix}\mathbf{x}\\1\end{pmatrix}, \begin{pmatrix}\mathbf{w}\\b\end{pmatrix}\right\rangle$.

Then, the update rule becomes $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}$.

Theorem 1.2.8 (convergence theorem)
Suppose there exists $\mathbf{w}^*$ such that $\mathsf{y}_i\langle \mathbf{x}_i,\mathbf{w}^*,\mathbf{x}_i,\mathbf{w}^*\rangle > 0$ for all i. Assume that $\|\mathbf{x}_i\|_2 \le C$ for all i, and we normalize the $\mathbf{w}^*$ such that $\|\mathbf{w}^*\|_2 = 1$. Define the margin $\gamma := \min_i |\langle \mathbf{x}_i,\mathbf{w}^*\rangle|$.

Then, the perceptron algorithm converges after C^2/γ^2 mistakes.

Proof. Recall the update on the mistake $(\mathbf{x},\mathsf{y})$ is $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}$.

Then, the inner product $\langle \mathbf{w},\mathbf{w}^*\rangle$ is

$$\begin{aligned}\langle \mathbf{w}+\mathsf{y}\mathbf{x},\mathbf{w}^*\rangle &= \langle \mathbf{w},\mathbf{w}^*\rangle + \mathsf{y}\langle \mathbf{x},\mathbf{w}^*\rangle \\ &= \langle \mathbf{w},\mathbf{w}^*\rangle + |\langle \mathbf{x},\mathbf{w}^*\rangle| \\ &\ge \langle \mathbf{w},\mathbf{w}^*\rangle + \gamma\end{aligned}$$

because $\mathsf{y}\langle \mathbf{x},\mathbf{w}^*\rangle$ must be positive if $\mathbf{w}^*$ is optimal. So for each update, $\langle \mathbf{w},\mathbf{w}^*\rangle$ grows by at least $\gamma > 0$. That is, after M updates, $\langle \mathbf{w},\mathbf{w}^*\rangle \ge M\gamma$.

Likewise, the inner product $\langle \mathbf{w},\mathbf{w}\rangle$ is

$$\begin{aligned}\langle \mathbf{w}+\mathsf{y}\mathbf{x},\mathbf{w}+\mathsf{y}\mathbf{x}\rangle &= \langle \mathbf{w},\mathbf{w}\rangle + \underbrace{2\mathsf{y}\langle \mathbf{w},\mathbf{x}\rangle}_{<0 \text{ because an update means it's wrong}} + \overbrace{\mathsf{y}^2\langle \mathbf{w},\mathbf{w}\rangle}^{\in[0,C^2] \text{ by construction}} \\ &\le \langle \mathbf{w},\mathbf{w}\rangle + C^2\end{aligned}$$

so each update grows $\langle \mathbf{w},\mathbf{w}\rangle$ by at most C^2, meaning that after M updates, $\langle \mathbf{w},\mathbf{w}\rangle \le MC^2$.

Finally, recall from linear algebra that $1 \ge \cos(\mathbf{w},\mathbf{w}^*) = \frac{\langle \mathbf{w},\mathbf{w}^*\rangle}{\|\mathbf{w}\|_2\|\mathbf{w}^*\|_2}$. Then,

$$\begin{aligned}1 &\ge \frac{\langle \mathbf{w},\mathbf{w}^*\rangle}{\|\mathbf{w}\|_2\cdot\|\mathbf{w}^*\|_2} \\ &\ge \frac{M\gamma}{\sqrt{MC^2}\cdot 1} \\ &= \sqrt{M}\frac{\gamma}{C}\end{aligned}$$

which implies $M \le C^2/\gamma^2$. □

Therefore, the larger the margin γ is, the more linearly separable the data is, and the faster the perceptron algorithm will converge.

Optimization perspective We can equivalently characterize the perceptron algorithm as an optimization problem. Given the linear classifier $\hat{y} = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$, we want to minimize the perceptron loss

$$
\begin{aligned}
\ell(\mathbf{w}, \mathbf{x}_t, \mathsf{y}_t) &= -\mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle \cdot \mathbb{I}[\text{mistake on } \mathbf{x}_t] \\
&= -\min\{\mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle, 0\} \\
L(\mathbf{w}) &= -\frac{1}{n} \sum_{t=1}^{n} (\mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle \cdot \mathbb{I}[\text{mistake on } \mathbf{x}_t])
\end{aligned}
$$

Then, the gradient descent update (see section 1.8) is

$$
\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w}_t, \mathbf{x}_t, \mathsf{y}_t) \\
&= \mathbf{w}_t + \eta_t \mathsf{y}_t \mathbf{x}_t \cdot \mathbb{I}[\text{mistake on } \mathbf{x}_t]
\end{aligned}
$$

With step size $\eta_t = 1$, we recover the update rule $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathsf{y}_t \mathbf{x}_t$.

Remark 1.2.9. The solution to perceptron is not unique, since there are many possible lines separating the data.

To pick the "best" line, we can maximize the margin γ. This leads to support vector machines (see sections 1.5 and 1.6).

Example 1.2.10 (XOR dataset)**.** Consider the XOR function

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
y	$-$	$+$	$+$	$+$

There is no separating hyperplane.

Proof. Suppose there exist $\mathbf{w}$ and b such that $\mathsf{y}(\langle \mathbf{x}, \mathbf{w} \rangle + b) > 0$. Then,

$$
\begin{aligned}
x_1 &= (0,0), \mathsf{y}_1 = - \implies b < 0 \\
x_2 &= (1,0), \mathsf{y}_2 = + \implies w_1 + b > 0 \\
x_3 &= (0,1), \mathsf{y}_3 = + \implies w_1 + b > 0 \implies w_1 + w_2 + 2b > 0 \\
x_4 &= (1,1), \mathsf{y}_4 = - \implies w_1 + w_2 + b < 0 \implies b > 0
\end{aligned}
$$

which is a contradiction. □

This leads us to a theorem.

