Machine Learning I Lecture VII: Logistic Regression

Jakob H Macke
Max Planck Institute for Biological Cybernetics
Bernstein Center for Computational Neuroscience

November 30, 2012

Plan for today

Logistic Regression

Maximum likelihood estimation of Logistic Regression

Bayesian Logistic Regression: Approximating the posterior distribution

For the linear classification model, we assumed the class-conditional distributions to be Gaussian

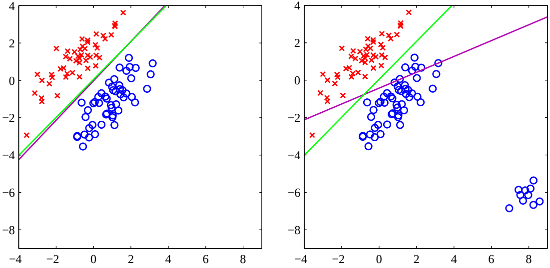
- We assumed $x|(t=1) \sim \mathcal{N}(\mu_+, \Sigma_+)$ and $x|(t=-1) \sim \mathcal{N}(\mu_-, \Sigma_-)$, and two class-probabilities $P(t=1)$ and $P(t=-1)$.

For the linear classification model, we assumed the class-conditional distributions to be Gaussian

- We assumed $x|(t = 1) \sim \mathcal{N}(\mu_+, \Sigma_+)$ and $x|(t = -1) \sim \mathcal{N}(\mu_-, \Sigma_-)$, and two class-probabilities $P(t = 1)$ and $P(t = -1)$.
- This is called an generative model, as we have written down a full joint model over the data.

For the linear classification model, we assumed the class-conditional distributions to be Gaussian

- We assumed $x|(t=1) \sim \mathcal{N}(\mu_+, \Sigma_+)$ and $x|(t=-1) \sim \mathcal{N}(\mu_-, \Sigma_-)$, and two class-probabilities $P(t=1)$ and $P(t=-1)$.
- This is called an generative model, as we have written down a full joint model over the data.
- We saw that violations of the model assumption can lead to 'bad' decision boundaries.



For regression, we assumed Gaussian outputs, but did not need assumptions about the distribution of inputs.

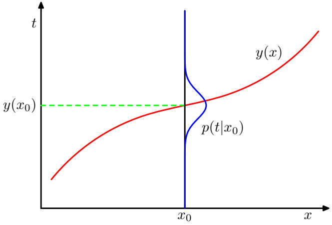
- For linear regression, we conditioned on x, and assumed a Gaussian distribution over t: $t|x \sim \mathcal{N}(y(x), \gamma^2)$

For regression, we assumed Gaussian outputs, but did not need assumptions about the distribution of inputs.

- For linear regression, we conditioned on x, and assumed a Gaussian distribution over t: $t|x \sim \mathcal{N}(y(x), \gamma^2)$
- We maximized the conditional log-likelihood $L(\omega) = \sum_n \log p(t_n|x_n, \omega)$, i.e we assumed that the x were given.

For regression, we assumed Gaussian outputs, but did not need assumptions about the distribution of inputs.

- For linear regression, we conditioned on x, and assumed a Gaussian distribution over t: $t|x \sim \mathcal{N}(y(x), \gamma^2)$
- We maximized the conditional log-likelihood $L(\omega) = \sum_n \log p(t_n|x_n, \omega)$, i.e we assumed that the x were given.
- Therefore, this approach to linear regression works for any distribution over x.
- x is typically high-dimensional, so it is difficult to make appropriate distributional assumptions for it.



We can define a discriminative model for classification by modelling the conditional class probabilities.

- From the homework-exercise, we know that $P(t = 1|z(x)) = \sigma(z(x))$ where $\sigma(z) = 1/(1 + \exp(-z))$ and $z(x) = \omega^\top x + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $s_n = 1$ as the label for the positive class, and $s_n = 0$ als label for the negative class.
- In other words, $s|x \sim \text{Bernoulli}(\sigma(y(x))$.
- Also, we set $y_n = \sigma(z(x))$.

