Linear classifiers (again)

- Let \( f(x) \) be a linear function of its argument vector \( x = (x_1, \ldots, x_m) \)
  \[
  f(x) = \langle \mathbf{w} \cdot x \rangle + b = \mathbf{w}^\top x + b = \sum_{i=1}^{m} w_i x_i + b
  \]

- Input \( x \) is classified positive, if \( f(x) \geq 0 \) and otherwise \( x \) is classified negative
  \[
  \text{sign}(f(x)) = \begin{cases} 
  1, & f(x) \geq 0 \\
  -1, & \text{otherwise}
  \end{cases}
  \]

- The hyperplane determined by the equation \( \langle \mathbf{w} \cdot x \rangle + b = 0 \) divides the input space in two halfspaces
- If for an example \((x, y)\) and hypothesis \( h \) it holds that \( y h(x) > 0 \), then the example has been correctly classified
- \( y = y h(x) \) is the margin of the example w.r.t. \( h \)

18.9 Support Vector Machines

- Vladimir Vapnik & colleagues, 1990's
- The SVM framework is currently the most popular approach for "off-the-shelf" supervised learning
- Three advantages of SVMs:
  1. SVMs construct a maximum margin separator – a decision boundary with the largest possible distance to example points. This helps them to generalize well
  2. SVMs create a linear separatable hyperplane, but they have the ability to embed the data into a higher-dimensional space using the so-called kernel trick. Often data that are not linearly separable in the original input space are easily separable in the higher-dimensional space. The high-dimensional linear separator is actually nonlinear in the original space. This means the hypothesis space is greatly expanded over methods that use strictly linear representations
3. SVMs are a nonparametric method – they retain training examples and potentially need to store them all. In practice they often end up retaining only a small fraction of the number of examples – sometimes as few as a small constant times the number of dimensions. SVMs combine the advantages of nonparametric and parametric models: they have flexibility to represent complex functions, but they are resistant to overfitting

- Consider a linearly separable binary classification problem. There is an infinite number of decision boundaries consistent with all the data
- From the point of view of 0/1 loss each would be equally good
- Logistic regression would find some separating line, the exact location of which depends on all the example points
- The key insight of SVMs is that some examples are more important than other, and that paying attention to them can lead to better generalization

Instead of minimizing expected empirical loss on the training data, SVMs attempt to minimize expected generalization loss

- Under the probabilistic assumption that the as-yet-unseen points are drawn from the same distribution as the previously seen examples, there are arguments suggesting that we minimize generalization loss by choosing the separator that is farthest away from the examples we have seen so far
- This is called the maximum margin separator
- The margin is twice the distance from the separator to the nearest example point
- Traditionally SVMs use the convention that class labels are +1 and -1
- The separator is defined as the set of points \( \{ x \mid w \cdot x + b = 0 \} \)
- Gradient descent in the space of \( w \) and \( b \) could be used to find the parameters that maximize the margin while correctly classifying all the examples
• There is another approach to solving the problem
• There is an alternative representation called the dual representation, in which the optimal solution is found by solving
\[ \arg \max_{\alpha} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k) \]
subject to constraints \( \alpha_j \geq 0 \) and \( \sum_j \alpha_j y_j = 0 \)
• This is a quadratic programming optimization problem, for which there are good software packages
• Once we have found the vector \( \alpha \) we can get back to \( w \) with the equation \( w = \sum_j \alpha_j x_j \), or we can stay in the dual representation
• The first important property of the above dual optimization problem is that the expression is convex: it has a single global maximum that can be found efficiently

• Second, the data enter the expression only in the form of dot products of pairs of points
• This is also true of the equation for the separator itself; once the optimal \( \alpha_j \) have been calculated, it is
\[ h(x) = \text{sign} \left( \sum_j \alpha_j y_j (x \cdot x_j) - b \right) \]
• The third important property is that the weights \( \alpha_j \) associated with each data point are zero except for support vectors – the points closest to the separator
• Because there are usually many fewer support vectors than examples, SVMs gain some of the advantages of parametric models
• However, we would not usually expect to find a linear separator in the input space \( x \), but we can find linear separators in a high-dimensional feature space \( F(x) \) simply by replacing \( x \cdot x \) in the dual representation optimization problem with \( F(x_j) \cdot F(x_k) \).
• Kernel function \( K \) allows us to find linear separators in the high-dimensional feature space \( F(x) \) simply by replacing \( x_j \cdot x_k \) with a kernel function \( K(x_j, x_k) \).
• Thus we can learn in the higher-dimensional space, but we compute only kernel functions rather than the full list of features for each data point.
• As an example, \( F(x_j) \cdot F(x_k) = (x_j \cdot x_k)^2 \) when we use three features:
  \[
  f_1 = x_j^2, \quad f_2 = x_k^2, \quad f_3 = \sqrt{2} \cdot x_j \cdot x_k
  \]

  Kernel trick: Plugging the kernels into the optimization problem optimal linear separators can be found efficiently in feature spaces with billions of (or, in some cases, infinitely many) dimensions.