Theorem 1.2.11
If there is no perfect separating hyperplane, then the perceptron algorithm cycles.

The proof is really complicated, and we will not cover it.

In this case, we can allow some wrong answers by setting a reasonable loss ℓ and regularizer reg:

$$\min_{\mathbf{w}} \hat{\mathbb{E}}[\ell(\mathsf{y}\hat{y}) + \operatorname{reg}(\mathbf{w})] \quad \text{s.t.} \quad \hat{y} := \langle \mathbf{x}, \mathbf{w} \rangle + b$$

We stop running perceptron when either:

- the maximum number of iterations is reached (i.e., we keep a constant maxiter),
- the maximum allowed runtime is reached,
- the training error stops changing, or
- the validation error stops decreasing.

If we have multiple classes (c of them), we can run perceptron as either one-vs.-all or one-vs.-one.

In one-vs.-all perceptron, for each class k, let it be positive, and all others be negative. We train weights $\mathbf{w}_k$ to get c imbalanced perceptrons. Then, predict according to the highest score

$$\hat{\mathsf{y}} := \arg\max_k \langle \mathbf{x}, \mathbf{w}_k \rangle .$$

In one-vs.-one perceptron, for each pair of classes (k, l), let k be positive, l be negative, and ignore all other classes. Then, train weights $\mathbf{w}_{k,l}$ for a total of $\binom{c}{2}$ balanced perceptrons. We predict by majority vote

$$\hat{\mathsf{y}} := \arg\max_k \sum_{l:l\neq k} \langle \mathbf{x}, \mathbf{w}_{k,l} \rangle .$$

1.3 Linear Regression

Problem 1.3.1 (regression)
Given training data $(\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^{d+t}$, find $f : \mathcal{X} \to \mathcal{Y}$ such that $f(\mathbf{x}_i) \approx \mathsf{y}_i$.

Lecture 4
Jan 18

The problem is that for finite training data, there are an infinite number of functions that exactly hit each point.

Theorem 1.3.2 (exact interpolation is always possible)
For any finite training data $(\mathbf{x}_i, \mathsf{y}_i) : i = 1, \ldots, n$ such that $\mathbf{x}_i \neq \mathbf{x}_j$ for all $i \neq j$, there exist infinitely many functions $f : \mathbb{R}^d \to \mathbb{R}^t$ such that for all i, $f(\mathbf{x}_i) = \mathsf{y}_i$.

TODO: ...up to slide 14 (geometry of linear regression)

$\uparrow$ *Lectures 3 and 4 taken from slides and Neysa since I was sick* $\uparrow$

Lecture 5
Jan 23

Theorem 1.3.3 (Fermat's necessary condition for optimality)
If $\mathbf{w}$ is a minimizer/maximizer of a differentiable function f over an open set, then $f'(\mathbf{w}) = \mathbf{0}$.

We can use this property to solve linear regression.

Recall the loss is $\text{Loss}(\mathbf{W}) = \frac{1}{n}\|\mathbf{WX} - \mathsf{Y}\|_F^2$. Then, the derivative $\nabla_{\mathbf{W}} \text{Loss}(\mathbf{W}) = \frac{2}{n}(\mathbf{WX} - \mathsf{Y})\mathbf{X}^\top$.

We can derive the normal equation:

$$\begin{aligned}
\frac{2}{n}(\mathbf{WX} - \mathsf{Y})\mathbf{X}^\top &= 0 \\
\mathbf{WX}\mathbf{X}^\top - \mathsf{Y}\mathbf{X}^\top &= 0 \\
\boxed{\mathbf{WX}\mathbf{X}^\top = \mathsf{Y}\mathbf{X}^\top} \\
\mathbf{W} &= \mathsf{Y}\mathbf{X}^\top(\mathbf{XX}^\top)^{-1}
\end{aligned}$$

Once we find $\mathbf{W}$, we can predict on unseen data $\mathbf{X}_{test}$ with $\hat{\mathsf{Y}}_{test} = \mathbf{WX}_{test}$.

Then,

Suppose $\mathbf{X} = \begin{bmatrix} 0 & \epsilon \\ 1 & 1 \end{bmatrix}$ and $\mathsf{y} = \begin{bmatrix} 1 & -1 \end{bmatrix}$.

Then, solving the linear least squares regression we get $\mathbf{w} = \mathsf{y}\mathbf{X}^\top(\mathbf{XX}^\top)^{-1} = \begin{bmatrix} -2/\epsilon & 1 \end{bmatrix}$. This is chaotic!

Why does this happen? As $\epsilon \to 0$, two columns in $\mathbf{X}$ become almost linearly dependent with incongruent corresponding y-values. This leads to a contradiction and an unstable $\mathbf{w}$.

To solve this, we add a $\lambda\|\mathbf{W}\|_F^2$ term.