We can define a discriminative model for classification by modelling the conditional class probabilities.

- From the homework-exercise, we know that $P(t = 1|z(x)) = \sigma(z(x))$ where $\sigma(z) = 1/(1 + \exp(-z))$ and $z(x) = \omega^\top x + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $s_n = 1$ as the label for the positive class, and $s_n = 0$ als label for the negative class.
- In other words, $s|x \sim \text{Bernoulli}(\sigma(y(x))$.
- Also, we set $y_n = \sigma(z(x))$.
- The parameters of $z(x) = \omega^\top x + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\omega) = \sum_n \log p(t_n|x_n, \omega)$ [on board]

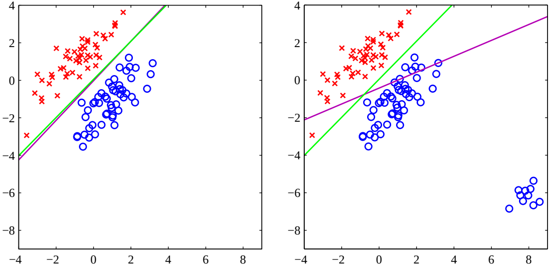
We can define a discriminative model for classification by modelling the conditional class probabilities.

- From the homework-exercise, we know that $P(t = 1|z(x)) = \sigma(z(x))$ where $\sigma(z) = 1/(1 + \exp(-z))$ and $z(x) = \omega^\top x + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $s_n = 1$ as the label for the positive class, and $s_n = 0$ als label for the negative class.
- In other words, $s|x \sim \text{Bernoulli}(\sigma(y(x))$.
- Also, we set $y_n = \sigma(z(x))$.
- The parameters of $z(x) = \omega^\top x + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\omega) = \sum_n \log p(t_n|x_n, \omega)$ [on board]
- This is an discriminative approach to classification, as we only model the labels, and not the inputs.

We can define a discriminative model for classification by modelling the conditional class probabilities.

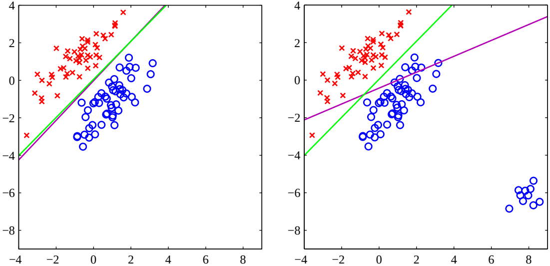
- From the homework-exercise, we know that $P(t = 1|z(x)) = \sigma(z(x))$ where $\sigma(z) = 1/(1 + \exp(-z))$ and $z(x) = \omega^\top x + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $s_n = 1$ as the label for the positive class, and $s_n = 0$ als label for the negative class.
- In other words, $s|x \sim \mathrm{Bernoulli}(\sigma(y(x)))$.
- Also, we set $y_n = \sigma(z(x))$.
- The parameters of $z(x) = \omega^\top x + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\omega) = \sum_n \log p(t_n|x_n, \omega)$ [on board]
- This is an discriminative approach to classification, as we only model the labels, and not the inputs.
- Decision rule and function shape of $p(t|x)$ will be the same for the generative ('Linear Discriminant Analysis') and the discriminative model, but the parameters were obtained differently.

Maximum likelihood estimation of Logistic Regression



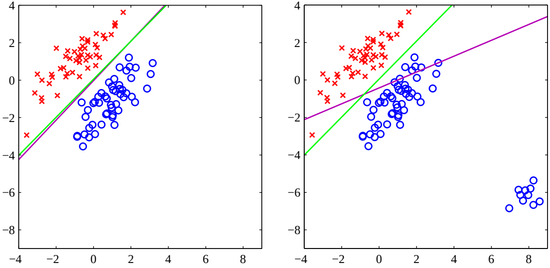
- This algorithm is called logistic regression, and is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.

