# Deep Learning - Foundations and Concepts

## Chapter 5. Single-layer Networks: Classification

nonlineark@github

February 16, 2025

# Outline

# Discriminant functions

- The goal in classification is to take an input vector $x \in \mathbb{R}^D$ and assign it to one of $K$ discrete classes $\mathcal{C}_k$.
- A discriminant is a function that takes an input vector $x$ and assigns it to one of $K$ classes, denoted $\mathcal{C}_k$.
- We will restrict attention to linear discriminants, for which the decision surfaces are hyperplaines.

# Two classes

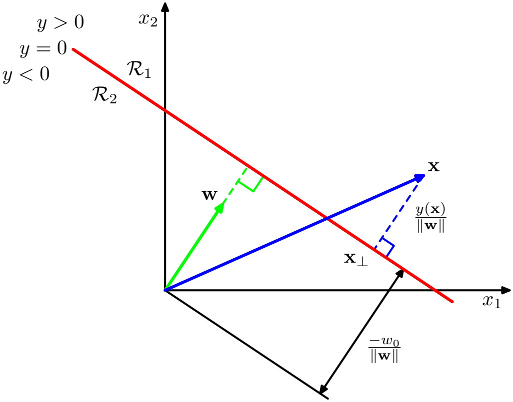
Taking a linear function of the input vector:

$$y(x) = w^T x + w_0$$

- An input vector is assigned to class $\mathcal{C}_1$ if $y(x) \geq 0$ and to class $\mathcal{C}_2$ otherwise.
- The decision boundary is a $(D-1)$-dimensional hyperplane.

# Two classes

Figure: The geometry of a linear discriminant function in two dimensions



# Two classes

It's easy to see that:

- $w$ is orthogonal to the decision surface.
- $w$ points to the direction of the increase of $y$.

Also the value of $y(x)$ gives a signed measure of the perpendicular distance $r$ of the point $x$ from the decision surface:

$$
\begin{aligned}
x &= x_{\perp} + r\frac{w}{||w||} \\
y(x) &= w^T x + w_0 = w^T x_{\perp} + w_0 + r||w|| = r||w|| \\
r &= \frac{y(x)}{||w||}
\end{aligned}
$$

In particular, the signed distance of the origin from the decision surface is given by $\frac{w_0}{||w||}$.

# Multiple classes

Building a $K$-class discriminant by combining a number of two-class discriminant functions usually doens't work:

- One-versus-the-rest.
- One-versus-one.

# Multiple classes

Consider a single $K$-class discriminant comprising $K$ linear functions of the form:

$$y_k(x) = w_k^T x + w_{k0}$$

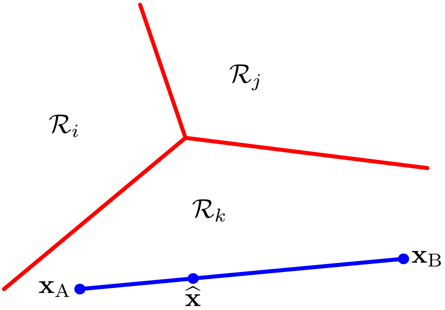
Assign a point $x$ to class $\mathcal{C}_k$ if $y_k(x) > y_j(x)$ for all $j \neq k$. The decision boundary between class $\mathcal{C}_k$ and $\mathcal{C}_j$ is given by $y_k(x) = y_j(x)$ and corresponds to a $(D-1)$-dimensional hyperplane:

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$

The decision regions of such a discriminant are always singly connected and convex.