Definition 1.3.4 (ridge regression)
Take the linear regression and add a regularization term:

$$\min_{\mathbf{W}} \frac{1}{n}\|\mathbf{WX} - \mathsf{Y}\|_F^2 + \lambda\|\mathbf{W}\|_F^2$$

This gives a new normal equation:

$$\begin{aligned}
\text{Loss}(\mathbf{W}) &= \frac{1}{n}\|\mathbf{WX} - \mathsf{Y}\|_F^2 + \lambda\|\mathbf{W}\|_F^2 \\
\nabla_{\mathbf{W}} \text{Loss}(\mathbf{W}) &= \frac{2}{n}(\mathbf{WX} - \mathsf{Y})\mathbf{X}^\top + 2\lambda\mathbf{W} \\
0 &= \frac{2}{n}(\mathbf{WX} - \mathsf{Y})\mathbf{X}^\top + 2\lambda\mathbf{W} \\
\boxed{\mathbf{W}(\mathbf{XX}^\top + n\lambda I) = \mathsf{Y}\mathbf{X}^\top} \\
\mathbf{W} &= \mathsf{Y}\mathbf{X}^\top(\mathbf{XX}^\top + n\lambda I)^{-1}
\end{aligned}$$

Proposition 1.3.5

$\mathbf{X}\mathbf{X}^\top + n\lambda I$ is far from rank-deficient for large λ.

Proof. Recall from linear algebra that we can always take the singular value decomposition of any matrix $M = U\Sigma V^\top$ where U and V are orthogonal and Σ is non-negative diagonal where the rank is the number of non-zero entries in Σ.

Consider the SVD of $\mathbf{X}$:

$$\begin{aligned} \mathbf{X} &= U\Sigma V^\top \\ \mathbf{X}\mathbf{X}^\top &= U\Sigma V^\top V \Sigma^\top U^\top = U\Sigma^2 U^\top \\ \mathbf{X}\mathbf{X}^\top + n\lambda I &= U\Sigma^2 U^\top + U(n\lambda I)U^\top \\ &= U(\Sigma^2 + n\lambda I)U^\top \end{aligned}$$

The matrix $\Sigma^2 + n\lambda I$ is a diagonal matrix with strictly positive elements for sufficiently large λ. Therefore, $\mathbf{X}\mathbf{X}^\top + n\lambda I$ has full rank and thus no singular values. □

Remark 1.3.6. Performing a ridge regularization is identical to augmenting the data.

Notice that

$$\frac{1}{n}\|\mathbf{W}\mathbf{X} - \mathsf{Y}\|_F^2 + \lambda\|\mathbf{W}\|_F^2 = \frac{1}{n}\left\|\mathbf{W}\begin{bmatrix}\mathbf{X} & \sqrt{n\lambda}I\end{bmatrix} - \begin{bmatrix}\mathsf{Y} & \mathbf{0}\end{bmatrix}\right\|_F^2$$

so if we augment $\mathbf{X}$ with $\sqrt{n\lambda}I$ and Y with $\mathbf{0}$, i.e., p data points $\mathbf{x}_j = \sqrt{n\lambda}\mathbf{e}_j$ and $\mathsf{y}_j = 0$.

1.4 Logistic Regression

Return to the linear classification problem.

Recall that we took $\hat{\mathsf{y}} = \operatorname{sgn}(\langle \mathbf{x}, \mathbf{w}\rangle)$ where $\mathbf{x} = \begin{pmatrix}\mathbf{x}\\1\end{pmatrix}$ and $\mathbf{w} = \begin{pmatrix}\mathbf{w}\\b\end{pmatrix}$ in $\mathbb{R}^{d+1}$.

How confident are we in our prediction $\hat{\mathsf{y}}$? We can use the margin (or logit) $|\langle \mathbf{x}, \mathbf{w}\rangle|$ ("how far away is the point from the decision boundary?").

The margin is unnormalized with respect to the data, so we cannot really interpret it until we somehow cram it into $[0, 1]$.

We can try directly learning hte confidence.

Let $\mathcal{Y} = \{0, 1\}$. Consider confidence $p(\mathbf{x}; \mathbf{w}) := \Pr[\mathsf{Y} = 1 \mid \mathsf{X} = \mathbf{x}]$. Given independent $(\mathbf{x}_i, \mathsf{y}_i)$:

$$\begin{aligned} &\Pr[\mathsf{Y}_1 = \mathsf{y}_1, \dots, \mathsf{Y}_n = \mathsf{y}_n \mid \mathsf{X}_1 = \mathbf{x}_1, \dots, \mathsf{X}_n = \mathbf{x}_n] \\ =& \prod_{i=1}^n \Pr[\mathsf{Y}_i = \mathsf{y}_i \mid \mathsf{X}_i = \mathbf{x}_i] \\ =& \prod_{i=1}^n [p(\mathbf{x}_i; \mathbf{w})]^{\mathsf{y}_i}[1 - p(\mathbf{x}_i; \mathbf{w})]^{1-\mathsf{y}_i} \end{aligned}$$

and we can get our maximum likelihood estimation

Definition 1.4.1 (maximum likelihood estimation)

$$\max_{\mathbf{w}} \prod_{i=1}^{n} [p(\mathbf{x}_i; \mathbf{w})]^{\mathsf{y}_i} [1 - p(\mathbf{x}_i; \mathbf{w})]^{1-\mathsf{y}_i}$$

or equivalently the minimum minus log-likelihood

$$\min_{\mathbf{w}} \sum_{i=1}^{n} [-\mathsf{y}_i \log p(\mathbf{x}_i; \mathbf{w}) - (1 - \mathsf{y}_i) \log(1 - p(\mathbf{x}_i; \mathbf{w}))]$$

Now, how do we define the probability p based on $\mathbf{w}$?

We will assume that the log of the odds ratio $\log \frac{\text{probability of event}}{\text{probability of no event}} = \log \frac{p(\mathbf{x};\mathbf{w})}{1-p(\mathbf{x};\mathbf{w})} = \langle \mathbf{x}, \mathbf{w} \rangle$ is linear.