Maximum likelihood estimation of Logistic Regression



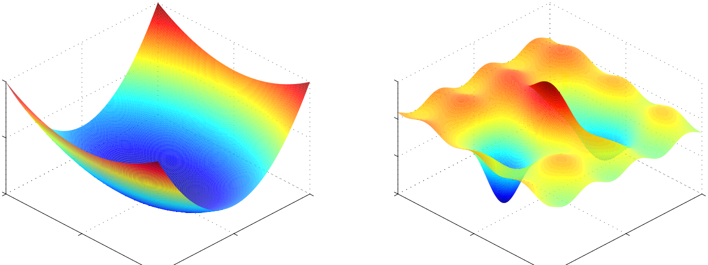
- This algorithm is called logistic regression, and is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.
- People typically minimize the negative log-likelihood $\mathcal{L}$ rather than maximize the log-likelihood...

Maximum likelihood estimation of Logistic Regression

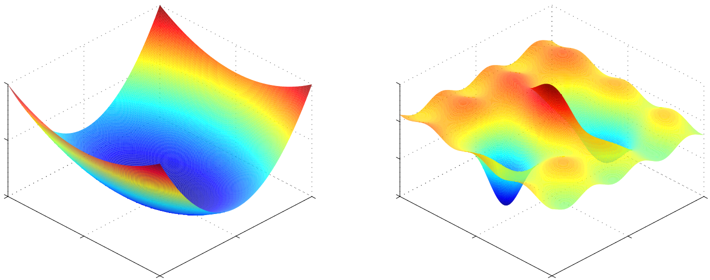


- This algorithm is called logistic regression, and is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.
- People typically minimize the negative log-likelihood $\mathcal{L}$ rather than maximize the log-likelihood...
- To numerically minimize the negative log-likelihood, we need its gradient (and maybe its hessian) [on board]

The cost-function for logistic regression is convex.

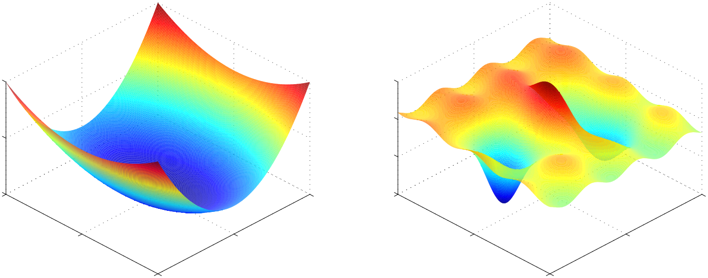


The cost-function for logistic regression is convex.



- Fact: The negative log-likelihood is *convex* – this makes life much more easier.

The cost-function for logistic regression is convex.



- Fact: The negative log-likelihood is *convex* – this makes life much more easier.
- There are no local minima to get stuck in, and there is good optimization techniques for convex problems.

Gradient descent is a simple method for numerically minimizing a function.

- The gradient $\nabla\mathcal{L}$ of a function points into the direction of steepest descent.

Gradient descent is a simple method for numerically minimizing a function.

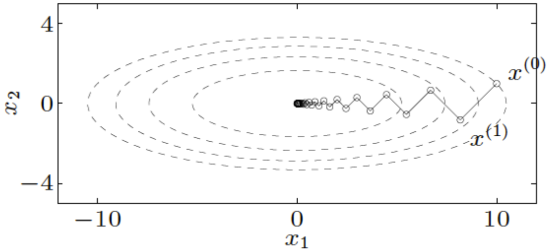
- The gradient $\nabla\mathcal{L}$ of a function points into the direction of steepest descent.
- Gradient descent: 'run down the gradient' $\omega_{new} = \omega_{old} - \alpha\nabla\mathcal{L}_{\omega}$, with learning rate α.

Gradient descent is a simple method for numerically minimizing a function.