18.10 Ensemble Learning

• Select a whole collection, or ensemble, of hypotheses and combine their prediction
• For example, we might generate a hundred different decision trees from the same training set and have them vote on the best classification for a new instance.
• If an ensemble contains 5 hypotheses and we combine their predictions using simple majority voting, then misclassifying an instance requires that at least three of the five hypotheses have to misclassify it.
• In general, we hope that the misclassification of three hypotheses is less likely than a misclassification by a single hypothesis.
• The hypotheses in the collection are not independent of each other, but if the hypotheses are at least a little bit different that reduces the correlation between their errors.

• Using a hypothesis collection increases the expressiveness of the hypothesis class without incurring much additional computational or algorithmic complexity.

• In a weighted training set each example has an associated weight \( w_j \geq 0 \), which expresses its importance.

• *Boosting* [Schapire 1990, Freund & Schapire 1996] is the most widely used ensemble method.

• Generate the first hypothesis \( h_1 \) from the normal training set in which all the examples have weight \( w_j = 1/n \).

• Hypothesis \( h_1 \) will classify some of the training examples correctly and some incorrectly.

• We would like the next hypothesis to do better on the misclassified examples.

• We increase the weights of misclassified examples while decreasing the weights of correctly classified examples.

• From this new weighted training set, we generate hypothesis \( h_2 \).

• The process continues in this way until we have generated \( K \) hypotheses, where \( K \) is an input to the boosting algorithm.

• The final ensemble hypothesis is a weighted-majority combination of all the \( K \) hypothesis, each weighted according to how well it performed on the training set.
Algorithm AdaBoost($S$, $L$, $K$)

**Input:** $S$, training set $(x_1, y_1), \ldots, (x_n, y_n)$; $L$, a learning algorithm; $K$, the number of hypotheses in the example

$w \leftarrow (1/n, \ldots, 1/n)$ ;

for $k = 1$ to $K$

$h_k \leftarrow L(S, w)$;

error $\leftarrow 0$

for $j = 1$ to $n$

if $h_k(x_j) \neq y_j$ then error $\leftarrow$ error + $w_j$;

for $j = 1$ to $n$

if $h_k(x_j) = y_j$ then $w_j \leftarrow w_j \cdot$error/(1–error);

$w \leftarrow$ Normalize($w$);

$z_k \leftarrow \log (1–$error)/error; % $z_k$ is the weight of hypothesis $k$

return Weighted-Majority($h, z$);

- The developers of AdaBoost were awarded the Gödel-prize in 2003
- A weak learning algorithm always returns a hypothesis with accuracy on the training set that is always slightly better that random guessing (i.e., 50%+ for Boolean classification)
- AdaBoost provably boosts the accuracy of the original weak learner on the training data to classify the data perfectly (for large enough $K$)
- An often used hypothesis space is decision trees with just one test, at the root a.k.a. decision stumps
- Increasing the ensemble size $K$ reduces both training error and test error (at least to begin with)
- The test set performance may still continue to increase long after the training set error has reached zero
20 LEARNING PROBABILISTIC MODELS

- Surprise candy comes in two flavors: cherry and lime
- Each piece of candy is wrapped in the same opaque wrapper, regardless of flavor
- Five kinds of bags of candy are indistinguishable from the outside
  \[ h_1: 100\% \text{ cherry} \]
  \[ h_2: 75\% \text{ cherry} + 25\% \text{ lime} \]
  \[ h_3: 50\% \text{ cherry} + 50\% \text{ lime} \]
  \[ h_4: 25\% \text{ cherry} + 75\% \text{ lime} \]
  \[ h_5: 100\% \text{ lime} \]
- Random variable \( H \) denotes the type of the bag and observation variables \( D_1, \ldots, D_n \) are the flavors of opened candies
- The task is to predict the flavor of the next piece of candy

Bayesian learning simply calculates the probability of each hypothesis, given the data \( D \), with observed value \( d \)

Using Bayes’ rule:

\[
P(h_i | d) = \alpha P(d | h_i) P(h_i)
\]

When we want to make a prediction about an unknown quantity \( X \), then we have

\[
P(X | d) = \sum_i P(X | d, h_i) P(h_i | d)
\]

We have assumed that each hypothesis \( h_i \) determines a probability distribution over \( X \)

Predictions are weighted averages over the predictions of individual hypothesis
The key quantities in the Bayesian approach are the **hypothesis prior**, $P(h_i)$, and the **likelihood** of the data under each hypothesis, $P(d \mid h_i)$.