This leads us to the sigmoid transformation.

Definition 1.4.2 (sigmoid transformation)

$$p(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\langle \mathbf{x}, \mathbf{w} \rangle)}$$

If we return now to the MLE we defined earlier, we get

$$\begin{aligned}
&\min_{\mathbf{w}} \sum_{i=1}^{n} [-\mathsf{y}_i \log p(\mathbf{x}_i; \mathbf{w}) - (1 - \mathsf{y}_i) \log(1 - p(\mathbf{x}_i; \mathbf{w}))] \\
=&\min_{\mathbf{w}} \sum_{i=1}^{n} \left[-\mathsf{y}_i \log \frac{1}{1 + \exp(-\langle \mathbf{x}, \mathbf{w} \rangle)} - (1 - \mathsf{y}_i) \frac{\exp\{-\langle \mathbf{x}, \mathbf{w} \rangle\}}{1 + \exp(-\langle \mathbf{x}, \mathbf{w} \rangle)} \right] \\
=&\min_{\mathbf{w}} \sum_{i=1}^{n} [\mathsf{y}_i \log(1 + \exp(-\langle \mathbf{x}, \mathbf{w} \rangle)) + (1 - \mathsf{y}_i) \log(1 + \exp(-\langle \mathbf{x}, \mathbf{w} \rangle)) + (1 - \mathsf{y}_i) \langle \mathbf{x}, \mathbf{w} \rangle] \\
=&\min_{\mathbf{w}} \sum_{i=1}^{n} \log[1 + \exp(-\langle \mathbf{x}_i, \mathbf{w} \rangle)] + (1 - \mathsf{y}_i)(\langle \mathbf{x}_i, \mathbf{w} \rangle)
\end{aligned}$$

If we redefine $\mathsf{y}'_i = \frac{\mathsf{y}_i + 1}{2}$, i.e., $\mathsf{y}' \in \{\pm 1\}$, then we get the <u>logistic loss</u>

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \log[1 + \exp(-\mathsf{y}'_i \langle \mathbf{x}, \mathbf{w} \rangle)]$$

There is no closed form solution for this problem, so we use the gradient descent algorithm (covered in section 1.8).

Suppose we have found an optimal $\mathbf{w}$. Then, we can set $\hat{y} = 1 \iff p(\mathbf{x}; \mathbf{w}) = \Pr[\mathsf{Y} = 1 \mid \mathsf{X} = \mathbf{x}] > \frac{1}{2}$. The value of $p(\mathbf{x}; \mathbf{w})$ is our confidence.

Remember: All this is under the assumption that the log of the odds ratio is linear. Everything is meaningless if it is not.

Extending to the multiclass case Suppose we instead have $\mathsf{y} \in \{1, \ldots, c\}$ and we need to learn $\mathbf{w}_i$ for each class. The sigmoid function becomes the softmax function

$$\Pr[\mathsf{Y} = k \mid \mathsf{X} = \mathbf{x}; \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_c]] = \frac{\exp\langle \mathbf{x}, \mathbf{w}_k \rangle}{\sum_{l=1}^{c} \exp\langle \mathbf{x}, \mathbf{w}_l \rangle}$$

This maps the real-valued vector $\mathbf{x}$ to a probability vector. Notice that the softmax values for each class are all non-negative and sum to 1.

To train, we use the MLE again

To predict, pick the index of the highest softmax value

$$\hat{\mathsf{y}} = \arg\max_k \Pr[\mathsf{Y} = k \mid \mathsf{X} = \mathbf{x}; \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_c]]$$

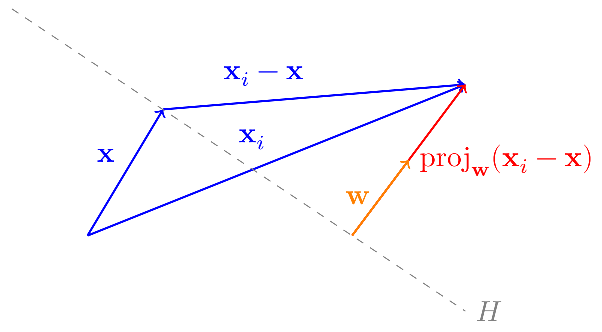
1.5 Hard-Margin Support Vector Machines

Lecture 6 Jan 25

Recall that the perceptron is a feasibility program, i.e., a linear program with $\mathbf{c}^\top \mathbf{x} = \mathbf{0}$. It has infinite solutions.

Naturally, some are much better than others. To take advantage of better algorithms, we can instead maximize the separation.

Let H be a the hyperplane defined by $\langle \mathbf{x}, \mathbf{w} \rangle + b = 0$. The separation (distance) between a point $\mathbf{x}_i$ and H is the length of the projection of $\mathbf{x}_i - \mathbf{x}$ onto the normal vector $\mathbf{w}$.


$\mathbf{x}_i - \mathbf{x}$
$\mathbf{x}$
$\mathbf{x}_i$
$\text{proj}_{\mathbf{w}}(\mathbf{x}_i - \mathbf{x})$
$\mathbf{w}$
H

Simplfiying, we can express this as

$$\begin{aligned}
\frac{\langle \mathbf{x}_i - \mathbf{x}, \mathbf{w} \rangle}{\|\mathbf{w}\|_2} &= \frac{\langle \mathbf{x}_i, \mathbf{w} \rangle - \langle \mathbf{x}, \mathbf{w} \rangle}{\|\mathbf{w}\|_2} && \text{(linearity)} \\
&= \frac{\langle \mathbf{x}_i, \mathbf{w} \rangle + b}{\|\mathbf{w}\|_2} && (\mathbf{x} \in H \Leftrightarrow \langle \mathbf{x}, \mathbf{w} \rangle + b = 0) \\
&= \frac{\mathsf{y}_i \hat{y}_i}{\|\mathbf{w}\|_2}
\end{aligned}$$

We now have something to maximize.