# Multiple classes

Figure: The decision regions for a multi-class linear discriminant



# Linear squares for classification

Consider a general classification problem with $K$ classes:

- There are $N$ input data: $x^1, \ldots, x^N$, where $x^n \in \mathbb{R}^D$.
- There are $N$ target data: $t^1, \ldots, t^N$ using a 1-of-$K$ binary coding scheme, thus $t^n \in \mathbb{R}^K$.
    - Let $T = \begin{pmatrix} t^1 & t^2 & \ldots & t^N \end{pmatrix}^T \in \mathbb{R}^{N \times K}$.
- Each class $\mathcal{C}_k$ is described by its own linear model so that $y_k(x) = w_k^T x + w_{k0}$.
    - Let $\tilde{w}_k = \begin{pmatrix} w_{k0} \\ w_k \end{pmatrix}$ and $\tilde{W} = \begin{pmatrix} \tilde{w}_1 & \tilde{w}_2 & \ldots & \tilde{w}_K \end{pmatrix} \in \mathbb{R}^{(D+1) \times K}$.
    - Let $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$ and $\tilde{X} = \begin{pmatrix} \tilde{x}^1 & \tilde{x}^2 & \ldots & \tilde{x}^N \end{pmatrix}^T \in \mathbb{R}^{N \times (D+1)}$.
    - Then $y_k(x) = \tilde{w}_k^T \tilde{x}$ and $y(x) = \tilde{W}^T \tilde{x}$.

# Linear squares for classification

Let's determine the parameter matrix $\tilde{W}$ by minimizing a sum-of-squares error function:

$$
\begin{aligned}
E_D(\tilde{W}) &= \frac{1}{2}\sum_{i,j}(\tilde{X}\tilde{W} - T)^2_{ij} \\
&= \frac{1}{2}\mathrm{tr}((\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T)) \\
DE_D(\tilde{W})H &= \mathrm{tr}((\tilde{X}\tilde{W} - T)^T\tilde{X}H) \\
\tilde{W}_* &= (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T
\end{aligned}
$$

# Linear squares for classification

What property does $y(x) = \tilde{W}_*^T \tilde{x}$ has? Because $t^n$ is using a 1-of-$K$ binary coding scheme, we know:

$$\begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} t^n = 1$$

Thus we have:

$$\begin{aligned}
\begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} y(x) &= \begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} \tilde{W}_*^T \tilde{x} \\
&= \begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} T^T \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} \\
&= \begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} \\
&= \mathrm{e}_1^T \tilde{X}^T \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} = \mathrm{e}_1^T \tilde{x} = 1
\end{aligned}$$

That is, the predictions made by the model will have the property that the elements of $y(x)$ will sum to $1$ for any value of $x$.

# Linear squares for classification

- The model outputs cannot be interpreted as probabilities because they are not contrained to lie within the interval $(0, 1)$.
- If the true distribution of the data is markedly different from being Gaussian, the least squares can give poor results.
- Least squares is very sensitive to the presence of outliers (a.k.a., lack robustness).

# Misclassification rate

To minimize the chance of assigning $x$ to the wrong class, intuitively we would choose the class having the higher posterior probability.

- Divide the input space into regions $\mathcal{R}_k$ called decision regions.
- All points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$.

We want to maximize the probability of being correct:

$$p(\text{correct}) = \sum_{k=1}^{K} p(x \in \mathcal{R}_k, \mathcal{C}_k) = \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathcal{C}_k | x) p(x) \mathrm{d}x$$

It's easy to see that this is maximized when the regions $\mathcal{R}_k$ are chosen such that each $x$ is assigned to the class for which $p(\mathcal{C}_k|x)$ is largest. So the intuition is indeed correct.

# Expected loss

- Sometimes, our objective will be more complex than minimizing the number of misclassifications.
- We can introduce a loss function which measure loss incurred in taking any of the available decisions or actions and minimize the total loss.

If the true class for $x$ is $\mathcal{C}_k$ and we assign $x$ to $\mathcal{C}_j$, we incur some level of loss denoted by $L_{kj}$. Because we do not know the true class, instead of minimizing the loss function, we minimize its average:

$$E(L) = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(x, \mathcal{C}_k) \mathrm{d}x = \sum_j \int_{\mathcal{R}_j} \sum_k L_{kj} p(\mathcal{C}_k | x) p(x) \mathrm{d}x$$

The decision rule that minimizes the expected loss assigns $x$ to the class $j$ for which $\sum_k L_{kj} p(\mathcal{C}_k | x)$ is a minimum.

