Week 5: Support Vector Machines and Tensor Backpropagation

https://mlvu.github.io

February 28, 2020

1 Support Vector Machines

1.1 Tensor loss

The basic optimization objective for Support Vector Machines is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\| + C\sum_i p_i \\ \text{such that} \quad & y_i(w^T x_i + b) \geqslant 1 - p_i \\ \text{and} \quad & p_i \geqslant 0 \end{aligned}$$

question 1: What does i index? How many terms does the sum have, and how many constraints are there?

It iterates over the data. There are as many terms as there are instances in the data. There are two constraints per instance.

What is the value of y_i in this expression? What is its function?

y_i is 1 for positive examples, and -1 for negative examples. It allows us to rewrite the constraints

$$\begin{aligned} w^T x_i + b \geqslant 1 \text{ if } x_i \text{ is positive} \\ w^T x_i + b \leqslant -1 \text{ if } x_i \text{ is negative} \end{aligned}$$

to a single constraint.

question 2: There are two common ways to rewrite this expression before implementing it. What are they (in general terms) and what are their benefits?

The first option is to rewrite everything in terms of $\boldsymbol{w}$ and b in order to get rid of the constraints. This is useful when we want to use the SVM as the last layer in a neural network. Without constraints, we are free to use basic backpropagation.

The second option is to use Lagrange multipliers to get rid of $\boldsymbol{w}$ and rewrite everything in terms of the multipliers. This expresses the solution purely in terms of the support vectors.

The main benefit is that the whole algorithm can be written in terms of the dot products of pairs of instances. This means we do not need to see the actual feature vectors to compute the support vectors, only the dot products. This allows us to apply the *kernel trick*.

1.2 Lagrange multipliers

Lagrange multipliers are a useful trick to know. In the lectures we only had time to describe the trick itself, and the rules for applying it. That should be enough to do the exam questions and the questions below, but if you want some more intuition for why Lagrange multipliers work the way they do please read this article from the required reading.

We have the following optimization problem:

$$\begin{aligned} \text{minize } f(a, b) &= a^2 + 2b \\ \text{such that } a^2 &= -b^2 + 1 \end{aligned}$$

question 3: The first step is to rewrite the constraint so that the right side is equal to zero. Do so. $a^2 + b^2 - 1 = 0$ What does the constraint say about the allowed inputs (what shape do the allowed inputs make in the (a, b)-plane)? The solutions are constrained to a circle, centered on the origin, with radius 1 (the so-called bi-unit circle).

We now define a function $L(a, b, \alpha) = f(a, b) + \alpha G$, where G is the left hand side of the constraint equal to zero (how much any given a and b violate the constraint).[1]

question 4: Write out $L(a, b, \alpha)$ for our problem. $L(a, b, \alpha) = a^2 + 2b^2 + \alpha(a^2 + b^2 - 1)$

We take the derivative of L with respect to each of its *three* parameters, and set these equal to zero.

[1]For plain Lagrange multipliers, where the constraints are all equalities, we can either add or subtract the term containing the constraint. For inequality constrains, it depends on whether we are maximizing or minimizing.

question 5: Fill in the blanks

$$\begin{aligned}
\frac{\partial L}{\partial a} &= \frac{\partial(a^2 + 2b^2 + \alpha a^2 + \alpha b^2 - \alpha)}{\partial a} \\
&= 2a + 2\alpha a \\
\frac{\partial L}{\partial b} &= 4b + 2\alpha b \\
\frac{\partial L}{\partial \alpha} &= a^2 + b^2 - 1
\end{aligned}$$

$$a(2 + 2\alpha) = 0 \tag{1}$$

$$b(4 + 2\alpha) = 0 \tag{2}$$

$$a^2 + b^2 = 1 \tag{3}$$

Note that the last line recovers the original constraint. We now have three equations with three unknowns, so we can solve for a and b. From the shape of the function (it's symmetric in both the a and b axes), we should expect at least two solutions.

We can get these from the above equations by noting that if a and b are both nonzero, we can derive a contradiction. Thus either a or b must be zero.

question 6: Give the solutions for both cases (remember that $x^2 = 1$ has *two* solutions).

From line (3), above we see that if $a = 0$, then $b^2 = 1$ and vice versa. This gives us

$$\begin{aligned}
a &= 0, b = 1 \\
a &= 0, b = -1 \\
a &= 1, b = 0 \\
a &= -1, b = 0
\end{aligned}$$

as extrema. Filling these in, we find that the last two lines minimize the function. (The first two are maxima, as can be seen by clicking the Wolfram Alpha link below.

Happily, Wolfram Alpha agrees with us (and provides some informative plots).