Let the prior distribution of candy bag types $h_1, \ldots, h_5$ be $[0.1, 0.2, 0.4, 0.2, 0.1]$.

We make the **i.i.d.-assumption** (independently and identically distributed) concerning the observations: each observation is independent of the others and is drawn from the same probability distribution, hence

$$P(d \mid h_i) = \prod_j P(d_j \mid h_i)$$

E.g., suppose the bag is really an all-lime bag ($h_5$) and the first 10 candies are all lime; then

$$P(d \mid h_5) = 0.5^{10} \approx 0.001$$

Because $h_3$ has the highest prior probability, it is initially the most likely hypothesis.

Observing one piece of candy with lime flavor, does not yet change the situation, but already after two lime candies $h_4$ becomes the most likely hypothesis.

Starting after three lime observations (the correct bag) $h_5$ is the most likely.

Hence, the correct hypothesis will eventually dominate the prediction.

The predicted probability that the next candy is lime increases monotonically toward 1.

The Bayesian prediction eventually agrees with the true hypothesis.
• Any fixed prior that does not rule out the true hypothesis, the posterior probability of any false hypothesis will (under certain technical conditions) eventually vanish.
• This happens simply because the probability of generating “uncharacteristic” data indefinitely is vanishingly small.
• The Bayesian prediction is optimal: any other prediction is expected to be correct less often (given the hypothesis prior).
• For real learning problems, the hypothesis space is usually very large or infinite.
• In some cases, the summation (or integration, in the continuous case) over the hypothesis class can be carried out tractably, but in most cases we must resort to approximate or simplified methods.

A very common approximation is to make predictions based on a single most probable; i.e., the one that maximizes the value $P(h_i \mid d)$.
• This is often called a maximum a posteriori (MAP) hypothesis $h_{MAP}$.
• As more data arrive, the MAP prediction $P(X \mid h_{MAP})$ and Bayesian prediction $P(X \mid d)$ become closer, because the competitors to MAP hypothesis become less and less probable.
• Instead of a summation (or integration) we now have to solve an optimization problem.
• In our candy bag example, after three pieces of candy $h_{MAP} = h_5$ and the fourth candy is predicted to have lime flavor with probability 1.0, while the true Bayesian probability (averaged over all hypotheses) would be 0.8.
To guard against overfitting, Bayesian and MAP learning methods penalize complex hypotheses with a low prior probability. More complex hypotheses have a greater capacity to fit the data. If, e.g., \( H \) contains only deterministic hypotheses, then \( P(d | h) \) is 1 if \( h \) is consistent and 0 otherwise. Then \( h_{\text{MAP}} \) is, in the spirit of Occam’s razor, the simplest logical theory that is consistent with the data.

On the other hand, choosing \( h_{\text{MAP}} \) to maximize \( P(d | h) P(h) \) is equivalent to minimizing

\[
-\log_2 P(d | h) - \log_2 P(h)
\]

The \( -\log_2 P(h) \) term equals the number of bits required to specify the hypothesis \( h \).

Furthermore, \( -\log_2 P(d | h) \) is the additional number of bits required to specify the data, given the hypothesis. E.g. if the hypothesis predicts the data exactly \( (P(d | h) = 1) \), then no extra bits are required \( (\log_2 1 = 0) \). Hence, MAP learning is choosing the hypothesis that provides maximum compression of the data (cf. MDL principle by Rissanen).

Assuming a uniform prior over the space of hypotheses, reduces MAP learning to choosing an \( h \) that maximizes the likelihood of data \( P(d | h) \).