# The reject option

- Classification errors arise when the largest of the posterior probabilities is significantly less than $1$.
- Reject option: Avoid making decisions on such cases to obtain a lower error rate.
- Introduce a threshold $\theta$ and reject inputs $x$ when the largest of the posterior probabilities is less than or equal to $\theta$:
  - $\theta = 1$: All examples are rejected.
  - $\theta < \frac{1}{K}$: No examples are rejected.

# Inference and decision

There are three distinct approaches to solving decision problems:

- Generative models:
  - Solve the inference problem of determining the class-conditional densities $p(x|\mathcal{C}_k)$.
  - Infer the prior class probabilities $p(\mathcal{C}_k)$.
  - Find the posterior class probabilities $p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$.
  - Use decision theory to determine the class membership for each new input $x$.
- Discriminative models:
  - Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k|x)$.
  - Use decision theory to assign each new $x$ to one of the classes.
- Discriminant functions:
  - Find a function that maps each input $x$ directly onto a class label.

# Inference and decision

There are many reasons for wanting to compute the posterior probabilities:

- Minimizing risk: What if the loss matrix are subjected to revision from time to time?
- Reject option.
- Compensating for class priors: What if one class occupies $99.9\%$ of the cases (we want a balanced data set to find a more accurate model)?
- Combining models:
  - Combine the outputs of smaller models use the rules of probability.
  - Models can easily be made differentiable with respect to adjustable parameters, which allows them to be composed and trained jointly.

# Classifier accuracy

Consider a cancer screening example:

- True positive: The classifier predicts that a person has cancer and is correct.
- False positive (type $1$ errors): The classifier predicts that a person has cancer and is wrong.
- True negative: The classifier predicts that a person does not have cancer and is correct.
- False negative (type $2$ errors): The classifier predicts that a person does not have cancer and is wrong.

# Classifier accuracy

$$
\begin{aligned}
\text{Accuracy} &= \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{TN} + N_{FN}} \\
\text{Precision} &= \frac{N_{TP}}{N_{TP} + N_{FP}} \\
\text{Recall} &= \frac{N_{TP}}{N_{TP} + N_{FN}} \\
\text{False positive rate} &= \frac{N_{FP}}{N_{FP} + N_{TN}} \\
\text{False discovery rate} &= \frac{N_{FP}}{N_{FP} + N_{TP}}
\end{aligned}
$$

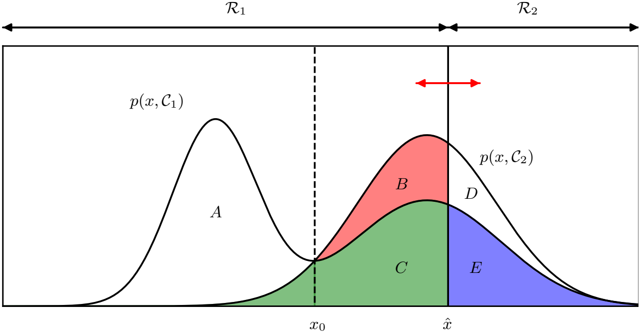
# ROC curve

There is a trade-off between type $1$ errors and type $2$ errors. To better understand this trade-off, it is useful to plot the ROC (receiver operating characteristic) curve:

- $x$-axis: $\text{False positive rate} = \frac{N_{FP}}{N_{FP}+N_{TN}}$.
- $y$-axis: $\text{True positive rate} = \frac{N_{TP}}{N_{TP}+N_{FN}}$.

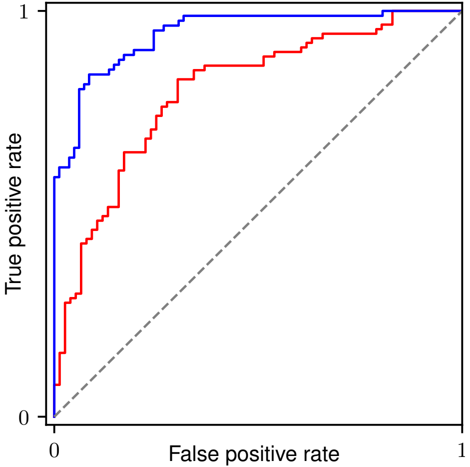
# ROC curve

Figure: As the decision boundary is moved from $\infty$ to $-\infty$, the ROC curve is traced out



# ROC curve

Figure: The ROC (receiver operating characteristic) curve



# ROC curve

Some observations:

- The bottom left corner represents a classifier that always outputs negative.
- The top left corner represents the best possible classifier.
- The top right corner represents a classifier that always outputs positive.
- The diagonal line represents a simple random classifier.