1.3 The kernel trick

The feature space of k is a projection of point $\mathbf{a}$ to point $\mathbf{a}'$ such that

$$k(\mathbf{a}, \mathbf{b}) = \mathbf{a}'^{\mathsf{T}} \mathbf{b}' \,.$$

We have a dataset with two features Let $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$. We define the *kernel*

$$k_1(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^{\mathsf{T}} \mathbf{b})^2 \,.$$

question 7: Show that the feature space defined by$k_!$ is

$$\begin{pmatrix} {x_1}^2 \\ \sqrt{2} x_1 x_2 \\ {x_2}^2 \end{pmatrix} \,.$$

Hint: start by writing out the definition as a scalar function. See if you can re-arrange this back into a dot procut of two other vectors.

Starting from the definition of k:

$$\begin{aligned}
k(\mathbf{a}, \mathbf{b}) &= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 \\
&= (a_1 b_1 + a_2 b_2)^2 \\
&= a_1 b_1 a_1 b_1 + 2 a_2 b_2 a_1 b_1 + a_2 b_2 a_2 b_2 \\
&= a_1 a_1 \cdot b_1 b_1 + 2 \cdot a_1 a_2 \cdot b_1 b_2 + a_2 a_2 \cdot b_2 b_2 \\
&= \begin{pmatrix} a_1 a_1 \\ \sqrt{2} a_1 a_2 \\ a_2 a_2 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} b_1 b_1 \\ \sqrt{2} b_1 b_2 \\ b_2 b_2 \end{pmatrix}
\end{aligned}$$

question 8: What is the feature space for the kernel

$$k_2(\mathbf{a}, \mathbf{b}) = \left(\mathbf{a}^{\mathsf{T}} \mathbf{b} + 1 \right)^2 \quad ?$$

$$
\begin{aligned}
k(\mathbf{a}, \mathbf{b}) &= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \begin{pmatrix} b_1 \\ b2 \end{pmatrix} + 1 \right)^2 \\
&= (a_1 b_1 + a_2 b_2)^2 + 2(a_1 b_1 + a_2 b_2) + 1 \\
&= a_1 b_1 a_1 b_1 + 2 a_1 b_1 a_2 b_2 + a_2 b_2 a_2 b_2 + 2 a_1 b_1 + 2 a_2 b_2 + 1 \\
&= \begin{pmatrix} 1 \\ \sqrt{2} a_1 \\ \sqrt{2} a_2 \\ {a_1}^2 \\ \sqrt{2} a_1 a_2 \\ {a_2}^2 \end{pmatrix}^T \begin{pmatrix} 1 \\ \sqrt{2} b_1 \\ \sqrt{2} b_2 \\ {b_1}^2 \\ \sqrt{2} b_1 b_2 \\ {b_2}^2 \end{pmatrix}
\end{aligned}
$$

2 Backpropagation Revisited

2.1 The multivariate chain rule

If we want to do backpropagation for a computation graph where an output variable depend in an input variable through multiple intermediate values (the graph contains a diamond), we require the *multivariate chain rule*. If you don't quite remember how it goes, re-read slides 24–29 fo the deep learning lecture before trying these questions.

We will use the backpropagation algorithm to find the derivative of the function

$$f(x) = \sin(x^2)\cos(x^3)$$

with respect to x.

question 9: First, we break the function up into modules. Fill in the blanks.

$$
\begin{aligned}
f &= ab \\
a &= \sin(c) \\
b &= \cos(d) \\
c &= x^2 \\
d &= x^3
\end{aligned}
$$

question 10: Draw the computation graph with nodes x, c, s, a, b, f

$$\begin{tikzcd}
 & c \arrow[r] & a \arrow[dr] & \\
x \arrow[ur] \arrow[dr] & & & f \\
 & d \arrow[r] & b \arrow[ur] &
\end{tikzcd}$$

question 11: Work out the *local* derivatives. Which is the correct expression for the gradient $\partial f/\partial x$?

$$\begin{aligned}
\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial x} && \text{multivariate chain rule}\\
&= \frac{\partial f}{\partial a}\frac{\partial a}{\partial c}\frac{\partial c}{\partial x} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial x} && \text{chain rule for } c\text{: } \frac{a}{x} = \frac{\partial a}{\partial c}\frac{\partial c}{\partial x}\\
&= \frac{\partial f}{\partial a}\frac{\partial a}{\partial c}\frac{\partial c}{\partial x} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial d}\frac{d}{x} && \text{chain rule for } d\\
&= b \cdot \cos(c) \cdot 2x - a \cdot \sin(d) \cdot 3x^2 && \text{work out all local derivatives}
\end{aligned}$$

This is a common exam question so make sure to practice if you're not sure. You can easily create new questions for yourself by coming up with any function $f(x)$ with a diamond shape in the computation (just make sure that x is used twice).

2.2 Tensor Backpropagation

In scalar backpropagtion, we need to work out only the local derivatives. Once we have these, we can multiply them in any order to get the global derivative. If we want to apply backpropagation to tensors, things are not so easy.

question 12: Why not? What is it about the local derivatives in tensor backpropagation that makes it difficult to apply in this way?