Definition 1.5.1 (margin)

Given a hyperplane $H := \{\mathbf{x} : \langle \mathbf{x}, \mathbf{w} \rangle + b = 0\}$ separating the data, the <u>margin</u> is the smallest distance between a data point $\mathbf{x}_i$ and H.

That is, $\min_i \frac{\mathsf{y}_i \hat{y}_i}{\|\mathbf{w}\|_2}$.

The goal is the maximize the margin across all possible hyperplanes:

$$\max_{\mathbf{w},b} \min_i \frac{\mathsf{y}_i \hat{y}_i}{\|\mathbf{w}\|_2} \quad \text{s.t.} \quad \forall i, \mathsf{y}_i \hat{y}_i > 0 \quad \text{where} \quad \hat{y}_i := \langle \mathbf{x}_i, \mathbf{w} \rangle + b$$

We claim that we can arbitrarily scale the numerator. Let $c > 0$. Then, $(\mathbf{w}, b)$ has the same loss as $(c\mathbf{w}, cb)$ because $\frac{\langle \mathbf{x}_i, c\mathbf{w} \rangle + cb}{\|c\mathbf{w}\|_2} = \frac{c\langle \mathbf{x}_i, \mathbf{w} \rangle + cb}{c\|\mathbf{w}\|_2} = \frac{\langle \mathbf{x}_i, \mathbf{w} \rangle + b}{\|\mathbf{w}\|_2}$.

Therefore, we can equivalently write

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2} \quad \text{s.t.} \quad \min_i \mathsf{y}_i \hat{y}_i = 1 \quad \text{where} \quad \hat{y}_i := \langle \mathbf{x}_i, \mathbf{w} \rangle + b$$

or even better:

$$\min_{\mathbf{w},b} \|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad \forall i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1$$

Finally, consider the points that are closest to the boundary.

Definition 1.5.2

For the separating hyperplane $H = \{\langle \mathbf{x}_i, \mathbf{w} \rangle + b = 0\}$, the two <u>supporting hyperplanes</u> are the parallel hyperplanes $H_+ := \{\langle \mathbf{x}_i, \mathbf{w} \rangle + b = 1\}$ and $H_- := \{\langle \mathbf{x}_i, \mathbf{w} \rangle + b = -1\}$ which represent the margin boundaries.

A <u>support vector</u> is a data point $\mathbf{x}_i \in H_+ \cup H_-$.

The support vectors are rare, but decisive because they reach the boundary of the constraint.

Explanation from the dual perspective Recall the SVM quadratic program

$$\min_{\mathbf{w}_b} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad \forall i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1$$

Introduce Lagrangian multipliers (dual variables) $\boldsymbol{\alpha} \in \mathbb{R}^n$.

$$\begin{aligned}
&\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} > \mathbf{0}} \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_i \alpha_i[\mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1] \\
&= \min_{\mathbf{w},b} \begin{cases} +\infty & \exists i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) < 1 (\text{set } \alpha_i \text{ as } +\infty) \\ \frac{1}{2}\|\mathbf{w}\|_2^2 & \forall i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 (\text{set all } \alpha_i \text{ as } 0) \end{cases} \\
&= \min_{\mathbf{w}_b} \frac{1}{2}\|\mathbf{w}\|_2^2, \quad s.t. \forall i, \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1
\end{aligned}$$

Therefore, we only need to study the minimax problem. Assuming that the problem is convex (which it is, outside the scope of the course), we can express this as

$$\max_{\boldsymbol{\alpha}>\mathbf{0}} \overbrace{\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_i \alpha_i[\mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1]}_{\text{Loss}(\mathbf{w},b,\alpha)}}^{\text{Loss}(\alpha)}$$

and take the derivative of the interior with respect to $\mathbf{w}$ and b:

$$\begin{aligned}\frac{\partial\,\text{Loss}(\mathbf{w},b,\alpha)}{\partial\mathbf{w}} &= \mathbf{w} - \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i = 0\\ \mathbf{w}^* &= \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\\ \frac{\partial\,\text{Loss}(\mathbf{w},b,\alpha)}{\partial b} &= -\sum_i \alpha_i \mathsf{y}_i = 0\\ \sum_i \alpha_i \mathsf{y}_i &= 0\end{aligned}$$

Substitute back into Loss(α):

$$\begin{aligned}\text{Loss}(\alpha) &:= \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_i \alpha[\mathsf{y}_i(\langle \mathbf{x}, \mathbf{w}\rangle + b) - 1]\\ &= \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 - \left\langle \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i, \mathbf{w}\right\rangle - b\sum_i \alpha_i \mathsf{y}_i + \sum_i \alpha_i\\ &= \frac{1}{2}\left\|\sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\right\|_2^2 - \left\langle \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i, \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\right\rangle + \sum_i \alpha_i && (\text{s.t. } \textstyle\sum_i \alpha_i \mathsf{y}_i = 0)\\ &= -\frac{1}{2}\left\|\sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\right\|_2^2 + \sum_i \alpha_i && (\text{s.t. } \textstyle\sum_i \alpha_i \mathsf{y}_i = 0)\end{aligned}$$