Sometimes it is useful to have a single number that characterises the whole ROC curve:

- The AUC (area under the curve):
  - $0.5$: Random guessing.
  - $1.0$: Perfect classifier.
- The F-score: $F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2N_{TP}}{2N_{TP} + N_{FP} + N_{FN}}$.

# Activation functions

In linear regression, the model prediction is given by:

$$y(x; w) = w^T x + w_0$$

which gives a continuous-valued output in the range $(-\infty, +\infty)$. For classification problems, we wish to predict posterior probabilities in the range $(0, 1)$, which could be achieved using an activation function:

$$y(x; w) = f(w^T x + w_0)$$

We see that the decision surfaces are linear functions of $x$. For this reason, these models are called generalized linear models.

# Activation functions

For two classes, we can use the logistic sigmoid function $\sigma(a) = \frac{1}{1+\exp(-a)}$ as the activation function:

$$p(\mathcal{C}_1|x) = \sigma(a(x)) = \frac{1}{1+\exp(-a(x))}$$

Compare with:

$$\begin{aligned} p(\mathcal{C}_1|x) &= \frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x|\mathcal{C}_1)p(\mathcal{C}_1) + p(x|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(x|\mathcal{C}_2)p(\mathcal{C}_2)}{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}} \end{aligned}$$

We see that:

$$a(x) = \log \frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x|\mathcal{C}_2)p(\mathcal{C}_2)}$$

# Activation functions

The softmax function is defined by:

$$\operatorname{softmax}(a) = \frac{1}{\sum_{k=1}^{K} \exp(a_k)} \begin{pmatrix} \exp(a_1) \\ \vdots \\ \exp(a_K) \end{pmatrix}$$

For multiple classes, we can use the softmax function as the activation function:

$$\begin{pmatrix} p(\mathcal{C}_1|x) \\ \vdots \\ p(\mathcal{C}_K|x) \end{pmatrix} = \operatorname{softmax}(a_1(x), \ldots, a_K(x))$$

Compare with:

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(x|\mathcal{C}_j)p(\mathcal{C}_j)}$$

We see that:

$$a_k(x) = \log(p(x|\mathcal{C}_k)p(\mathcal{C}_k))$$

# Continuous inputs

Let's assume that the class-conditional densities are Guassian. To start with, we will assume that all classes share the same covariance matrix $\Sigma$:

$$p(x|\mathcal{C}_k) = \frac{1}{(2\pi)^{\frac{D}{2}}(\det \Sigma)^{\frac{1}{2}}} \exp(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k))$$

We will see that this lead to generalized linear models.

# Continuous inputs

For two classes:

$$p(\mathcal{C}_1|x) = \sigma(a(x))$$

where:

$$\begin{aligned}
a(x) &= w^T x + w_0 \\
w &= \Sigma^{-1}(\mu_1 - \mu_2) \\
w_0 &= -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \log \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}
\end{aligned}$$

# Continuous inputs

For multiple classes:

$$\begin{pmatrix} p(\mathcal{C}_1|x) \\ \vdots \\ p(\mathcal{C}_K|x) \end{pmatrix} = \mathrm{softmax}(a_1(x), \ldots, a_K(x))$$

where:

$$\begin{aligned} a_k(x) &= w_k^T x + w_{k0} \\ w_k &= \Sigma^{-1}\mu_k \\ w_{k0} &= -\frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \log p(\mathcal{C}_k) \end{aligned}$$

# Maximum likelihood solution

- There are $N$ input data: $x^1, \ldots, x^N$, where $x^n \in \mathbb{R}^D$.
- There are $N$ target data: $t^1, \ldots, t^N$ using a 1-of-$K$ binary coding scheme, thus $t^n \in \mathbb{R}^K$.
- The total number of data points in class $\mathcal{C}_k$ is denoted by $N_k$.
- Prior class probabilities are denoted by $\pi_k = p(\mathcal{C}_k)$.
- Class-conditional densities are Gaussian: $p(x|\mathcal{C}_k) = \mathcal{N}(x; \mu_k, \Sigma)$.