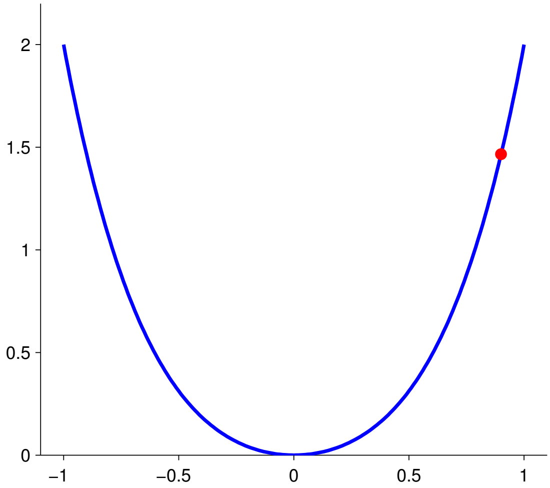
- The gradient $\nabla\mathcal{L}$ of a function points into the direction of steepest descent.
- Gradient descent: 'run down the gradient' $\omega_{new} = \omega_{old} - \alpha\nabla\mathcal{L}_\omega$, with learning rate α.
- Slightly more sophisticated version: numerically optimize α for each step by doing a *line search*.

Gradient descent is a simple method for numerically minimizing a function.

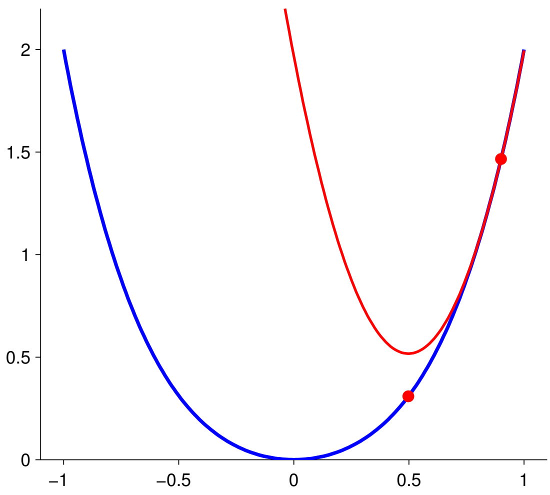
- The gradient $\nabla\mathcal{L}$ of a function points into the direction of steepest descent.
- Gradient descent: 'run down the gradient' $\omega_{new} = \omega_{old} - \alpha\nabla\mathcal{L}_\omega$, with learning rate α.
- Slightly more sophisticated version: numerically optimize α for each step by doing a *line search*.
- Convergence can be very slow if cost-function has 'valleys'.



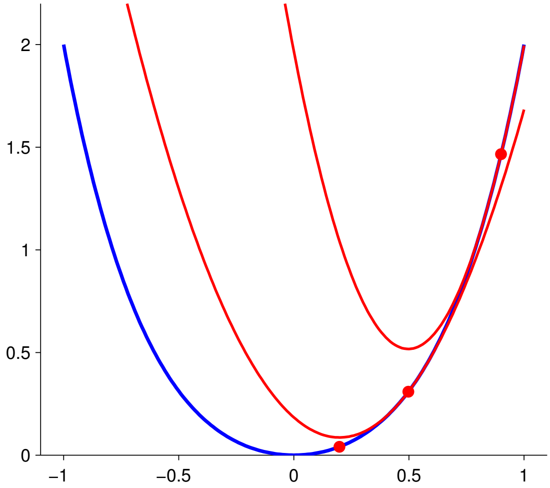
Iterative Least Squares: Approximate by parabola, minimze, iterate.



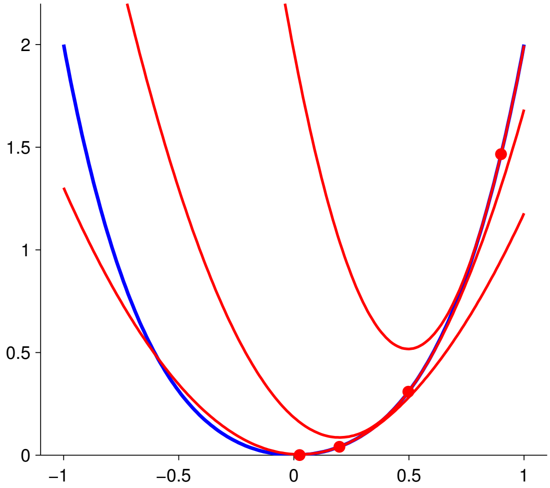
Iterative Least Squares: Approximate by parabola, minimze, iterate.