Therefore, we can write the dual problem as

$$\min_{\boldsymbol{\alpha}\geq 0} -\sum_i \alpha_i + \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j \mathsf{y}_i\mathsf{y}_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \quad \text{s.t.} \quad \sum_i \alpha_i \mathsf{y}_i = 0$$

We prefer this dual problem because it admits a very easy way to use a non-linear mapping $\mathbf{x} \xrightarrow{\phi} \phi(\mathbf{x})$ to transform non-linearly separable data $\mathbf{x}$ into linearly separable $\phi(\mathbf{x})$. After applying the unknown non-linear mapping, we get

$$\min_{\boldsymbol{\alpha}\geq 0} -\sum_i \alpha_i + \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j \mathsf{y}_i\mathsf{y}_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle \quad \text{s.t.} \quad \sum_i \alpha_i \mathsf{y}_i = 0$$

which we can find *without explicitly applying* ϕ by using the "kernel trick" from section 1.7, writing the inner product directly as a non-linear function.

1.6 Soft-Margin Support Vector Machines

Lecture 7
Jan 30

One of the drawbacks of the hard-margin SVM is that the data must be linearly separable. That is, there must exist a non-zero margin between the data.

If we have a small number of outliers on the wrong side of the decision boundary, we can instead just penalize it in the loss. We do this by relaxing the constraint in hard-margin SVM and including failures in the objective function.

Definition 1.6.1 (hinge loss)
Given label $\mathsf{y} \in \{-1, +1\}$ and score $\hat{y} := \langle \mathbf{x}, \mathbf{w} \rangle + b$, let $\mathsf{y}\hat{y}$ be the confidence.

Define $\ell_{\text{hinge}} = (1 - \mathsf{y}\hat{y})^+ = \begin{cases} 1 - \mathsf{y}\hat{y} & \mathsf{y}\hat{y} < 1 \\ 0 & \text{otherwise} \end{cases}$

In general, notate x^+ to mean $\max\{x, 0\}$.

Now, we can formulate the soft-margin SVM as

$$\boxed{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_i (1 - \mathsf{y}_i\hat{y}_i)^+ \quad \text{s.t.} \quad \hat{y}_i = \langle \mathbf{x}_i, \mathbf{w} \rangle + b} \tag{1.6.a}$$

(margin maximization, regularization hyperparameter, error penalty). Notice that the hard-margin SVM is the limiting behaviour of the soft-margin SVM as $C \to \infty$.

Why do we use the hinge loss? Consider the probability that $\mathsf{Y} \neq \operatorname{sgn}(\hat{\mathsf{Y}})$

$$\Pr\left[\mathsf{Y} \neq \operatorname{sgn}(\hat{\mathsf{Y}})\right] = \Pr\left[\mathsf{Y}\hat{\mathsf{Y}} \leq 0\right] = \mathbb{E}[\mathbb{I}[\mathsf{Y}\hat{\mathsf{Y}} \leq 0]] =: \mathbb{E}[\ell_{0-1}(\mathsf{Y}\hat{\mathsf{Y}})]$$

We want to minimize $\mathbb{E}[\ell_{0-1}(\mathsf{Y}\hat{\mathsf{Y}})]$. Minimizing this value is hard because ℓ_{0-1} is discontinuous at 0 and has gradient $\mathbf{0}$ almost everywhere.

By Bayes' rule, we can rewrite as $\mathbb{E}_{\mathsf{X}}\, \mathbb{E}_{\mathsf{Y}|\mathsf{X}}[\ell_{0-1}(\mathsf{Y}\hat{\mathsf{Y}})]$. Then, we can minimize instead

$$\eta(\mathbf{x}) = \operatorname*{arg\,min}_{\hat{y} \in \mathbb{R}} \mathbb{E}_{\mathsf{Y}|\mathsf{X}=\mathbf{x}}[\ell_{0-1}(\mathsf{Y}\hat{y})]$$

since setting $\mathsf{Y} = \eta(\mathsf{X})$.

Definition 1.6.2 (classification calibrated)
We say a loss function $\ell(\mathsf{y}\hat{y})$ is classification calibrated if for all $\mathbf{x}$,

$$\hat{\mathsf{y}}(\mathbf{x}) := \operatorname*{arg\,min}_{\hat{y} \in \mathbb{R}} \mathbb{E}_{\mathsf{Y}|\mathbf{X}=\mathbf{x}}[\ell(Y\hat{y})]$$

has the same sign as the Bayes rule $\eta(\mathbf{x})$.

Due to Bartlett, we have a helpful theorem

Theorem 1.6.3 (characterization under convexity)
Any convex loss ℓ is classification calibrated if and only if ℓ is differentiable at 0 and $\ell'(0) < 0$.

Corollary 1.6.4. A classifier that minimizes the expected hinge loss also minimizes the expected 0-1 loss.

This theorem is also one of the big reasons why the perceptron cannot generalize well.

Remark 1.6.5. The perceptron loss $\ell(\mathsf{y}\hat{y}) = -\min\{\mathsf{y}\hat{y}, 0\}$ is not differentiable at 0, so it is not classification calibrated and cannot generalize.