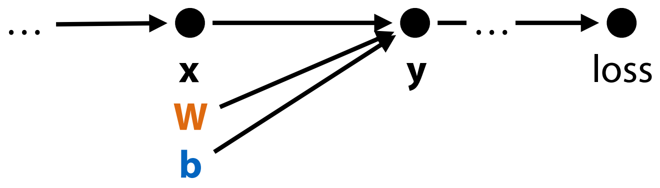
In tensor backpropagation, the local derivatives are often, for instance, the derivative of a vector function over matrix parameters, like the local derivative $\partial \mathbf{k}/\partial \mathbf{W}$ shown in the slides. In this case, the collection of all scalar derivatives (every output with every input) is best represented as a 3 tensor. This would take way too much memory, and there's then no natural way to multiply this local derivative with the others to compute the global derivative.

The solution is not to compute the local derivatives explicitly, but only to compute the gradients on the inputs over the loss, given the gradients of the outputs over the loss.

We will practice this for a simple feedforward layer (without activation). Consider a module f that computes:[2]

$$\mathbf{y} = f(\mathbf{x}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Assume that this module is part of a much larger network. Its output $\mathbf{y}$ is fed as input to another module, which produces a new output that is fed to another module and so on. At the end, a single scalar loss L is produced.



Ultimately, we want to work out the derivative of this loss, with respect to our inputs:

$$\frac{\partial L}{\partial \mathbf{x}} \quad \frac{\partial L}{\partial \mathbf{W}} \quad \frac{\partial L}{\partial \mathbf{b}}$$

We'll start with the bias term $\mathbf{b}$. Since matrix/vector calculus can get complicated, and doesn't translate to tensors of higher rank, we'll develop everything in terms of scalar calculus. Once we've taken the derivatives we'll transform everything to matrix operations to make the implementation efficient.

We're interested in $\frac{\partial L}{\mathbf{y}} \frac{\partial \mathbf{y}}{\partial b_j}$, but we need matrix/vector calculus to do that. Instead we will consider $L(f(\mathbf{b}))$ as a *scalar* computation, which has multiple intermediate values $y_1, y_2 \ldots, y_k$. By the multivariate chain rule, we can just take the derivative over each path (through each value y_i), and sum them:

$$\frac{\partial L}{\partial b_j} = \sum_i \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial b_j}$$

We will assume that $\partial L/\partial y_i$ is given to us by the AD engine as a vector $\mathbf{d}$ with $d_i = \partial L/\partial y_i$. All we need to work out is a simple matrix operation that computes $\partial L/\partial b_j$ for all j and lays the results out in the same shape as $\mathbf{b}$. We'll start by working out the scalar derivative.

[2]Normally, we would consider $\mathbf{x}$ the input, and $\mathbf{W}$ and $\mathbf{b}$ the parameters, but for our AD engine, the distinction does not matter: everything going in to the computation is an input, and we may need the gradient over all three.

question 13: Fill in the gaps

$$
\begin{aligned}
\frac{\partial L}{\partial b_j} &= \sum_i \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial b_j} = \sum_i d_i \frac{\partial y_i}{\partial b_j} \\
&= \sum_i d_i \frac{\partial [\mathbf{W}\mathbf{x} + \mathbf{b}]_i}{\partial b_j} = \sum_i d_i \frac{\partial [\mathbf{W}\mathbf{x}]_i + b_i}{\partial b_j} \\
&= \sum_i d_i \frac{\partial b_i}{\partial b_j} = d_j
\end{aligned}
$$

question 14: What should the `backward()` function for f return as the gradient of $\mathbf{b}$?

We should return a vector $\mathbf{b}'$ with the same shape as $\mathbf{b}$, where $b'_j = \partial L/\partial b_j$. Since d_j is the gradient for b_j, we can just return the vector $\mathbf{d}$.

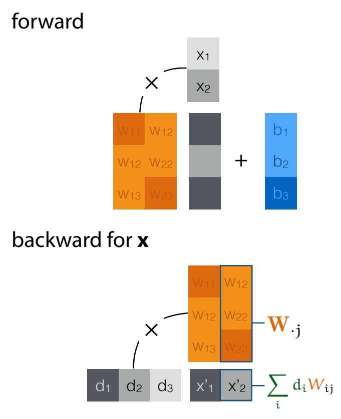
We'll do the same for the gradient over $\mathbf{x}$.

question 15: Work out the scalar derivative $\partial L/\partial x_j$ using the multivariate chain rule (again taking $\partial L/\partial y_i$ as given).

$$
\begin{aligned}
\frac{\partial L}{\partial x_j} &= \sum_i \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial x_j} = \sum_i d_i \frac{\partial y_i}{\partial x_j} \\
&= \sum_i d_i \frac{\partial \mathbf{W}\mathbf{x}}{\partial x_j} = \sum_i d_i \frac{\partial \mathbf{W}_{i\cdot} \times \mathbf{x} + b}{\partial x_j} \\
&= \sum_i d_i \frac{\partial \sum_k \mathbf{W}_{ik} x_k}{\partial x_j} = \sum_i d_i \frac{\partial \mathbf{W}_{ij} x_j}{\partial x_j} \\
&= \sum_i d_i \mathbf{W}_{ij}
\end{aligned}
$$

question 16: What should the `backward()` function for f return as the gradient of $\mathbf{x}$?