# Maximum likelihood solution

$$
\begin{aligned}
p(t^n|x^n) &= \prod_{k=1}^{K} p(\mathcal{C}_k|x^n)^{t_k^n} = \prod_{k=1}^{K} \left(\frac{p(x^n|\mathcal{C}_k)\pi_k}{p(x^n)}\right)^{t_k^n} \\
&= \frac{1}{p(x^n)} \prod_{k=1}^{K} \pi_k^{t_k^n} \prod_{k=1}^{K} \mathcal{N}(x^n; \mu_k, \Sigma)^{t_k^n} \\
-\log p(t^n|x^n) &= -\sum_{k=1}^{K} t_k^n \log \pi_k \\
&+ \frac{1}{2}\log\det\Sigma + \frac{1}{2}\sum_{k=1}^{K} t_k^n (x^n - \mu_k)^T \Sigma^{-1} (x^n - \mu_k) \\
&+ \log p(x^n) + \frac{D}{2}\log 2\pi
\end{aligned}
$$

# Maximum likelihood solution

$$L = -\log p(t^1, \ldots, t^N | x^1, \ldots, x^N) = -\sum_{n=1}^{N} \log p(t^n | x^n)$$

$$= -\sum_{k=1}^{K} N_k \log \pi_k$$

$$+ \frac{N}{2} \log \det \Sigma + \frac{1}{2} \sum_{k=1}^{K} \sum_{n=1}^{N} t_k^n (x^n - \mu_k)^T \Sigma^{-1} (x^n - \mu_k)$$

$$+ \sum_{n=1}^{N} \log p(x^n) + \frac{ND}{2} \log 2\pi$$

# Maximum likelihood solution

$$\frac{\partial L}{\partial \pi_k} = \frac{N_K}{\pi_K} - \frac{N_k}{\pi_k} \qquad \pi_k = \frac{N_k}{N}$$

$$\frac{\partial L}{\partial \mu_k} = (N_k \mu_k - \sum_{x^n \in \mathcal{C}_k} x^n)^T \Sigma^{-1} \qquad \mu_k = \frac{1}{N_k} \sum_{x^n \in \mathcal{C}_k} x^n$$

$$\frac{\partial L}{\partial \Lambda}(\Lambda) H = \frac{1}{2} \mathrm{tr}((\sum_{k=1}^{K} \sum_{x^n \in \mathcal{C}_k} (x^n - \mu_k)(x^n - \mu_k)^T - N\Sigma) H)$$

$$\Sigma = \sum_{k=1}^{K} \frac{N_k}{N} S_k \qquad S_k = \frac{1}{N_k} \sum_{x^n \in \mathcal{C}_k} (x^n - \mu_k)(x^n - \mu_k)^T$$

# Discrete features

Suppose $x \in \mathbb{R}^D$ is a feature vector, where each feature $x_d \in \{0, 1\}$. And further suppose that the different features are independent when conditioned on the class $\mathcal{C}_k$. So we have:

$$p(x|\mathcal{C}_k) = \prod_{d=1}^{D} p(x_d|\mathcal{C}_k) = \prod_{d=1}^{D} \mu_{dk}^{x_d}(1 - \mu_{dk})^{1-x_d}$$

Using a softmax activation function, we see that:

$$a_k(x) = \sum_{d=1}^{D}(x_d \log \mu_{dk} + (1 - x_d)\log(1 - \mu_{dk})) + \log p(\mathcal{C}_k)$$

which again are linear functions of the input values $x$.