Generating the dual Recall the soft-margin SVM

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_i (1 - \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b))^+$$

Notice that we can write $C \cdot (t)^+ = \max\{Ct, 0\} = \max_{0\le\alpha\le C} \alpha t$ to get

$$\min_{\mathbf{w},b} \max_{0\le\boldsymbol{\alpha}\le C} \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_i \alpha_i(1 - \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b))$$

As before, swap min with max:

$$\max_{0\le\boldsymbol{\alpha}\le C} \overbrace{\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_i \alpha_i(1 - \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b))}_{\text{Loss}(\mathbf{w},b,\alpha)}}^{\text{Loss}(\alpha)}$$

Now, set our optimality conditions

$$\frac{\partial \text{Loss}(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i = \mathbf{0} \qquad \frac{\partial \text{Loss}(\mathbf{w},b,\alpha)}{\partial b} = -\sum_i \alpha_i \mathsf{y}_i = 0$$

$$\mathbf{w} = \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i \qquad \sum_i \alpha_i \mathsf{y}_i = 0$$

and substitute into Loss(α):

$$\begin{aligned}
\text{Loss}(\alpha) &:= \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_i \alpha_i(1 - \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b)) \\
&= \frac{1}{2}\left\|\sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\right\|_2^2 + \sum_i \alpha_i - \left\langle \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i, \sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i \right\rangle \\
&= -\frac{1}{2}\left\|\sum_i \alpha_i \mathsf{y}_i \mathbf{x}_i\right\|_2^2 + \sum_i \alpha_i
\end{aligned}$$

Switching from max to min and expanding the norm, we get

$$\boxed{\min_{0\leq\boldsymbol{\alpha}\leq C} -\sum_i \alpha_i + \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \quad \text{s.t.} \quad \sum_i \alpha_i \mathsf{y}_i = 0} \tag{1.6.b}$$

which is identical to the hard-margin SVM dual with an upper bound C on $\boldsymbol{\alpha}$.

Lecture 8
Feb 1

Suppose we solve the dual (eq. 1.6.b) with optimal solution $\boldsymbol{\alpha}^*$. Then,

$$\mathbf{w}^* = \sum_i \alpha_i^* \mathsf{y}_i \mathbf{x}_i. \tag{1.6.c}$$

If we have a point on $H_{\pm 1}$, i.e., $\mathsf{y}\hat{y} = 1$, we can recover b^* as $\mathsf{y} - \langle \mathbf{x}, \mathbf{w}^* \rangle$.

Training by gradient descent Suppose we have a minimization problem $\min_{\mathbf{x}} f(\mathbf{x})$. Then, to make a guess $\mathbf{x}$ better, set $\mathbf{x} \leftarrow \mathbf{x} - \eta \cdot \nabla_{\mathbf{x}} f(\mathbf{x})$ for some <u>learning rate</u> $\eta > 0$.

Given the problem

$$\min_{\mathbf{w},b} \frac{1}{2\lambda}\|\mathbf{w}\|_2^2 + C\sum_i \ell(\mathsf{y}_i\hat{y}_i) \quad \text{where} \quad \hat{y}_i = \langle \mathbf{x}_i, \mathbf{w}, \mathbf{x}_i, \mathbf{w} \rangle + b$$

with loss function ℓ, the gradient descent steps are

$$\begin{aligned}
\mathbf{w} &\leftarrow \mathbf{w} - \eta \cdot \nabla_{\mathbf{w}}\left(\frac{1}{2\lambda}\|\mathbf{w}\|_2^2 + C\sum_i \ell(\mathsf{y}_i\hat{y}_i)\right) \\
&= \mathbf{w} - \eta\left[\frac{\mathbf{w}}{\lambda} + C\sum_i \ell'(\mathsf{y}_i\hat{y}_i)\mathsf{y}_i\mathbf{x}_i\right] \\
b &\leftarrow b - \eta \cdot \nabla_b\left(\frac{1}{2\lambda}\|\mathbf{w}\|_2^2 + C\sum_i \ell(\mathsf{y}_i\hat{y}_i)\right) \\
&= b - \eta\left[C\sum_i \ell'(\mathsf{y}_i\hat{y}_i)\mathsf{y}_i\right]
\end{aligned}$$

because $\nabla_{\mathbf{w}}\ell(\mathsf{y}_i\hat{y}_i) = \ell'(\mathsf{y}_i\hat{y}_i)\cdot\mathsf{y}_i\nabla_{\mathbf{w}}(\hat{y}_i) = \ell'(\mathsf{y}_i\hat{y}_i)\mathsf{y}_i\mathbf{x}_i$ and $\nabla_b\ell(\mathsf{y}_i\hat{y}_i) = \ell'(\mathsf{y}_i\hat{y}_i)\cdot\mathsf{y}_i\nabla_b(\hat{y}_i) = \ell'(\mathsf{y}_i\hat{y}_i)\cdot\mathsf{y}_i$.

If ℓ is hinge loss, we define the derivative $\ell'(t) = \begin{cases} -1 & t \leq 1 \\ 0 & t > 1 \end{cases}$.

If ℓ is perceptron loss, we define $\ell'(t) = \begin{cases} -1 & t \leq 0 \\ 0 & t > 1 \end{cases}$.

All other common loss functions are easily differentiable.

1.7 Reproducing Kernels

We have dealt with data that is perfectly linearly separable (hard-margin SVM) and mostly linearly separable (soft-margin SVM).

Problem 1.7.1

How can we use our existing techniques to classify a fully non-linearly separable dataset?

In the linear classifier, we used an affine function $\langle \mathbf{w}, \mathbf{x} \rangle + b$. Now, we define a quadratic classifier.

Definition 1.7.2 (quadratic classifier)

A function $f : \mathbb{R}^d \to \mathbb{R}^d$ of the form $f(\mathbf{x}) = \langle \mathbf{x}, Q\mathbf{x} \rangle + \sqrt{2} \langle \mathbf{x}, \mathbf{p} \rangle + b$ where the weights to be learned are $Q \in \mathbb{R}^{d \times d}$, $\mathbf{p} \in \mathbb{R}^d$, and $b \in \mathbb{R}$.