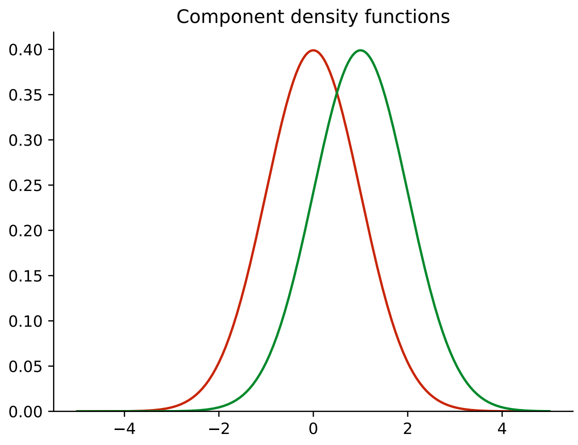
We are looking for a function that returns a vector $\mathbf{x}'$ where $x'_j = \sum_i d_i \mathbf{W}_{ij}$. In other words, x'_j should be the dot product between $\mathbf{d}$ and and the j-th column of $\mathbf{W}$. We get this if we compute $\mathbf{x}' = \mathbf{d}^T\mathbf{W}$.

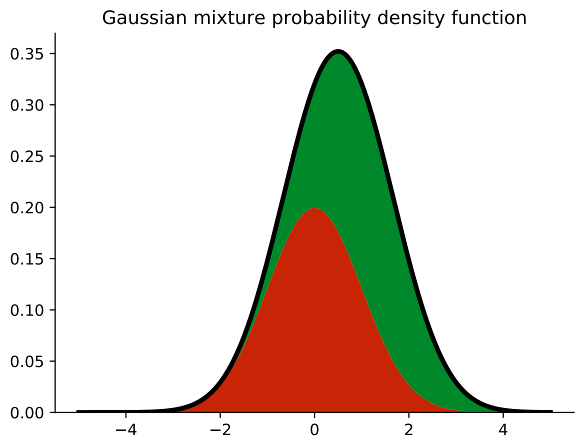


3 Bonus: Expectation Maximization

This is not an exam question, but it's helpful to do if you want to understand the EM algorithm.

Assume we have a Gaussian Mixture Model in one dimension with two components: $N(0, 1)$ and $N(1, 1)$. The weights w_1 and w_2 of the components are equal.





question 17: Compute the probability density of the point 0, under the Gaussian Mixture.

$$\begin{aligned}
p(0) &= \frac{1}{2}N(0,1) + \frac{1}{2}N(1,1) \\
&= \frac{1}{2\sqrt{2\pi}}\exp(0) + \frac{1}{2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\right) \\
&= \frac{1}{2}\frac{1}{\sqrt{2\pi}} + \frac{1}{2}\frac{1}{\sqrt{2\pi e}} \approx 0.32
\end{aligned}$$

question 18: Under the EM algorithm, what responsibility is assigned to each component for the point 0?

To compute the responsibility, we compute the probability of the component z given the point x: $p(z \mid x)$, and normalize over all components. We get for component 1:

$$\frac{\frac{1}{2\sqrt{2\pi}}\exp(0)}{\frac{1}{2\sqrt{2\pi}}\exp(0) + \frac{1}{2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\right)} = \frac{1}{1+\frac{1}{\sqrt{e}}} \approx 0.62$$

and for component 2:

$$\frac{\frac{1}{2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\right)}{\frac{1}{2\sqrt{2\pi}}\exp(0) + \frac{1}{2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\right)} = \frac{\frac{1}{\sqrt{e}}}{1+\frac{1}{\sqrt{e}}} \approx 0.37$$

Note: Using Bayes' rule, this translates to $p(z \mid x) = \frac{p(x|z)p(z)}{p(x)}$. The denominator is the sum computed in the previous exercise, and the responsibilities are the proportions of each term to the total.[3]

This is no accident, it is essentially what Bayes' rule tells us: to compute $p(z \mid x)$, we find $p(x)$ by marginalizing Z out of $p(X = x, Z)$. This gives us a big sum, with one term for each z. The proportion of this term to the total is $p(z \mid x)$.