# Exponential family

If the class-conditional densities $p(x|\mathcal{C}_k)$ are members of the subset of the exponential family of distributions given by:

$$p(x|\mathcal{C}_k; \lambda_k, s) = \frac{1}{s} h(\frac{1}{s} x) g(\lambda_k) \exp(\frac{1}{s} \lambda_k^T x)$$

the resulting model will be a generalized linear model. For two classes using the logistic sigmoid activation function:

$$a(x) = \frac{1}{s}(\lambda_1 - \lambda_2)^T x + \log \frac{g(\lambda_1)}{g(\lambda_2)} + \log \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

For multiple classes using the softmax activation function:

$$a_k(x) = \frac{1}{s} \lambda_k^T x + \log g(\lambda_k) + \log p(\mathcal{C}_k)$$

# Discriminative classifiers

- For generative models, we have seen that the posterior probability $p(\mathcal{C}_k|x)$ can be written as a logistic sigmoid or softmax acting on a linear function of $x$, for a wide choice of class-conditional distributions $p(x|\mathcal{C}_k)$ from the exponential family.
- An alternative approach is to maximize a likelihood function defined through the conditional distribution $p(\mathcal{C}_k|x)$ directly.

# Fixed basis functions

Similar to linear regression, we can introduce a vector of nonlinear basis functions $\phi : \mathbb{R}^D \to \mathbb{R}^M$, so that each $x \in \mathbb{R}^D$ in the input space is transformed to a $\phi(x) \in \mathbb{R}^M$ in the feature space:

- Classes that are not linearly separable in the input space may be linearly separable in the feature space.
- One of the basis functions is typically set to a constant, say $\phi_0(x) = 1$, to accommodate the bias.

# Logistic regression

For two-class classification, we model the posterior probability of class $\mathcal{C}_1$ as:

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(a(\phi)) = \sigma(w^T\phi)$$

with $p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$. We see that this logistic regression model has far less parameters than the corresponding generative model:

- For an $M$-dimensional feature space, this logistic regression model has $M$ parameters.
- For comparison, if we fit Gaussian class-conditional densities using maximum likelihood:
  - $2M$ parameters for the mean.
  - $\frac{M(M+1)}{2}$ parameters for the shared covariance.
  - $1$ parameter for the class prior $p(\mathcal{C}_1)$.

# Logistic regression

Using maximum likelihood to determine the parameters:

$$
\begin{aligned}
p(t_n|\phi_n; w) &= y_n^{t_n}(1-y_n)^{1-t_n} \\
\log p(t_n|\phi_n; w) &= t_n \log y_n + (1-t_n)\log(1-y_n) \\
E(w) &= -\log p(t_1, \dots, t_N|\phi_1, \dots, \phi_N; w) \\
&= -\sum_{n=1}^{N}(t_n \log y_n + (1-t_n)\log(1-y_n)) \\
\nabla E(w) &= -\sum_{n=1}^{N}\left(\frac{t_n}{y_n} - \frac{1-t_n}{1-y_n}\right)\frac{\mathrm{d}y_n}{\mathrm{d}a_n}\nabla_w a_n \\
&= \sum_{n=1}^{N}(y_n - t_n)\phi_n
\end{aligned}
$$

# Logistic regression

- $\nabla E(w)$ takes precisely the same form as the gradient of the sum-of-squares error function.
- Due to the nonlinearity in $y$, this equation does not have a closed-form solution.
- One approach to finding a maximum likelihood solution would be to use stochastic gradient descent.
- Maximum likelihood can exhibit severe over-fitting for data sets that are linearly separable, which can be avoided by adding a regularization term to the error function.

# Logistic regression

For multi-class classification, we model the posterior probabilities as:

$$\begin{aligned}
\begin{pmatrix} p(\mathcal{C}_1|\phi) \\ \vdots \\ p(\mathcal{C}_K|\phi) \end{pmatrix} &= \begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} \\
&= \mathrm{softmax}(a_1(\phi), \ldots, a_K(\phi)) \\
&= \mathrm{softmax}(w_1^T\phi, \ldots, w_K^T\phi)
\end{aligned}$$