Maximum-likelihood (ML) learning provides a good approximation to Bayesian and MAP learning when the data set is large, but it has problems with small data sets.
20.2 Learning with Complete Data

- A probability model (Bayesian network) has a fixed structure, we try to determine the values of its numerical parameters (conditional probabilities)
- We assume that the observations are complete; i.e., each data point contains values for every variable in the probability model being learned
- If the proportions of cherry-lime flavor in a candy bag can be arbitrary, there is a continuum of hypotheses
- Let the proportion of cherry candies in a bag be $\theta$, it is the only parameter and the corresponding hypothesis is $h_0$
- The Bayesian network needs to have a node corresponding to a single random variable ($\text{Flavor}$), which can assume values cherry (with probability $\theta$) and lime (with probability $1 - \theta$)

Suppose we unwrap $N$ candies, out of which $c$ pieces turn out to be cherry flavored and $\ell = N - c$ pieces of lime flavor

- We assume all flavor mix ratios to be equally probable a priori

maximum-likelihood approach

$$ P(d | h_0) = \prod_{j=1}^{d} P(d_j | h_0) = \theta^c (1 - \theta)\ell $$

- The maximum-likelihood hypothesis is given by the value $\theta$ that maximizes this expression
- The same value is obtained by maximizing the logarithm of the likelihood (log likelihood)

$$ L(d | h_0) = \log P(d | h_0) = c \log \theta + \ell \log(1 - \theta) $$
• Thus, the product over the data reduces to a sum over the data, which is usually easier to maximize.

• Differentiating $L$ with respect to $\theta$ and setting the resulting expression to zero gives the maximum-likelihood value of $\theta$:

$$\frac{dL(d | h_0)}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1 - \theta} = 0$$

$$\Rightarrow \theta = \frac{c}{c + \ell} = \frac{c}{N}$$

• The maximum-likelihood hypothesis $h_{ML}$ asserts that the actual proportion of cherries in the bag is equal to the observed proportion in the candies unwrapped so far.

• This approach can be used more generally to disclose the values of more than one parameters.

• A significant problem of the approach: if some events have not yet been observed (in a small enough data set), then $h_{ML}$ assigns zero probability to those events.

• Let us change the example so that, depending on the flavor of the candy, it is wrapped by a probabilistic rule either in red or green wrapper.

| Flavor | P(red | F) |
|--------|---------|
| cherry | $\theta_1$ |
| lime   | $\theta_2$ |
• Now the probability model has three parameters $\Theta, \Theta_1,$ and $\Theta_2$.
• From the standard semantics of Bayesian networks, we can compute likelihoods of events, e.g.
  \[ P(\text{Flavor} = \text{cherry}, \text{Wrapper} = \text{green} | h_{\Theta, \Theta_1, \Theta_2}) \]
  \[ = P(\text{Flavor} = \text{cherry} | h_{\Theta, \Theta_1, \Theta_2}) \cdot \]
  \[ P(\text{Wrapper} = \text{green} | \text{Flavor} = \text{cherry}, h_{\Theta, \Theta_1, \Theta_2}) \]
  \[ = \Theta (1 - \Theta_1) \]
• Now we unwrap $N$ candies, of which $c$ are cherries and $\ell$ are limes. The corresponding wrapper counts are $r_c, g_c, r_\ell,$ and $g_\ell$.
• The likelihood of the data $P(d | h_{\Theta, \Theta_1, \Theta_2})$ is
  \[ \Theta^c (1 - \Theta)^\ell \Theta_1^r_c (1 - \Theta_1)^g_c \Theta_2^r_\ell (1 - \Theta_2)^g_\ell \]
• Taking logarithms simplifies the expression:
  \[ L = [c \log \Theta + \ell \log(1 - \Theta)] + [r_c \log \Theta_1 + g_c \log(1 - \Theta_1)] \]
  \[ + [r_\ell \log \Theta_2 + g_\ell \log(1 - \Theta_2)] \]
• When we take derivatives with respect to each parameter and set them to zero, we get
  \[ \frac{\partial L}{\partial \Theta} = c / \Theta - \ell / (1 - \Theta) = 0 \Rightarrow \Theta = c / (c + \ell) \]
  \[ \frac{\partial L}{\partial \Theta_1} = r_c / \Theta_1 - g_c / (1 - \Theta_1) = 0 \Rightarrow \Theta_1 = r_c / (r_c + g_c) \]
  \[ \frac{\partial L}{\partial \Theta_2} = r_\ell / \Theta_2 - g_\ell / (1 - \Theta_2) = 0 \Rightarrow \Theta_2 = r_\ell / (r_\ell + g_\ell) \]
• The solution for $\Theta$ is the same as before. The solution for $\Theta_1$ is the observed fraction of cherry candies with red wrappers, and similarly for $\Theta_2$.
• The maximum-likelihood parameter learning problem for a Bayesian network decomposes into separate learning problems – one for each parameter.