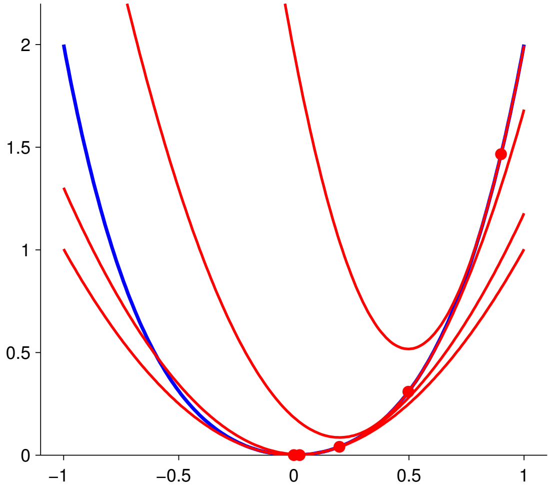
Iterative Least Squares: Approximate by parabola, minimze, iterate.



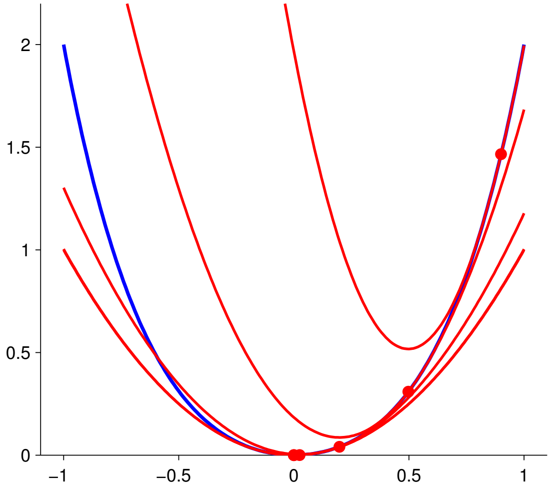
Iterative Least Squares: Approximate by parabola, minimze, iterate.



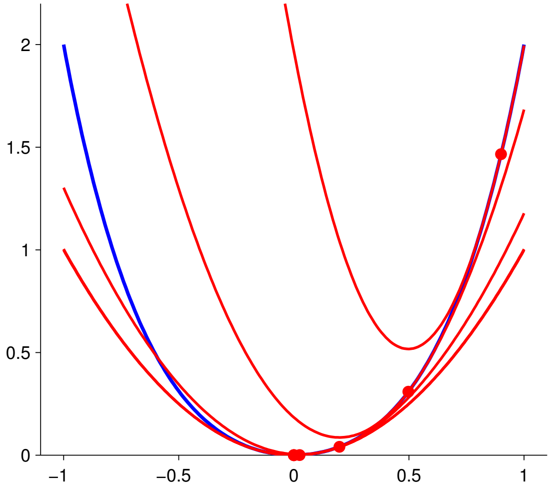
Iterative Least Squares: Approximate by parabola, minimze, iterate.



Iterative Least Squares: Approximate by parabola, minimze, iterate.



Iterative Least Squares: Approximate by parabola, minimze, iterate.



Iterative Least Squares is a more efficient method for minimizing the cost-function

- Newton-Raphson: $\omega_{new} = \omega_{old} - \alpha(\nabla\nabla\mathcal{L})^{-1}\nabla L_\omega$
- Pre-multiplying the gradient by the inverse-hessian speeds up convergence 'along valleys' (analogy with LDA)
- Motivation: For quadratic functions $F(x) = a + b^\top x + x^\top Bx$, Newton-Raphson finds the minimum in one iteration.
- In this contex, Newton-Raphson (with $\alpha = 1$) is often called iterative least squares.
- Note: Newtwn's method can be bad if problem is not convex, and can be slow if it is difficult to calculate/invert the Hessian. A large number of optimization algorithms exist which do not require the (complete) Hessian (quasi Newton/BFGS, etc..).