# Logistic regression

Using maximum likelihood to determine the parameters:

$$p(t^n|\phi_n; w_1, \ldots, w_K) = \prod_{k=1}^{K} (y_k^n)^{t_k^n}$$

$$\log p(t^n|\phi_n; w_1, \ldots, w_K) = \sum_{k=1}^{K} t_k^n \log y_k^n$$

$$E(w_1, \ldots, w_K) = -\log p(t^1, \ldots, t^N|\phi_1, \ldots, \phi_N; w_1, \ldots, w_K)$$

$$= -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^n \log y_k^n$$

$$\nabla_{w_j} E(w_1, \ldots, w_K) = -\sum_{n=1}^{N} (\sum_{k=1}^{K} \frac{t_k^n}{y_k^n} \frac{\partial y_k^n}{\partial a_j^n}) \nabla_{w_j} a_j^n = \sum_{n=1}^{N} (y_j^n - t_j^n) \phi_n$$

# Probit regression

Consider a noisy threshold model for the two-class classification. For input $\phi$, we evaluate $a = w^T\phi$ and then we set the target value according to:

$$\begin{cases} t = 1, & \text{if } a \geq \theta \\ t = 0, & \text{otherwise.} \end{cases}$$

If the value of $\theta$ is drawn from a probability density $p(\theta)$, then we have:

$$p(t = 1|a) = p(a \geq \theta) = p(\theta \leq a) = \int_{-\infty}^{a} p(\theta)\mathrm{d}\theta$$

Thus the activation function is given by $f(a) = \int_{-\infty}^{a} p(\theta)\mathrm{d}\theta$.

# Probit regression

As a specific example, suppose that the density $p(\theta)$ is given by a zero-mean, unit-variance Gaussian:

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta; 0, 1)\mathrm{d}\theta$$

which is known as the probit function. The generalized linear model based on a probit activation function is known as probit regression.

# Canonical link functions

We have seen multiple times that the derivative to the parameter $w$ of the contribution to the error function from a data point $n$ takes the form $\nabla E_n(w) = (y_n - t_n)\phi_n$:

- In linear regression, using the sum-of-squares error and the identity activation function.
- In two-class classification, using the cross-entropy error and the logistic sigmoid activation function.
- In multi-class classification, using the cross-entropy error and the softmax activation function.

We will see that this is not by chance.

# Canonical link functions

Consider conditional distribution of the target variable of the form:

$$p(t|\eta; s) = \frac{1}{s} h(\frac{t}{s}) g(\eta) \exp(\frac{\eta t}{s})$$

Now let's suppose we have this chain of relations:

- $a = w^T \phi$ is the pre-activation, where $w$ is the parameter vector.
- $y = f(a)$ is the post-activation, where $f$ is the activation function.
- $\eta = \psi(y)$ has the property that $y = E(t|\eta)$.

# Canonical link functions

$$
\begin{aligned}
E(w) &= -\log p(t_1, \ldots, t_N | \eta_1, \ldots, \eta_N; s) \\
&= N \log s - \sum_{n=1}^{N} \log h(\frac{t_n}{s}) - \sum_{n=1}^{N} \log g(\eta_n) - \frac{1}{s} \sum_{n=1}^{N} \eta_n t_n \\
\nabla E(w) &= \sum_{n=1}^{N} (-\frac{g'(\eta_n)}{g(\eta_n)} - \frac{t_n}{s}) \frac{\mathrm{d}\eta_n}{\mathrm{d}y_n} \frac{\mathrm{d}y_n}{\mathrm{d}a_n} \nabla a_n \\
&= \frac{1}{s} \sum_{n=1}^{N} (E(t|\eta_n) - t_n) \psi'(y_n) f'(a_n) \phi_n \\
&= \frac{1}{s} \sum_{n=1}^{N} (y_n - t_n) \psi'(y_n) f'(a_n) \phi_n
\end{aligned}
$$

# Canonical link functions

We see that if we select the activation function $f$ so that $f^{-1} = \psi$, then:

$$\psi'(y_n) f'(a_n) = 1$$

and we have:

$$\nabla E(w) = \frac{1}{s} \sum_{n=1}^{N} (y_n - t_n) \phi_n$$