Recall from linear algebra that for all A, B, C, $\langle AB, C \rangle = \langle B, A^\top C \rangle$ and $\langle A, BC \rangle = \langle AB^\top, C \rangle$.

Definition 1.7.3 (matrix vectorization)

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, let $\overrightarrow{\mathbf{A}} \in \mathbb{R}^{mn}$ be its vectorization. That is,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \implies \overrightarrow{\mathbf{A}} = \begin{bmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

Then, we can write the quadratic classifier as:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{x}, Q\mathbf{x} \rangle + \sqrt{2} \langle \mathbf{x}, \mathbf{p} \rangle + b \\ &= \langle \mathbf{x}\mathbf{x}^\top, Q \rangle + \left\langle \sqrt{2}\mathbf{x}, \mathbf{p} \right\rangle + b \\ &= \left\langle \begin{bmatrix} \overrightarrow{\mathbf{x}\mathbf{x}^\top} \\ \sqrt{2}\mathbf{x} \\ 1 \end{bmatrix}, \begin{bmatrix} \overrightarrow{Q} \\ \mathbf{p} \\ b \end{bmatrix} \right\rangle \end{aligned}$$

If we write $\phi(\mathbf{x}) = (\overrightarrow{\mathbf{x}\mathbf{x}^\top}, \sqrt{2}\mathbf{x}, 1)^\top$ and $\mathbf{w} = (\overrightarrow{Q}, \mathbf{p}, b)^\top$, then we can write f as

$$f(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w} \rangle$$

but this really blows up the dimension to $\mathbb{R}^{d^2+d+1}$. Recall that in the dual forms of SVM, all we need is to know the inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle$. With our new ϕ, we get

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) := \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \left\langle \begin{bmatrix} \overrightarrow{\mathbf{x}\mathbf{x}^\top} \\ \sqrt{2}\mathbf{x} \\ 1 \end{bmatrix}, \begin{bmatrix} \overrightarrow{\mathbf{z}\mathbf{z}^\top} \\ \sqrt{2}\mathbf{z} \\ 1 \end{bmatrix} \right\rangle \\ &= \left\langle \overrightarrow{\mathbf{x}\mathbf{x}^\top}, \overrightarrow{\mathbf{z}\mathbf{z}^\top} \right\rangle + \left\langle \sqrt{2}\mathbf{x}, \sqrt{2}\mathbf{z} \right\rangle + 1 \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2 + 2 \langle \mathbf{x}, \mathbf{z} \rangle + 1 \\ &= (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^2 \end{aligned}$$

This process is easily reproducable for a given ϕ. What about the other direction?

Definition 1.7.4 (reproducing kernel)

We call $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a reproducing kernel if there exists some $\phi : \mathcal{X} \to \mathcal{H}$ so that $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = k(\mathbf{x}, \mathbf{z})$.

Remark 1.7.5. When such a kernel exists, it may not be unique.

For example, the kernels $\phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2] \in \mathbb{R}^3$ and $\psi(\mathbf{x}) = [x_1^2, x_1x_2, x_1x_2, x_2^2] \in \mathbb{R}^4$ have the same inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = \langle \psi(\mathbf{x}), \psi(\mathbf{z}) \rangle$.

Theorem 1.7.6 (Mercer's theorem)

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if for any $n \in \mathbb{N}$ and any $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, the kernel matrix $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ is symmetric and positive semi-definite.

Recall from linear algebra: K is symmetric if $K_{ij} = K_{ji}$ for all indices, and positive semi-definite if $\langle \boldsymbol{\alpha}, K\boldsymbol{\alpha} \rangle \geq 0$ for all vectors $\boldsymbol{\alpha}$.

The proof is extremely convoluted and well beyond the scope of the course.

Example 1.7.7. The following are kernels:

- the polynomial kernel $k(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^p$ for hyperparameter p,
- the Gaussian kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\|\mathbf{x} - \mathbf{z}\|_2^2 / \sigma\right)$ for hyperparameter σ, and
- the Laplace kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\|\mathbf{x} - \mathbf{z}\|_2 / \sigma\right)$ for hyperparameter σ

Now, we can substitute our expression for the inner product to eqs. 1.6.a and 1.6.b, the primal and dual of the soft-margin SVM:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_i (1 - \mathsf{y}_i \hat{y}_i)^+ \quad \text{s.t.} \quad \hat{y}_i = \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle$$

$$\min_{0 \leq \boldsymbol{\alpha} \leq C} -\sum_i \alpha_i + \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{s.t.} \quad \sum_i \alpha_i \mathsf{y}_i = 0$$

Once we solve $\boldsymbol{\alpha}^*$, we can try to recover $\mathbf{w}^*$ as in eq. 1.6.c

$$\mathbf{w}^* = \sum \alpha_i^* \mathsf{y}_i \phi(\mathbf{x}_i)$$

but this will not work since we do not know ϕ explicitly. Instead, we only need to compute the score function

$$\begin{aligned} f(\mathbf{x}) &:= \langle \phi(\mathbf{x}), \mathbf{w}^* \rangle \\ &= \left\langle \phi(\mathbf{x}), \sum \alpha_i^* \mathsf{y}_i \phi(\mathbf{x}_i) \right\rangle \\ &= \sum \alpha_i^* \mathsf{y}_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle \\ &= \sum \alpha_i^* \mathsf{y}_i k(\mathbf{x}, \mathbf{x}_i) \end{aligned}$$

and return $\operatorname{sgn}(f(\mathbf{x}))$.

1.8 Gradient Descent