Visualizing the cost-function of logistic regression

[on board]

Bayesian inference for this model does not have a closed form solution

- Typically use Gaussian prior on ω.
- For linear regression, posterior distribution was Gaussian, with closed-form solutions for the mean and covariance.
- For logistic regression, the posterior distribution is non-Gaussian.

Bayesian inference for this model does not have a closed form solution

- Typically use Gaussian prior on ω.
- For linear regression, posterior distribution was Gaussian, with closed-form solutions for the mean and covariance.
- For logistic regression, the posterior distribution is non-Gaussian.
- Popular approximation: Approximate posterior by a Gaussian

$$p(\omega|D) \approx \mathcal{N}(\mu_{post}, \Sigma_{post}) \tag{1}$$

Bayesian inference for this model does not have a closed form solution

- Typically use Gaussian prior on ω.
- For linear regression, posterior distribution was Gaussian, with closed-form solutions for the mean and covariance.
- For logistic regression, the posterior distribution is non-Gaussian.
- Popular approximation: Approximate posterior by a Gaussian

$$p(\omega|D) \approx \mathcal{N}(\mu_{post}, \Sigma_{post}) \tag{1}$$

- Different methods exist for finding 'good' μ_{post} and Σ_{post}: Expectation Propagation (EP), Laplace Approximation, Variational Inference

The Laplace-Approximation is a simple Gaussian approximation to the posterior

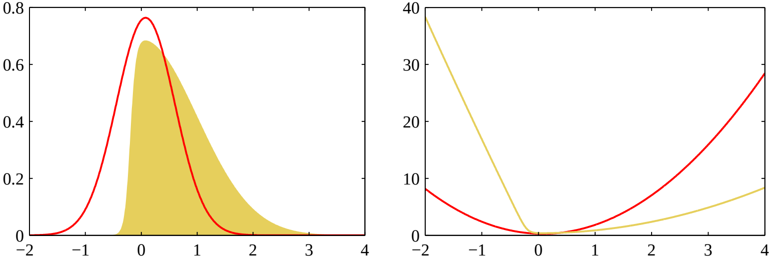
- Laplace approximation: $\mu_{post} = \omega_{MAP}$, $\Sigma_{post} = (\nabla\nabla_{\omega} L)^{-1}$.
- Take MAP as mean, and inverse hessian at MAP as covariance.

The Laplace-Approximation is a simple Gaussian approximation to the posterior

- Laplace approximation: $\mu_{post} = \omega_{MAP}$, $\Sigma_{post} = (\nabla\nabla_\omega L)^{-1}$.
- Take MAP as mean, and inverse hessian at MAP as covariance.
- Motivation: Curvature matching, Taylor-expansion [on board]

The Laplace-Approximation is a simple Gaussian approximation to the posterior

- Laplace approximation: $\mu_{post} = \omega_{MAP}$, $\Sigma_{post} = (\nabla\nabla_\omega L)^{-1}$.
- Take MAP as mean, and inverse hessian at MAP as covariance.
- Motivation: Curvature matching, Taylor-expansion [on board]
- Q: When will the Laplace approximation fail?



The posterior distribution can be used to calculate the predictive distribution and to optimze hyper-parameters

[on board]

One last bit of business: The exam

- Next Friday, 2pm *sharp*.
- You will have 90 minutes.
- You are allowed to use a pen or other writing utensils and your brain, no other tools/materials/books/notes will be allowed. All mobiles phones need to be switched off.
- Master-Students: Graded
- Everyone else: Pass/Fail. If you want a grade for whatever reason, let me know (but tit might not have any official meaning).

This is the end.

Have fun in the second half of the course!

This is the end.

Have fun in the second half of the course!

If you did like (some bits) from these lectures ...

... and want to do a lab-rotation on using machine-learning methods to analyse neural data and to model neural population dynamics

... write an email to jakobtuebingen.mpg.de.